Chapter 5
Optimal Estimation

Part 3

5.3 State Space Kalman Filters

Carnegie Mellon
THE ROBOTICS INSTITUTE

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

Carnegie Mellon
THE ROBOTICS INSTITUTE

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)

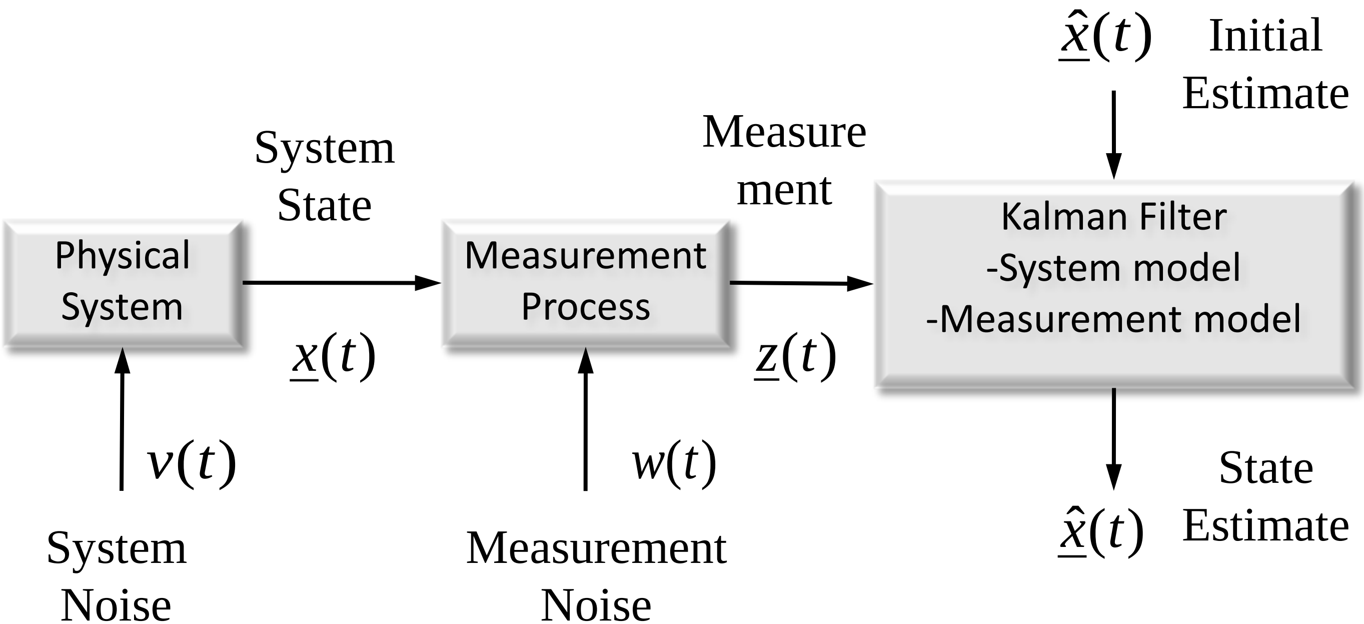
5.3.1 Introduction

- Recall the form of state space model of a system:

$$\dot{\underline{x}} = F\underline{x} + G\underline{w}$$

$$\underline{z} = H\underline{x} + \underline{v}$$

5.3.1 Overall Operation



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:

$$x_1 = x(t_1) \qquad\qquad z_2 = z(t_2)$$

- Not all of the difference between x1 and x2 is now due to error. Some of it is motion.
- Must compute x2 from x1 and then compare z2 to that.
- That also involves predicting the error in the prediction → recall how error compounds in dead reckoning.

5.3.1.3 Discrete Time System Model

- Continuous Time:

$$\dot{\underline{x}} = F\underline{x} + G\underline{w}$$ State or Process Model

$$\underline{z} = H\underline{x} + \underline{v}$$ Measurement or Observation Model

Continuous form rarely used in practice

- Discrete Time:

$$\hat{\underline{x}}_{k+1} = \Phi_k \hat{\underline{x}}_k + G_k \underline{w}_k$$ State or Process Model

$$\underline{z}_k = H_k \underline{x}_k + \underline{v}_k$$ Measurement or Observation Model

Nomenclature

Object	Size	Name	Comment
$\hat{\underline{x}}_k$	$n \times 1$	state vector estimate at time t_k	
Φ_k	$n \times n$	transition matrix	relates $\underline{x}_k$ to $\underline{x}_{k+1}$ in the absence of a forcing function
G_k	$n \times n$	process noise distribution matrix	transforms the $\underline{w}_k$ vector into the coordinates of $\underline{x}_k$
$\underline{w}_k$	$n \times 1$	disturbance sequence or process noise sequence	white, known covariance structure
$\underline{z}_k$	$m \times 1$	measurement at time t_k	
H_k	$m \times n$	measurement matrix or observation matrix	relates $\underline{x}_k$ to $\underline{z}_k$ in the absence of measurement noise
$\underline{v}_k$	$m \times 1$	measurement noise sequence	white, known covariance structure

n = # states **m = # measurements**

5.3.1.3 Noises

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

$$E(\underline{w}_k \underline{w}_i^T) = \delta_{ik} Q_k$$

$$E(\underline{v}_k \underline{v}_i^T) = \delta_{ik} R_k$$

$$E(\underline{w}_k \underline{v}_i^T) = 0, \forall (i, k)$$

5.3.1.4 Transition Matrix

- Converts continuous time ODEs to discrete time ones:
- The time continuous, matrix ODE:

$$\dot{\underline{x}} = F(t)\underline{x}$$

- Can always be converted to:

$$\underline{x}_{k+1} = \Phi_k \underline{x}_k$$

- 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 Δt 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
  - <u>5.3.2 Linear Discrete Time Kalman Filter</u>
  - 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

