Chapter 5
Optimal Estimation

Part 3
5.3 State Space Kalman Filters
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
Rudolph. E. Kalman

• Born in Budapest, Hungary, on May 19, 1930.
• “Magnetic personality”
• Did EE at MIT
• Professor at Stanford U
Impact

• One of the greatest and broadly applied discoveries in the history of statistical estimation theory.

• Navigation and Guidance Applications
  – Robotics
  – Aircraft
  – Automobiles
  – Spacecraft orbit determination
Impact

• Control and Estimation Applications
  – Continuous manufacturing processes (Power, Chemical)
  – Target tracking
  – Computer vision
  – Economic Forecasting
  – Stock Market Prediction !!!
Impact

• Subsystems Within Robotics
  – Perception,
  – Localization
  – Control

• Subproblems of Robotics
  – State estimation
  – Data association
  – Calibration, system identification

• Trade studies
  – Built-in simulation
Characterization

• Usually, the situation is more generic with measurements that are:
  – incomplete: related to some but not all of the variables of interest
  – indirect: related indirectly to the quantities of interest
  – intermittent: available at irregularly-spaced instants of time

• Also, the state vector of interest may be
  – changing with respect to time.

• The Kalman Filter can handle all of this.
Characterization

• An algorithm. Not hardware.
• Recursively estimates state of a dynamic system from noisy data.
  – System dynamics perturbed by white noise.
  – Measurements perturbed by white noise.
• For optimal (or even correct) results, errors must be:
  – Unbiased (have zero mean for all time)
  – Gaussian (have a Gaussian distribution for all time)
  – White (contain all frequencies)
5.3.1 Introduction

• Recall the form of **state space model** of a system:

\[
\dot{x} = Fx + Gw \\
z = Hx + v
\]
5.3.1 Overall Operation

Kalman Filter
- System model
- Measurement model

Physical System
System State $x(t)$
System Noise $v(t)$

Measurement Process
Measurement Noise $w(t)$
Measurement $z(t)$

Kalman Filter
Initial Estimate $\hat{x}(t)$
State Estimate $\hat{x}(t)$
5.3.1 Additional Capabilities of SS KF

- An SS KF can:
  - Predict state *between* and *beyond* the measurements.
  - Use *rate measurements* that are derivatives of required state variables.
  - Explicitly account for *modeling assumptions* and disturbances in a more precise way than just “noise”.
  - *Identify* a system (calibrate parameters) in real-time.
  - Correlations that it tracks make it possible to *remove effects of historical errors* once they become known.
5.3.1.1 Need for State Prediction

• Let subscripts denote times thus thus:

\[ x_1 = x(t_1) \quad \text{and} \quad z_2 = z(t_2) \]

• Not all of the difference between \( x_1 \) and \( x_2 \) is now due to error. Some of it is motion.

• Must compute \( x_2 \) from \( x_1 \) and then compare \( z_2 \) to that.

• That also involves predicting the error in the prediction \( \rightarrow \) recall how error compounds in dead reckoning.
5.3.1.3 Discrete Time System Model

• Continuous Time:

\[
\dot{x} = Fx + Gw \\
z = Hx + v
\]

State or Process Model
Measurement or Observation Model

• Discrete Time:

\[
\hat{x}_{k+1} = \Phi_k \hat{x}_k + G_k w_k \\
z_k = H_k \hat{x}_k + v_k
\]

State or Process Model
Measurement or Observation Model

Continuous form rarely used in practice
# Nomenclature

<table>
<thead>
<tr>
<th>Object</th>
<th>Size</th>
<th>Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}_k$</td>
<td>$n \times 1$</td>
<td>state vector estimate at time $t_k$</td>
<td></td>
</tr>
<tr>
<td>$\Phi_k$</td>
<td>$n \times n$</td>
<td>transition matrix</td>
<td>relates $x_k$ to $x_{k+1}$ in the absence of a forcing function</td>
</tr>
<tr>
<td>$G_k$</td>
<td>$n \times n$</td>
<td>process noise distribution matrix</td>
<td>transforms the $w_k$ vector into the coordinates of $x_k$</td>
</tr>
<tr>
<td>$w_k$</td>
<td>$n \times 1$</td>
<td>disturbance sequence or process noise sequence</td>
<td>white, known covariance structure</td>
</tr>
<tr>
<td>$z_{-k}$</td>
<td>$m \times 1$</td>
<td>measurement at time $t_k$</td>
<td></td>
</tr>
<tr>
<td>$H_k$</td>
<td>$m \times n$</td>
<td>measurement matrix or observation matrix</td>
<td>relates $x_k$ to $z_k$ in the absence of measurement noise</td>
</tr>
<tr>
<td>$v_{-k}$</td>
<td>$m \times 1$</td>
<td>measurement noise sequence</td>
<td>white, known covariance structure</td>
</tr>
</tbody>
</table>