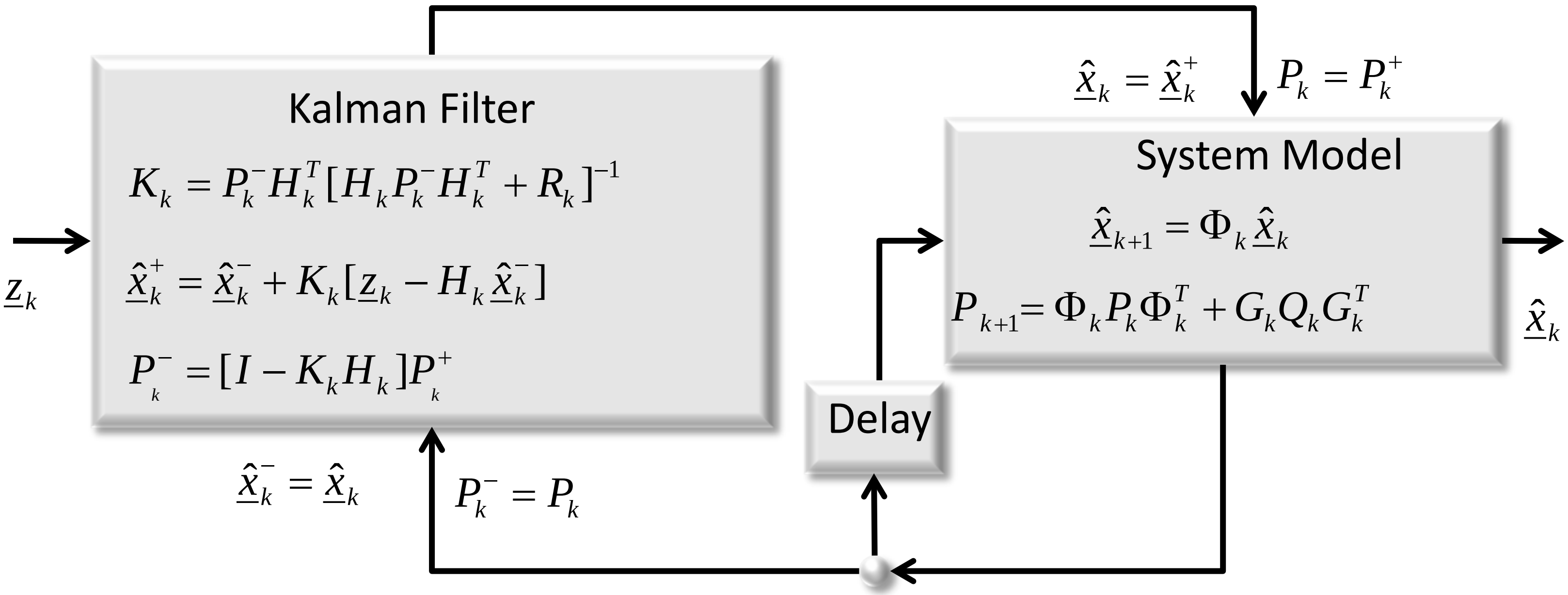
5.3.2.1 The Filter Equations – 2 Sets

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

System Model	$\hat{\underline{x}}_{k+1} = \Phi_k \hat{\underline{x}}_k$	*predict state*
	$P_{k+1} = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T$	*predict covariance*
Kalman Filter	$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$	*compute Kalman gain*
	$\hat{\underline{x}}_k^+ = \hat{\underline{x}}_k^- + K_k[\underline{z}_k - H_k \hat{\underline{x}}_k^-]$	*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_k:
  - you provide this
  - instantaneous uncertainty which corrupts the system model
  - random physical disturbances and process model errors
- R_k:
  - you provide this too
  - instantaneous uncertainty which corrupts the measurement model
  - random errors in sensor outputs
- P_k :
  - 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:

$$\dot{\underline{x}} = f(\underline{x}, t) + g(t)\underline{w}(t)$$

$$\underline{z} = h(\underline{x}, t) + \underline{v}(t)$$

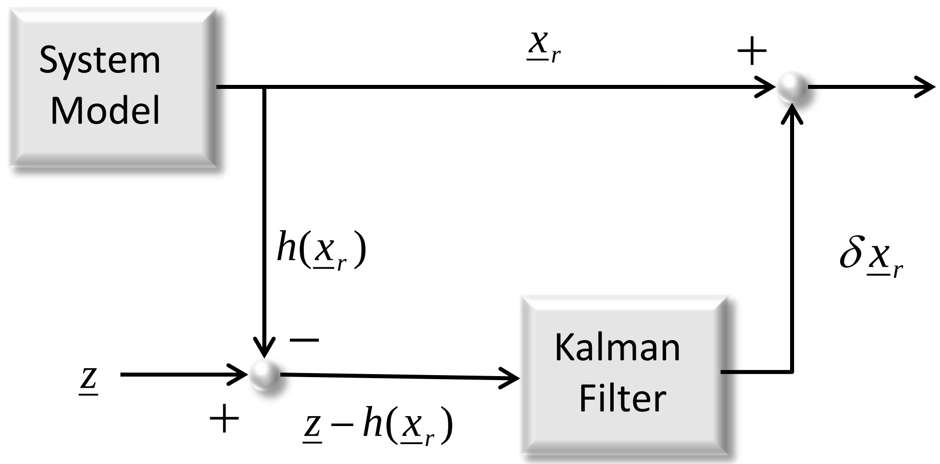
- Linearize about a reference trajectory x*(t)

$$\Delta\dot{\underline{x}} = \frac{\partial f}{\partial \underline{x}}(\underline{x}^*, t)\Delta\underline{x} + g(t)\underline{w}(t)$$

$$\underline{z} - h(\underline{x}^*, t) = \frac{\partial h}{\partial \underline{x}}(\underline{x}^*, t)\Delta\underline{x} + \underline{v}(t)$$