$n = \# \text{ states} \quad m = \# \text{ measurements}$
5.3.1.3 Noises

• Assume:
  – Process and measurement noises are white (uncorrelated with themselves in time).
  – Uncorrelated with each other.

\[
E(w_k w_i^T) = \delta_{ik} Q_k \\
E(v_k v_i^T) = \delta_{ik} R_k \\
E(w_k v_i^T) = 0, \forall (i, k)
\]
5.3.1.4 Transition Matrix

- Converts continuous time ODEs to discrete time ones:

- The time continuous, matrix ODE:

  \[ \dot{x} = F(t)x \]

- Can always be converted to:

  \[ x_{k+1} = \Phi_k x_k \]

- But it may not be easy.
5.3.1.4 Matrix Exponential

When $F(t)$ is actually time-independent ($F$):

$$\Phi_k = e^{F\Delta t} = I + F\Delta t + \frac{(F\Delta t)^2}{2!} + \ldots$$

Don’t panic! Its just adds and multiplies, ah..., forever.

For time varying $F(t)$, even when $\Delta t$ is sufficiently small relative to system time constants, can use:

$$\Phi_k \approx e^{F\Delta t} \approx I + F\Delta t$$
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
5.3.2.1 The Filter Equations – 2 Sets

The Kalman filter equations for the linear system model are as follows:

<table>
<thead>
<tr>
<th>System Model</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predict state</td>
<td>( \hat{x}_{k+1} = \Phi_k \hat{x}_k )</td>
</tr>
<tr>
<td>Predict covariance</td>
<td>( P_{k+1} = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T )</td>
</tr>
</tbody>
</table>

\[ K_k = P_k H_k^T [H_k P_k H_k^T + R_k]^{-1} \]  
compute Kalman gain

\[ \hat{x}_k^+ = \hat{x}_k^- + K_k [z_k - H_k \hat{x}_k^-] \]  
update state estimate

\[ P_k^+ = [I - K_k H_k] P_k^- \]  
update its covariance

+ means “after incorporation of measurement into estimate”
5.3.2.2 Time and Updates

- System model runs continuously (i.e. at high rates).
- Kalman filter runs when measurements are available.
5.3.2.3 Interpreting Uncertainty Matrices

- **Q<sub>k</sub>:**
  - you provide this
  - instantaneous uncertainty which corrupts the system model
  - random physical disturbances and process model errors

- **R<sub>k</sub>:**
  - you provide this too
  - instantaneous uncertainty which corrupts the measurement model
  - random errors in sensor outputs

- **P<sub>k</sub>:**
  - Filter mostly manages. You provide only P0
  - total integrated uncertainty in state estimate
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
Linearizing Nonlinear Problems

• Full nonlinear model:

\[ \dot{x} = f(x, t) + g(t)w(t) \]
\[ z = h(x, t) + v(t) \]

• Linearize about a reference trajectory \( x^*(t) \)

\[ \Delta\dot{x} = \frac{\partial f}{\partial x}(x^*, t)\Delta x + g(t)w(t) \]
\[ z - h(x^*, t) = \frac{\partial h}{\partial x}(x^*, t)\Delta x + v(t) \]
Linear (Feedforward) Kalman Filter

- **Does not** update the reference trajectory:

- State vector is the errors.
- **Advantage**: more responsive to dynamics (computed in reference trajectory).
- **Disadvantage**: diverges more quickly.
Extended Kalman Filter

- **Does** update the reference trajectory:

- State vector is the state.
- **Disadvantage**: less responsive to dynamics.
- **Advantage**: diverges less quickly.
Extended Kalman Filter