Linear (Feedforward) Kalman Filter

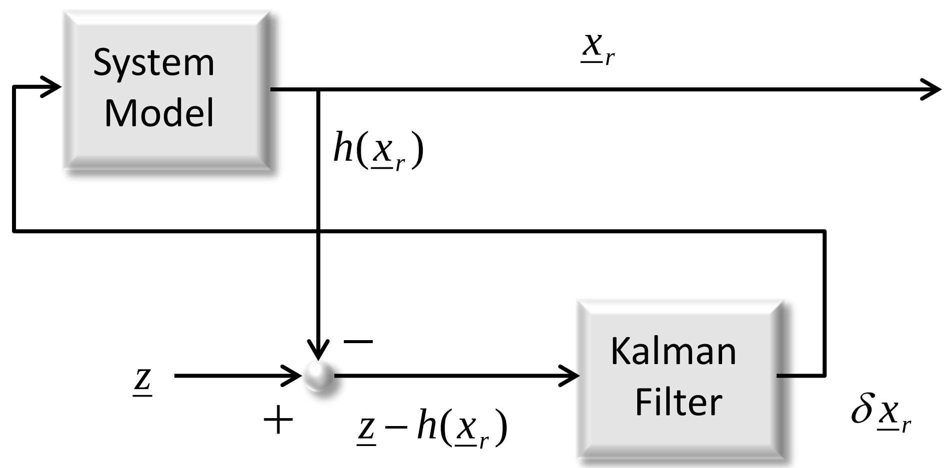
- Does not update the reference trajectory:



- 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:

$$F_k = \frac{\partial f}{\partial \underline{x}}(\hat{\underline{x}}_k^-) \quad G_k = \frac{\partial g}{\partial \underline{w}}(\hat{\underline{x}}_k^-) \quad H_k = \frac{\partial \underline{h}}{\partial \underline{x}}(\hat{\underline{x}}_k^-)$$

  - Compute Kalman gain:
  - Update state estimate:
  - Update its covariance:

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$$

$$\hat{\underline{x}}_k^+ = \hat{\underline{x}}_k^- + K_k[\underline{z}_k - h(\hat{\underline{x}}_k^-)]$$

$$P_k^+ = [I - K_k H_k] P_k^-$$

- System Model:
  - Project state:
  - Project covariance:

$$\hat{\underline{x}}_{k+1} = \phi_k(\hat{\underline{x}}_k)$$

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

These are the ones you will use for almost any filter.

State Transition – Nonlinear Problems

- When the system model is nonlinear:

$$\underline{\dot{x}} = f(\underline{x}(t), t)$$

- 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:

$$\underline{x}_{k+1} = \underline{x}_k + f(\underline{x}_k, t_k)\Delta t$$

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.

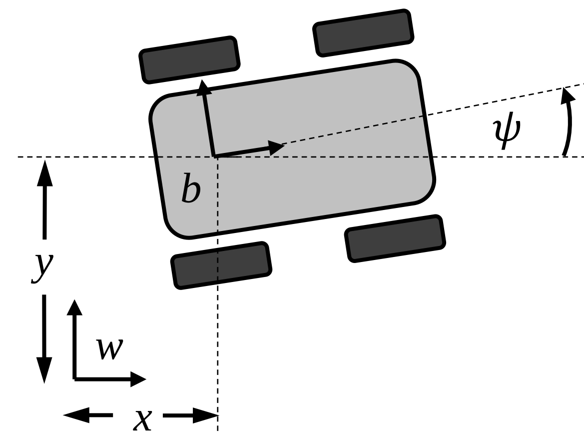
Outline

- 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
  - <u>5.3.4 Simple Example: 2D Mobile Robot</u>
  - 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:

$$\underline{x} = \left[\underbrace{x \;\; y \;\; \psi}_{\text{Care about this}} \;\; \underbrace{v \;\; \omega}_{\text{Need this to propagate state}}\right]^T$$



- Measurements

$$\underline{z} = \left[z_e \;\; z_g\right]^T$$

Transmission Encoder

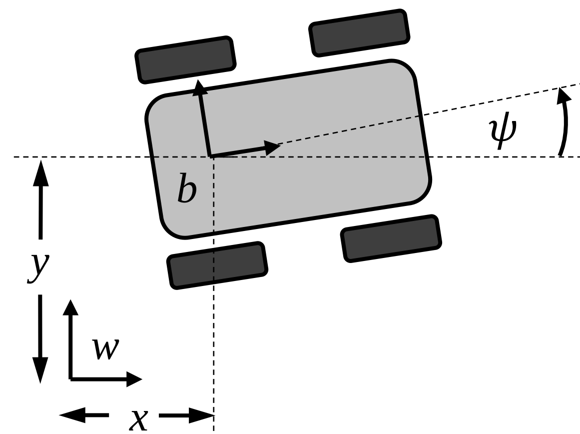
Gyro

5.3.4.1 System and Measurement Model (System Model)

- Generally of the form:

$$\dot{\underline{x}} = \frac{d\underline{x}}{dt} = f(\underline{x}, t)$$

- Here, it is:



$$\dot{\underline{x}} = \frac{d\underline{x}}{dt} = f(\underline{x}, t) \Rightarrow \frac{d}{dt}\begin{bmatrix} x & y & \psi & v & \omega \end{bmatrix}^T$$

$$\dot{\underline{x}} = \begin{bmatrix} vc\psi & vs\psi & \omega & 0 & 0 \end{bmatrix}^T$$

Nonlinear!