• Kalman Filter:
  – Jacobians:
  – Compute Kalman gain:
  – Update state estimate:
  – Update its covariance:

\[
F_k = \frac{\partial f}{\partial \tilde{x}}(\hat{x}_k) \quad G_k = \frac{\partial g}{\partial \tilde{w}}(\hat{x}_k) \quad H_k = \frac{\partial h}{\partial \tilde{x}}(\hat{x}_k)
\]

\[
K_k = P_k^+H_k^T[H_kP_k^+H_k^T + R_k]^{-1}
\]
\[
\hat{x}_k^+ = \hat{x}_k + K_k[\tilde{z}_k - h(\tilde{x}_k)]
\]
\[
P_k^+ = [I - K_kH_k]P_k^-
\]

• System Model:
  – Project state:
  – Project covariance:

\[
\hat{x}_{k+1} = \phi_k(\hat{x}_k)
\]
\[
P_{k+1} = \Phi_kP_k\Phi_k^T + G_kQ_kG_k^T
\]

These are the ones you will use for almost any filter.
State Transition – Nonlinear Problems

• When the system model is nonlinear:
  \[ \dot{x} = f(x(t), t) \]

• The previous expression:
  \[ \hat{x}_{k+1} = \phi_k(\hat{x}_k) \]

• Is just code for “solve the ODE”. The “transition matrix” can be generated from time linearization:
  \[ \bar{x}_{k+1} = \bar{x}_k + f(\bar{x}_k, t_k) \Delta t \]
Uncertainty Propagation – Nonlinear Problems

• The state covariance propagation is:

\[ P_{k+1} = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T \]

• This approximation can be used:

\[ \Phi_k = I + F \Delta t \]
System Identification

- A poorly known constant can be computed automatically if there are enough measurements to observe it.
- Its “state equation” is: $\dot{x}_i = 0$
- Just add it to the state vector and make sure to update $H$ to encode how measurement error depends linearly on its error.
5.3 State Space Kalman Filters

- 5.3.1 Introduction
- 5.3.2 Linear Discrete Time Kalman Filter
- 5.3.3 Kalman Filters for Nonlinear Systems
- 5.3.4 Simple Example: 2D Mobile Robot
- 5.3.5 Pragmatic Information for Kalman Filters
- 5.3.6 Other Forms of the Kalman Filter
- Summary
5.3.4 2D Mobile Robot Filter

- **State Vector:** 
  
  \[ x = \begin{bmatrix} x & y & \psi & v & \omega \end{bmatrix}^T \]

  - Care about this
  - Need this to propagate state

- **Measurements**

  \[ z = \begin{bmatrix} z_e & z_g \end{bmatrix}^T \]

  - Transmission Encoder
  - Gyro

![Diagram of a mobile robot with labels for x, y, \( \psi \), v, and \( \omega \)]
5.3.4.1 System and Measurement Model (System Model)

• Generally of the form:

\[ \dot{x} = \frac{dx}{dt} = f(x, t) \]

• Here, it is:

\[ \dot{x} = \frac{dx}{dt} = f(x, t) \Rightarrow \frac{d}{dt} [x \ y \ \psi \ \nu \ \omega]^T \]

\[ \dot{x} = \begin{bmatrix} \nu c\psi & \nu s\psi & \omega & 0 & 0 \end{bmatrix}^T \]

• Assumes constant velocity between measurements, but no worries because:
  – Measurements can change velocity.
  – Measurements may arrive at 100 Hz.
5.3.4.1 System and Measurement Model (System Jacobian)

- Recall:
  \[ \dot{x} = \begin{bmatrix} v_c \psi & v_s \psi & \omega & 0 & 0 \end{bmatrix}^T \]

- "Clearly":
  \[ F = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial \omega} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial \omega} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} & \frac{\partial \theta}{\partial v} & \frac{\partial \theta}{\partial \omega} \\ \frac{\partial \dot{v}}{\partial x} & \frac{\partial \dot{v}}{\partial y} & \frac{\partial \dot{v}}{\partial \theta} & \frac{\partial \dot{v}}{\partial v} & \frac{\partial \dot{v}}{\partial \omega} \\ \frac{\partial \dot{\omega}}{\partial x} & \frac{\partial \dot{\omega}}{\partial y} & \frac{\partial \dot{\omega}}{\partial \theta} & \frac{\partial \dot{\omega}}{\partial v} & \frac{\partial \dot{\omega}}{\partial \omega} \end{bmatrix} \]

\[
\begin{bmatrix} 0 & 0 & -v_s \psi & c \psi & 0 \\ 0 & 0 & v_c \psi & s \psi & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\]
5.3.4.2 Discretize and Linearize