- 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:

$$\dot{\underline{x}} = \begin{bmatrix} vc\psi & vs\psi & \omega & 0 & 0 \end{bmatrix}^T$$

- "Clearly":

ψ **F**

$$F = \begin{bmatrix} \partial\dot{x}/\partial x & \partial\dot{x}/\partial y & \partial\dot{x}/\partial\theta & \partial\dot{x}/\partial v & \partial\dot{x}/\partial\omega \\ \partial\dot{y}/\partial x & \partial\dot{y}/\partial y & \partial\dot{y}/\partial\theta & \partial\dot{y}/\partial v & \partial\dot{y}/\partial\omega \\ \partial\dot{\theta}/\partial x & \partial\dot{\theta}/\partial y & \partial\dot{\theta}/\partial\theta & \partial\dot{\theta}/\partial v & \partial\dot{\theta}/\partial\omega \\ \partial\dot{v}/\partial x & \partial\dot{v}/\partial y & \partial\dot{v}/\partial\theta & \partial\dot{v}/\partial v & \partial\dot{v}/\partial\omega \\ \partial\dot{\omega}/\partial x & \partial\dot{\omega}/\partial y & \partial\dot{\omega}/\partial\theta & \partial\dot{\omega}/\partial v & \partial\dot{\omega}/\partial\omega \end{bmatrix} = \boxed{\begin{bmatrix} 0 & 0 & -vs\psi & c\psi & 0 \\ 0 & 0 & vc\psi & s\psi & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}$$

5.3.4.2 Discretize and Linearize

- Linearize:

$$\underline{x}_{k+1} \approx \underline{x}_k + f(x, t)\Delta t$$

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

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

- Express in matrix form:

$$\underline{x}_{k+1} \approx \hat{\Phi}\underline{x}_k$$

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

THIS IS NOT Φ!

- 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:

$$\Phi_k \approx I + F\Delta t$$

- Where:

$$\Phi_k \approx \begin{bmatrix} 1 & 0 & -vs\psi\Delta t & c\psi\Delta t & 0 \\ 0 & 1 & vc\psi\Delta t & s\psi\Delta t & 0 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Note difference from F matrix 2 slides ago!

5.3.4.3 Initialization

- Be careful with P0:
  - Too little P0 and measurements will be ignored.
  - Too much P0 and numerical problems.
- Here assume:

$$P_0 = diag\left[\sigma_{xx}\ \sigma_{yy}\ \sigma_{\psi\psi}\ \sigma_{vv}\ \sigma_{\omega\omega}\right]$$

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 Δt.
- 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: $$Q_k = diag[\underbrace{k_{xx}, k_{yy}, k_{\psi\psi}, k_{vv} k_{\omega\omega}}_{\text{constants}}]\Delta t$$
- 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 …

$$\underline{\hat{x}}_{k+1} = \phi_k(\underline{\hat{x}}_k)$$

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

5.3.4.5.1 Transmission Encoder Measurement Model

- “Velocity” encoder:

$$z_e = v \qquad\qquad H_e = \frac{\partial z_e}{\partial \underline{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- 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:

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

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

Why | | ?

- Multiply by Δt_e to get:

$$R_e = \sigma_{ee} = \alpha|\Delta s|$$

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

5.3.4.5.2 Gyro Measurement Model/Uncertainty

- Gyro measurement:

$$z_g = \omega \qquad\qquad H_g = \frac{\partial z_g}{\partial \underline{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- For R, go with time dependent random walk:

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

- To convert to discrete time (multiply by Δt_g).
- Makes the variance of angle rate constant while variance of computed angle grow linearly with time.

1/2 DONE

☹

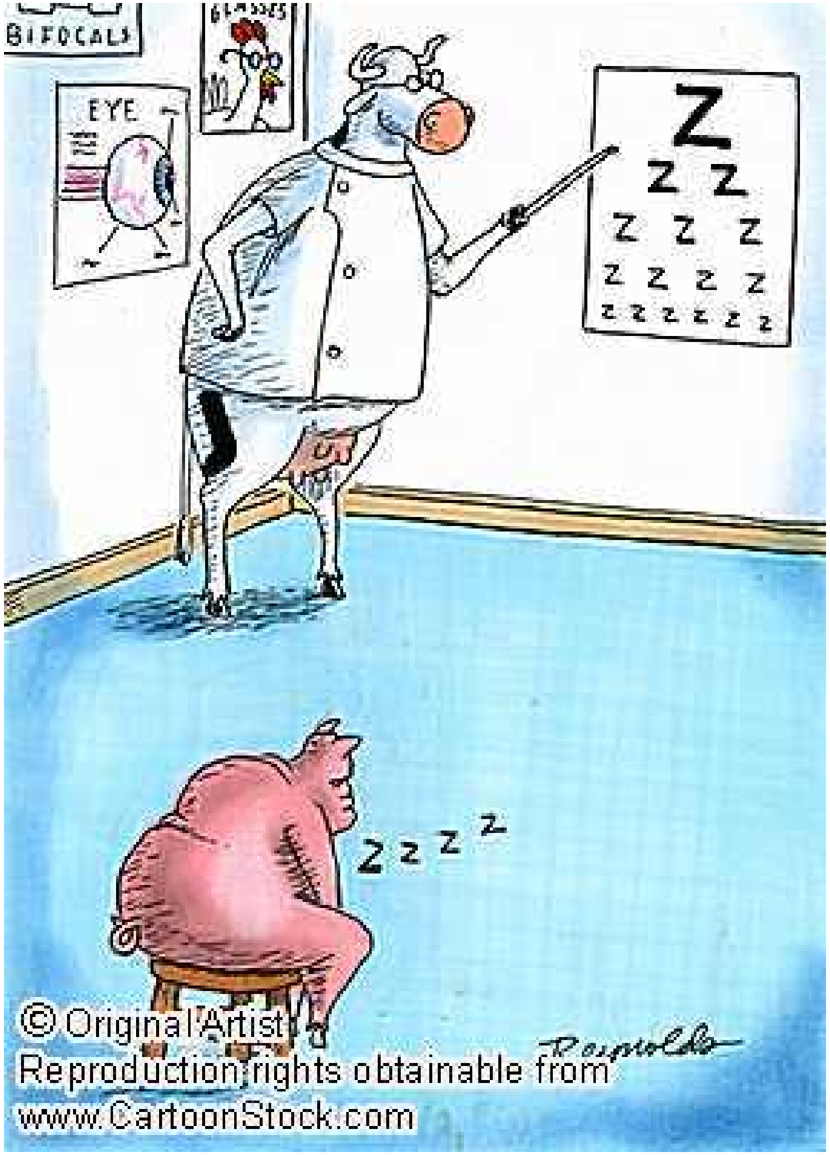
now have:

$z = h(x)$

& H

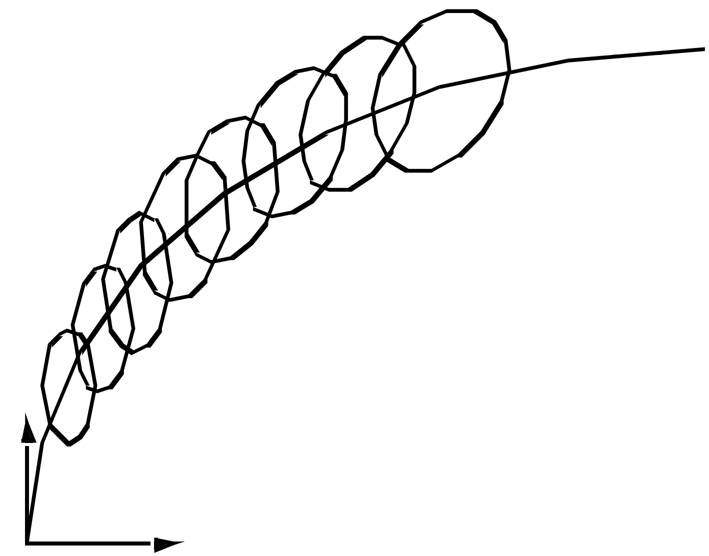
& R

Time for a Few Good Z's



Dead Reckoning

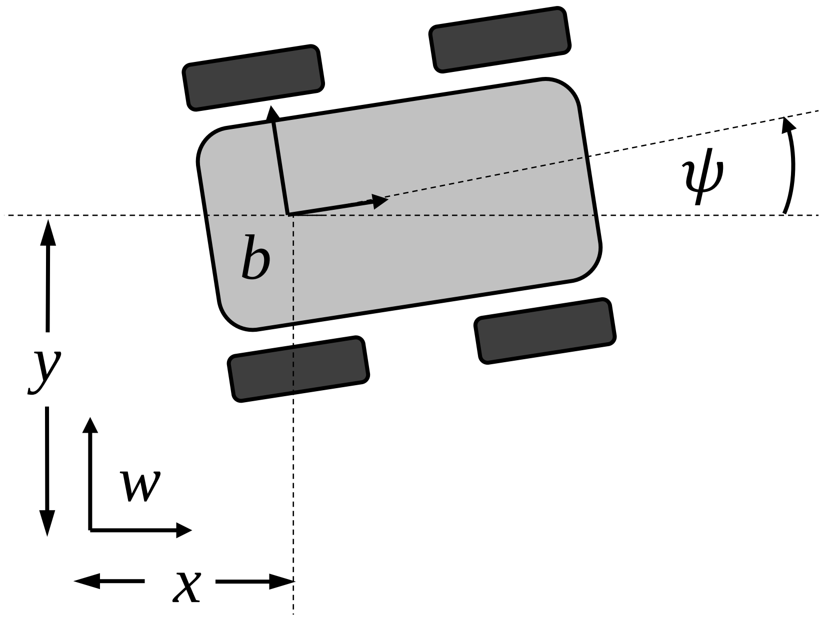
- So far, we have a lot of code that does this:
- Any process that only integrates noisy velocities must eventually (quickly?) get lost.
- Without pose "fixes", even an optimal estimate is not much use.



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 $\underline{u}$ rather than measurements $\underline{z}$.
- Errors in the velocities are then modeled in Q rather than R.
- The state vector is smaller: $\underline{x} = \begin{bmatrix} x & y & \psi \end{bmatrix}$

$$\underline{x}_{k+1} \approx \underline{x}_k + f(x, u, t)\Delta t$$

- System Model:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \end{bmatrix} \approx \begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix} + \begin{bmatrix} v_k c\psi_k \\ v_k s\psi_k \\ \omega_k \end{bmatrix} \Delta t_k$$

5.3.4.6.1 Forced Formulation

- System model in matrix form:

$$\underline{x}_{k+1} \approx \hat{\Phi}\underline{x}_k + Gu_k \qquad \Rightarrow \qquad \hat{\Phi} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad G \approx \begin{bmatrix} c\psi_k \Delta t_k \\ s\psi_k \Delta t_k \\ 0 \end{bmatrix}$$

- System Jacobian:

$$F = \frac{\partial \dot{\underline{x}}}{\partial \underline{x}} = \begin{bmatrix} \partial\dot{x}/\partial x & \partial\dot{x}/\partial y & \partial\dot{x}/\partial\theta \\ \partial\dot{y}/\partial x & \partial\dot{y}/\partial y & \partial\dot{y}/\partial\theta \\ \partial\dot{\theta}/\partial x & \partial\dot{\theta}/\partial y & \partial\dot{\theta}/\partial\theta \end{bmatrix} = \begin{bmatrix} 0 & 0 & -vs\psi \\ 0 & 0 & vc\psi \\ 0 & 0 & 0 \end{bmatrix}$$

- Φ_k matrix:

$$\Phi_k \approx I + F\Delta t = \begin{bmatrix} 1 & 0 & -vs\psi\Delta t \\ 0 & 1 & vc\psi\Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