• Linearize:

\[ x_{k+1} \approx x_k + f(x, t) \Delta t \]

\[
\begin{bmatrix}
  x_{k+1} \\
  y_{k+1} \\
  \psi_{k+1} \\
  v_{k+1} \\
  \omega_{k+1}
\end{bmatrix}
\approx
\begin{bmatrix}
  x_k \\
  y_k \\
  \psi_k \\
  v_k \\
  \omega_k
\end{bmatrix}
+ \begin{bmatrix}
  v_k c \psi_k \\
  v_k s \psi_k \\
  \omega_k \\
  0 \\
  0
\end{bmatrix}\Delta t_k
\]

• This is a linearized (called "Euler") approximation.

• Express in matrix form:

\[ x_{k+1} \approx \Phi x_k \]

\[
\Phi \approx \begin{bmatrix}
  1 & 0 & 0 & c\psi \Delta t & 0 \\
  0 & 1 & 0 & s\psi \Delta t & 0 \\
  0 & 0 & 1 & 0 & \Delta t \\
  0 & 0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

THIS IS NOT $\Phi$!

• Maybe its easier to code this.
5.3.4.2 Discretize and Linearize (State Uncertainty Propagation)

- Recall, its of the form: 
  \[ P_{k+1}^- = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T \]

- We approximate the transition matrix with:

\[ \Phi_k \approx I + F \Delta t \]

\[
\Phi_k \approx \begin{bmatrix}
1 & 0 & -v_s \psi \Delta t & c \psi \Delta t & 0 \\
0 & 1 & v_c \psi \Delta t & s \psi \Delta t & 0 \\
0 & 0 & 1 & 0 & \Delta t \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Note difference from F matrix 2 slides ago!
5.3.4.3 Initialization

• Be careful with $P_0$:
  – Too little $P_0$ and measurements will be ignored.
  – Too much $P_0$ and numerical problems.

• Here assume:

\[
P_0 = \text{diag}\begin{bmatrix}
\sigma_{xx} & \sigma_{yy} & \sigma_{\psi\psi} & \sigma_{vv} & \sigma_{\omega\omega}
\end{bmatrix}
\]

Means: the matrix whose diagonal is this vector
5.3.4.4 System Disturbances

• Error growth between measurements

\[ G_k Q_k G_k^T \]

• Use it to capture:
  – Incorrectness of flat terrain assumption.
  – Incorrectness of no Wheel slip assumption.
  – Incorrectness of constant velocity assumption.

• Would like it to be larger for larger \( \Delta t \).
• In the absence of real data, try something related to the Taylor remainder
  – First neglected term in dynamics linearization.
5.3.4.4 System Disturbances

• Try:

\[ Q_k = \text{diag}[k_{xx}, k_{yy}, k_{\psi\psi}, k_{vv}, k_{\omega\omega}] \Delta t \]

• But what is \( G_k \) ?

• Let \( k_{xx} \) and \( k_{yy} \) be interpreted in the body frame to allow asymmetric error magnitudes in direction of travel.

• Then \( G_k \) converts coordinates:

\[
G = \begin{bmatrix}
c\psi & -s\psi & 0 & 0 & 0 \\
s\psi & c\psi & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
OK. Breathe. We’re 1/4 Done 😞

We have the dynamics ...

\[
\hat{x}_{k+1} = \phi_k(\hat{x}_k)
\]
\[
P_{k+1} = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T
\]
5.3.4.5.1 Transmission Encoder Measurement Model

- "Velocity" encoder:

\[ z_e = v \quad H_e = \frac{\partial z_e}{\partial x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \]

- Always express measurements as a prediction based on:
  - The present state
  - No other measurements

- If you are sure you can’t predict the measurements from the state, add more state variables til you can.
5.3.4.5.1 Transmission Encoder Measurement Model (Error Model)

• Express uncertainty as “distance” dependent random walk.

• In continuous time:

  \[ \dot{R}_e = \dot{\sigma}_{ee} = \alpha |v| \]

• Multiply by \( \Delta t_e \) to get:

  \[ R_e = \sigma_{ee} = \alpha |\Delta s| \]

• Produces a position variance that grows **linearly with distance** between measurements.