- State Uncertainty Propagation:

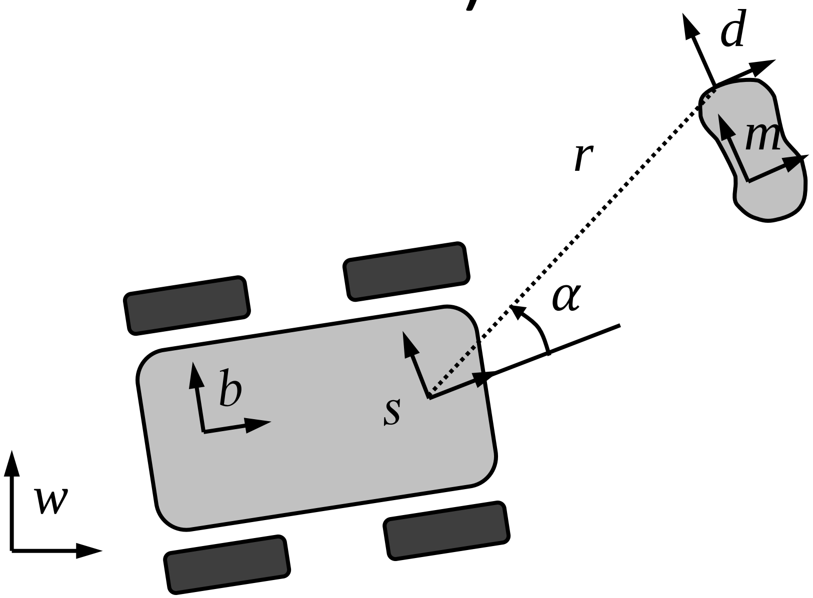
$$P^-_{k+1} = \Phi_k P_k \Phi_k^T + G_k Q_k G_k^T$$

5.3.4.6 Incorporating a Map

(Landmark Measurement Model)

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

$$\underline{x} = \begin{bmatrix} x_b^w & y_b^w & \psi \end{bmatrix}$$



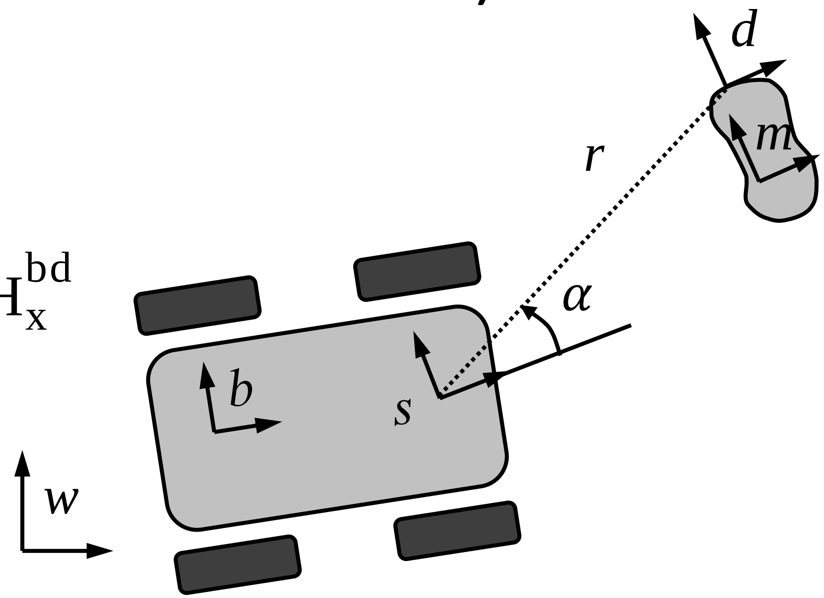
$$\underline{\rho}_d^s = \underline{\rho}_b^s * \underline{\rho}_w^b * \underline{\rho}_m^w * \underline{\rho}_d^m$$

$\underline{\mathbf{x}}$

5.3.4.6 Incorporating a Map
(Landmark Measurement Model)

- Jacobian w.r.t robot pose:

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd}$$



- Jacobian w.r.t landmark pose:

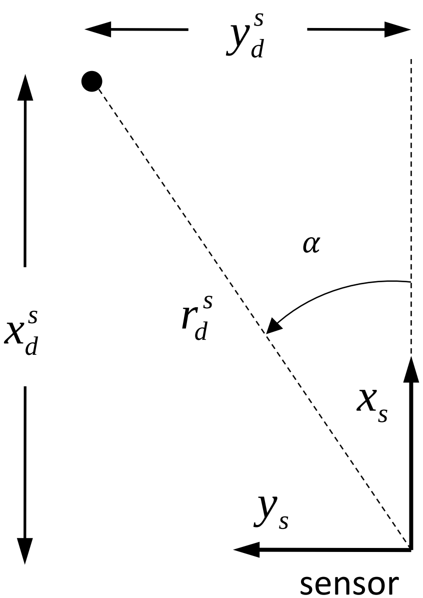
$$\underline{\rho}_d^s = \underline{\rho}_b^s * \underline{\rho}_w^b * \underline{\rho}_m^w * \underline{\rho}_d^m$$

$\underline{\mathbf{x}}_\mathbf{m}$

$$H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

5.3.4.6.2 Observer and Jacobian

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



$$\cos\alpha = x_d^s / r_d^s$$
$$\sin\alpha = y_d^s / r_d^s$$

Forward Kinematics

$$\underline{z}_{sen} = \begin{bmatrix} \alpha \\ r_d^s \end{bmatrix} = f(\underline{r}_d^s) = \begin{bmatrix} \operatorname{atan}(y_d^s / x_d^s) \\ \sqrt{(x_d^s)^2 + (y_d^s)^2} \end{bmatrix}$$

Inverse Kinematics

5.3.4.6.3 Sensor Referenced Observation

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd} \qquad H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

Occurs in 2 places

- Nothing here but tons of math……
- Recall:

$$\underline{z}_{sen} = \begin{bmatrix} \alpha \\ r_d^s \end{bmatrix} = f(\underline{r}_d^s) = \begin{bmatrix} \operatorname{atan}(y_d^s / x_d^s) \\ \sqrt{(x_d^s)^2 + (y_d^s)^2} \end{bmatrix}$$

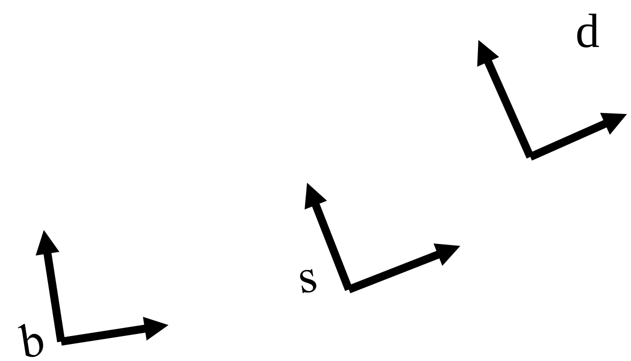
$$H_{sd}^z = \frac{\partial \underline{z}}{\partial \underline{r}_d^s} = \begin{bmatrix} H_s^\alpha \\ H_s^r \end{bmatrix} = \begin{bmatrix} \frac{1}{(r_d^s)^2}\begin{bmatrix} -y_d^s & x_d^s \end{bmatrix} \\ \frac{1}{r_d^s}\begin{bmatrix} x_d^s & y_d^s \end{bmatrix} \end{bmatrix} = \boxed{\begin{bmatrix} \frac{1}{r_d^s}\begin{bmatrix} -s\alpha & c\alpha \end{bmatrix} \\ \begin{bmatrix} c\alpha & s\alpha \end{bmatrix} \end{bmatrix}}$$

5.3.4.6.4 Body To Sensor

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd} \qquad H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

Occurs in 2 places

- We need: $\partial \underline{\rho}_d^s / \partial \underline{\rho}_d^b$
- Inverse is: $\partial \underline{\rho}_d^b / \partial \underline{\rho}_d^s$
- Compound-Right Pose Jacobian!



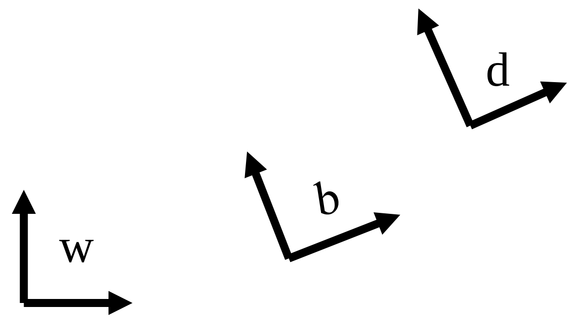
$$\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^s} = \begin{bmatrix} c\psi_s^b & -s\psi_s^b & 0 \\ s\psi_s^b & c\psi_s^b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{H_{bd}^{sd} = \left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^s}\right)^{-1} = \frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b} = \begin{bmatrix} c\psi_s^b & s\psi_s^b & 0 \\ -s\psi_s^b & c\psi_s^b & 0 \\ 0 & 0 & 1 \end{bmatrix}}$$

5.3.4.6.5 World to Body: First Jacobian

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd} \qquad H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

- We need: $\partial \underline{\rho}_d^b / \partial \underline{\rho}_d^w$
- Inverse is: $\partial \underline{\rho}_d^w / \partial \underline{\rho}_d^b$



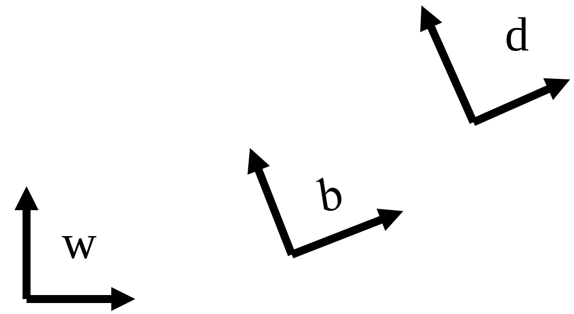
- Compound-Right Pose Jacobian

$$\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_d^b} = \begin{bmatrix} c\psi_b^w & -s\psi_b^w & 0 \\ s\psi_b^w & c\psi_b^w & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boxed{H_{wd}^{bd} = \left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_d^b}\right)^{-1} = \frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w} = \begin{bmatrix} c\psi_b^w & s\psi_b^w & 0 \\ -s\psi_b^w & c\psi_b^w & 0 \\ 0 & 0 & 1 \end{bmatrix}}$$

5.3.4.6.5 World to Body: Second Jacobian

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd} \qquad H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

- We need: $\partial \underline{\rho}_d^b / \partial \underline{\rho}_b^w$
- Right-Left Pose Jacobian



Minus Sign

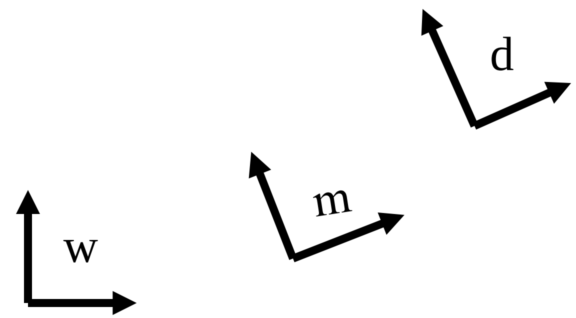
$$H_x^{bd} = \frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w} = -\begin{bmatrix} c\psi_b^w & s\psi_b^w & -y_d^b \\ -s\psi_b^w & c\psi_b^w & x_d^b \\ 0 & 0 & 1 \end{bmatrix}$$

This means there is info here on x and y and θ.