---

That is, when integrated wrt time, grows linearly wrt distance because \( Vdt = ds \)

Why | | ?
5.3.4.5.2 Gyro Measurement Model/Uncertainty

• Gyro measurement:

\[ z_g = \omega \]

\[ H_g = \frac{\partial z_g}{\partial x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} \]

• For R, go with time dependent random walk:

\[ \dot{R}_g = \dot{\sigma}_{gg} / \Delta t_g \]

• To convert to discrete time (multiply by \( \Delta t_g \)).
• Makes the variance of angle rate constant while variance of computed angle grow linearly with time.
1/2 DONE

😊

now have:

\[ z = h(x) \]

& H

& R
Time for a Few Good Z’s
Dead Reckoning

• So far, we have a lot of code that does this:

• Any process that only integrates noisy velocities must eventually (quickly?) get lost.

• Without pose “fixes”, even an optimal estimate is not much use.
Landmarks

• Suppose:
  – A map of where the landmarks are in the world.
  – A sensor which measures landmark positions relative to itself.

Note: The book presents a “forced formulation” which is better but not consistent with the homework assignment, so these slides cover an unforced formulation – where velocities remain in the state vector.
5.3.4.6.1 Forced Formulation

• Can treat velocity measurements as inputs $u$ rather than measurements $z$.
• Errors in the velocities are then modeled in $Q$ rather than $R$.
• The state vector is smaller:

$$x = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$$

$$x_{k+1} \approx x_k + f(x, u, t) \Delta t$$

• System Model:
5.3.4.6.1 Forced Formulation

• System model in matrix form:

\[ x_{k+1} \approx \Phi x_k + G u_k \quad \Rightarrow \quad \Phi \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad G \approx \begin{bmatrix} c \psi_k \Delta t_k \\ s \psi_k \Delta t_k \\ 0 \end{bmatrix} \]

• System Jacobian:

\[ F = \frac{\partial \hat{x}}{\partial x} = \begin{bmatrix} \partial \hat{x}/\partial x & \partial \hat{x}/\partial y & \partial \hat{x}/\partial \theta \\ \partial \hat{y}/\partial x & \partial \hat{y}/\partial y & \partial \hat{y}/\partial \theta \\ \partial \hat{\theta}/\partial x & \partial \hat{\theta}/\partial y & \partial \hat{\theta}/\partial \theta \end{bmatrix} = \begin{bmatrix} 0 & 0 & -v_s \psi \\ 0 & 0 & v_c \psi \\ 0 & 0 & 0 \end{bmatrix} \]

• $\Phi_k$ matrix:

\[ \Phi_k \approx I + F \Delta t = \begin{bmatrix} 1 & 0 & -v_s \psi \Delta t \\ 0 & 1 & v_c \psi \Delta t \\ 0 & 0 & 1 \end{bmatrix} \]

• State Uncertainty Propagation:

\[ P_{k+1}^- = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T \]
5.3.4.6 Incorporating a Map
(Landmark Measurement Model)

• This is of the form $z = h(x)$ where:

$$x = \begin{bmatrix} x_w^w & y_w^w & \psi \end{bmatrix}$$
5.3.4.6 Incorporating a Map
(Landmark Measurement Model)

- Jacobian w.r.t robot pose:
  \[ H_x^z = \left( \frac{\partial z}{\partial \rho_d^s} \right) \left( \frac{\partial \rho_d^s}{\partial \rho_d^b} \right) \left( \frac{\partial \rho_d^b}{\partial \rho_w^m} \right) = H_{sd}^z H_{bd}^{sd} H_{x}^{bd} \]

- Jacobian w.r.t landmark pose:
  \[ H_{wm}^z = \left( \frac{\partial z}{\partial \rho_d^s} \right) \left( \frac{\partial \rho_d^s}{\partial \rho_d^b} \right) \left( \frac{\partial \rho_d^b}{\partial \rho_w^m} \right) \left( \frac{\partial \rho_w^m}{\partial \rho_w^d} \right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd} \]
5.3.4.6.2 Observer and Jacobian

- A real sensor does not measure in Cartesian coordinates. Polar is more likely:

\[ \cos \alpha = \frac{x_d^s}{r_d^s} \]
\[ \sin \alpha = \frac{y_d^s}{r_d^s} \]

\[ z_{sen} = \begin{bmatrix} \alpha \\ r_d^s \end{bmatrix} = f(r_d^s) = \begin{bmatrix} \tan^{-1}(y_d^s/x_d^s) \\ \sqrt{(x_d^s)^2 + (y_d^s)^2} \end{bmatrix} \]
5.3.4.6.3 Sensor Referenced Observation

Nothing here but tons of math......

Recall:

\[
H_x^z = \left( \frac{\partial z}{\partial p_s} \right) \left( \frac{\partial p_d}{\partial p_s} \right) = H_{sd} H_{bd} H_x^z \quad \text{and} \quad H_w^z = \left( \frac{\partial z}{\partial p_s} \right) \left( \frac{\partial p_d}{\partial p_s} \right) = H_{sd} H_{bd} H_w^z
\]

Occurs in 2 places

\[
z_{sen} = \begin{bmatrix} \alpha \\ r_s^d \end{bmatrix} = f(r_s^d) = \begin{bmatrix} \arctan(y_s^d / x_s^d) \\ \sqrt{(x_s^d)^2 + (y_s^d)^2} \end{bmatrix}
\]

\[
H_{sd}^z = \frac{\partial z}{\partial r_s^d} = \begin{bmatrix} H_s^a \\ H_s^r \end{bmatrix} = \begin{bmatrix} \frac{1}{(r_s^d)^2} \begin{bmatrix} -y_s^d & x_s^d \end{bmatrix} \\ \frac{1}{r_s^d} \begin{bmatrix} x_s^d & y_s^d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \frac{1}{r_s^d} \begin{bmatrix} -s\alpha & c\alpha \end{bmatrix} \\ \begin{bmatrix} c\alpha & s\alpha \end{bmatrix} \end{bmatrix}
\]
5.3.4.6.4 Body To Sensor

\[ H^z_x = \left( \frac{\partial z}{\partial \rho^s_d} \right) \left( \frac{\partial \rho^b_d}{\partial \rho^b_d} \right) \left( \frac{\partial \rho^b_d}{\partial \rho^w_d} \right) = H^z_{sd} H^z_{bd} H^z_x \]

\[ H^z_w = \left( \frac{\partial z}{\partial \rho^s_d} \right) \left( \frac{\partial \rho^b_d}{\partial \rho^w_d} \right) \left( \frac{\partial \rho^w_d}{\partial \rho^w_m} \right) = H^z_{sd} H^z_{bd} H^z_{wd} H^z_w \]

- **We need:** \( \frac{\partial \rho^s_d}{\partial \rho^b_d} \)
- **Inverse is:** \( \frac{\partial \rho^b_d}{\partial \rho^s_d} \)
- **Compound-Right Pose Jacobian!**

\[
\frac{\partial \rho^b_d}{\partial \rho^s_d} = \begin{bmatrix} c\psi^b_s & -s\psi^b_s & 0 \\ s\psi^b_s & c\psi^b_s & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
H^{sd}_{bd} = \left( \frac{\partial \rho^b_d}{\partial \rho^s_d} \right)^{-1} = \frac{\partial \rho^s_d}{\partial \rho^b_d} = \begin{bmatrix} c\psi^b_s & s\psi^b_s & 0 \\ -s\psi^b_s & c\psi^b_s & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]
5.3.4.6.5 World to Body: First Jacobian

\[ H_x^z = \begin{pmatrix} \frac{\partial z^s}{\partial \rho^b_d} \frac{\partial \rho^b_d}{\partial \rho^w_d} \end{pmatrix} = H_{sd}^z H_{bd}^z H_x^b \]

\[ H_w^z = \begin{pmatrix} \frac{\partial z^s}{\partial \rho^b_d} \frac{\partial \rho^b_d}{\partial \rho^w_d} \end{pmatrix} = H_{sd}^z H_{bd}^z H_{wd}^w H_w^m \]

- We need: \( \frac{\partial \rho^b_d}{\partial \rho^w_d} \)
- Inverse is: \( \frac{\partial \rho^w_d}{\partial \rho^b_d} \)
- Compound-Right Pose Jacobian

\[ \frac{\partial \rho^w_d}{\partial \rho^b_d} = \begin{bmatrix} c\psi^w_b & -s\psi^w_b & 0 \\ s\psi^w_b & c\psi^w_b & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ H_{wd}^{bd} = \left( \frac{\partial \rho^w_d}{\partial \rho^b_d} \right)^{-1} = \frac{\partial \rho^b_d}{\partial \rho^w_d} = \begin{bmatrix} c\psi^w_b & s\psi^w_b & 0 \\ -s\psi^w_b & c\psi^w_b & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
5.3.4.6.5 World to Body: Second Jacobian

\[
H^z_x = \left( \frac{\partial z}{\partial \rho^s_d} \right) \left( \frac{\partial \rho^s_d}{\partial \rho^b_d} \right) \left( \frac{\partial \rho^b_d}{\partial \rho^w} \right) = H^z_{sd} H^s_{bd} H^b_x \quad H^z_w = \left( \frac{\partial z}{\partial \rho^s_d} \right) \left( \frac{\partial \rho^s_d}{\partial \rho^w} \right) = H^z_{sd} H^s_{bd} H^w_d H^w_{wm}
\]

- We need: \( \frac{\partial \rho^b_d}{\partial \rho^w} \)
- Right-Left Pose Jacobian

\[
H^{bd} = \frac{\partial \rho^b_d}{\partial \rho^w} = \begin{bmatrix} c \psi_b & s \psi_b & -y^b_d \\ -s \psi_b & c \psi_b & x^b_d \\ 0 & 0 & 1 \end{bmatrix}
\]

This means there is info here on x and y and \( \theta \).
5.3.4.6.6 Model to World

\[
H^z_x = \left( \frac{\partial z}{\partial \rho^s_d} \right) \left( \frac{\partial \rho^b_d}{\partial \rho^b_d} \right) = H^z_x H^{sd^d}_b H^{bd}_x
\]

\[
H^z_{wm} = \left( \frac{\partial z}{\partial \rho^s_d} \right) \left( \frac{\partial \rho^b_d}{\partial \rho^b_d} \right) \left( \frac{\partial \rho^w_d}{\partial \rho^w_m} \right) = H^z_x H^{sd^d}_b H^{bd}_w H^{wd}_{wm}
\]

- We need: \( \frac{\partial \rho^w_d}{\partial \rho^w_m} \)
- Compound-Left Pose Jacobian

\[
H^{wd}_{wm} = \frac{\partial \rho^w_d}{\partial \rho^w_m} = \begin{bmatrix}
1 & 0 & -(y^w_d - y^w_m) \\
0 & 1 & (x^w_d - x^w_m) \\
0 & 0 & 1
\end{bmatrix}
\]
Total Measurement Model: Point Features

• Compute it like this:

\[ r_d^s = T_b^s T_w^b (\rho_b^w) T_m^w (r_m^w) r_d^m \]

\[ z_{sen} = \begin{bmatrix} \alpha \\ r_d^s \end{bmatrix} = f(r_d^s) = \begin{bmatrix} \text{atan}(y_d^s/x_d^s) \\ \sqrt{(x_d^s)^2 + (y_d^s)^2} \end{bmatrix} \]

• Jacobians

\[ H^z_x = \begin{pmatrix} \frac{\partial z}{\partial \rho_d^s} & \frac{\partial z}{\partial \rho_b^b} & \frac{\partial z}{\partial \rho_w^w} \end{pmatrix} = H^z_{sd} H_{bd} H_{x}^{bd} \]

\[ H^z_{wm} = \begin{pmatrix} \frac{\partial z}{\partial \rho_d^s} & \frac{\partial z}{\partial \rho_b^b} & \frac{\partial z}{\partial \rho_w^w} & \frac{\partial z}{\partial \rho_m^m} \end{pmatrix} = H^z_{sd} H_{bd} H_{wd} H_{wm} \]
3/4 DONE!

.now have some really good z’s
Still
Not
Done! 😞
Data Association

• The Achilles Heel of the Kalman Filter.
• There are lots of landmarks out there. How do you know which ones you are looking at?
• One mistake and its all over:
  – A potentially massive change in the vehicle pose will occur.
  – This will cause more wrong associations and fewer or no right ones.
  – The filter will diverge, and the system will rapidly get lost.
Innovation Covariance

• This is the expression:

\[ S = HPH^T + R \]

• in the Kalman Gain calculation.

• Represents the covariance of the innovation $z-h(x)$.
  – I.E. how does the state error $P$ [in $h(x)$] and the measurement error $R$ [in $z$] combine to give the error in my prediction right now.
Validation Gates

- Recall the Mahalanobis distance - multidimensional deviation from the mean:
  \[ d = \sqrt{\Delta z^T S^{-1} \Delta z} \]

- Compute this for every landmark giving \( n \) d’s to look at.
- It turns out if the innovation is Gaussian, then the MHD is Chi square distributed. Confidence thresholds can be derived:

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>95% confidence gate</th>
<th>99% confidence gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.02</td>
<td>7.87</td>
</tr>
<tr>
<td>2</td>
<td>7.38</td>
<td>10.60</td>
</tr>
<tr>
<td>3</td>
<td>9.35</td>
<td>12.38</td>
</tr>
<tr>
<td>4</td>
<td>11.14</td>
<td>14.86</td>
</tr>
</tbody>
</table>
Validation Gates

• This leads to some good ideas for data association:
  – Require that any candidate association have a MD < “about 3”.
  – Require that there be no other candidate association with an MD < 6 or an even bigger number.
  – Require that an association be stable for several cycles before it is actually used.
Done!

This has been a ... really really useful ... Kalman Filter
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot – Harder Example
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
Just Kidding!

• Here are some graphs of a 3D filter.
3D AHRS Filter Results
3D AHRS Filter Results

Vertical Position

z coordinate in meters (/10)

cycle number (/ 100)

Zups
3D AHRS Filter Results
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
Sequential Measurement Processing

• All measurements do not have to come in at the same rate.
• Just process ‘em when you have ‘em after predicting state for their time of arrival.

```c
State_Update() /* enter every cycle */
{
    systemModel(dt);

    if( Doppler measurement available)
        run Kalman() on Doppler;
    if( Encoder measurement available)
        run Kalman() on encoder;
    if( AHRS measurement available)
        run Kalman() on AHRS;
    if( Steering measurement available)
        run Kalman() on steering;
}
Kalman()
{
}
```
Single Measurement Efficiency: Kalman Gain

• Recall:

\[
K_k = P_k^{-1} H_k^T (H_k P_k^{-1} H_k^T + R_k)^{-1}
\]

• Suppose only one direct measurement: \( R = \begin{bmatrix} r \end{bmatrix} \)

• Measurement Jacobian is:

\[
H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}
\]

• Define: \( p = P_{ss} \)

• Then, Kalman Gain is a scalar times s’th column of P:

\[
K = \left( \frac{1}{p + r} \right) P_{is}
\]
Uncertainty Propagation

• The formula: $P = [I - (KH)]P$

• takes $n^2(1+m) + n^3$ flops
  – $[1200]$ for $n=10, m=1$

• Can be computed more efficiently as:
  $P = P - K(HP)$

• which takes $n^2(1+m) + mn^2$ flops
  – $[300]$ for $n=10, m=1$
Uncertainty Propagation

• For a scalar measurement, recall:

\[ H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \]

• KHP is just a constant times the outer product:

\[(KHP)_{ij} = \left( \frac{1}{p + r} \right) P_{is} P_{sj} \quad \forall i \forall j\]
R matrix and cycle time

• It is slightly better to have every element of $R$ be proportional to $dt$. This tends to make your filter behave appropriately if you change the time step.

• If not, you can get weird behaviors like a filter which produces worse answers if you run it faster (because you are adding up more random numbers of the same variance).
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter - SKIP
  – Summary
Outline

• 5.3 State Space Kalman Filters
  – 5.3.1 Introduction
  – 5.3.2 Linear Discrete Time Kalman Filter
  – 5.3.3 Kalman Filters for Nonlinear Systems
  – 5.3.4 Simple Example: 2D Mobile Robot
  – 5.3.5 Pragmatic Information for Kalman Filters
  – 5.3.6 Other Forms of the Kalman Filter
  – Summary
Summary

• A SS KF is conceptually two sets of equations.
• Most cases require linearization. The “extended” form is the most useful.
• Handles the tricky issue of integration dead reckoning and position fixes automatically.
• Most measurements are scalar and we often assume decorrelation. Leads to processing efficiencies.