5.3.4.6.6 Model to World

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd} \qquad H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

- We need: $\partial \underline{\rho}_d^w / \partial \underline{\rho}_m^w$
- Compound-Left Pose Jacobian



$$H_{wm}^{wd} = \frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w} = \begin{bmatrix} 1 & 0 & -(y_d^w - y_m^w) \\ 0 & 1 & (x_d^w - x_m^w) \\ 0 & 0 & 1 \end{bmatrix}$$

Total Measurement Model: Point Features

- Compute it like this:

State Variables

Landmark Positions From Map

$$\underline{r}_d^s = T_b^s * T_w^b(\underline{\rho}_b^w) * T_m^w(\underline{r}_m^w) * \underline{r}_d^m$$

Predicted Sensor Reading

$$\underline{z}_{sen} = \begin{bmatrix} \alpha \\ r_d^s \end{bmatrix} = f(\underline{r}_d^s) = \begin{bmatrix} \operatorname{atan}(y_d^s / x_d^s) \\ \sqrt{(x_d^s)^2 + (y_d^s)^2} \end{bmatrix}$$

- Jacobians

Find Robot

$$H_x^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_b^w}\right) = H_{sd}^z H_{bd}^{sd} H_x^{bd}$$

Find Landmark

$$H_{wm}^z = \left(\frac{\partial \underline{z}}{\partial \underline{\rho}_d^s}\right)\left(\frac{\partial \underline{\rho}_d^s}{\partial \underline{\rho}_d^b}\right)\left(\frac{\partial \underline{\rho}_d^b}{\partial \underline{\rho}_d^w}\right)\left(\frac{\partial \underline{\rho}_d^w}{\partial \underline{\rho}_m^w}\right) = H_{sd}^z H_{bd}^{sd} H_{wd}^{bd} H_{wm}^{wd}$$

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 2. Chi Square Validation Gates

Degrees of Freedom	95% confidence gate	99% confidence gate
1	5.02	7.87
2	7.38	10.60
3	9.35	12.38
4	11.14	14.86

Variance Gates

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

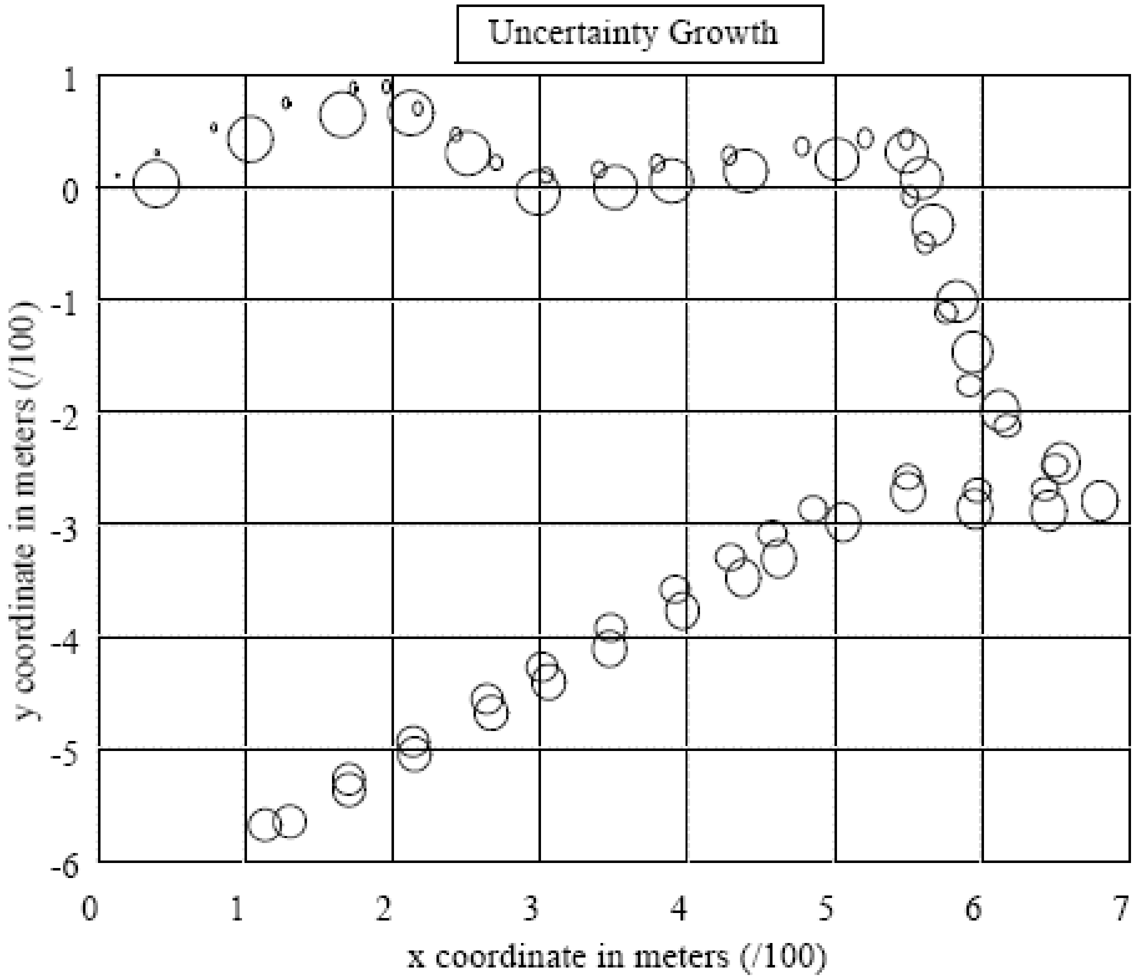
Outline

- 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

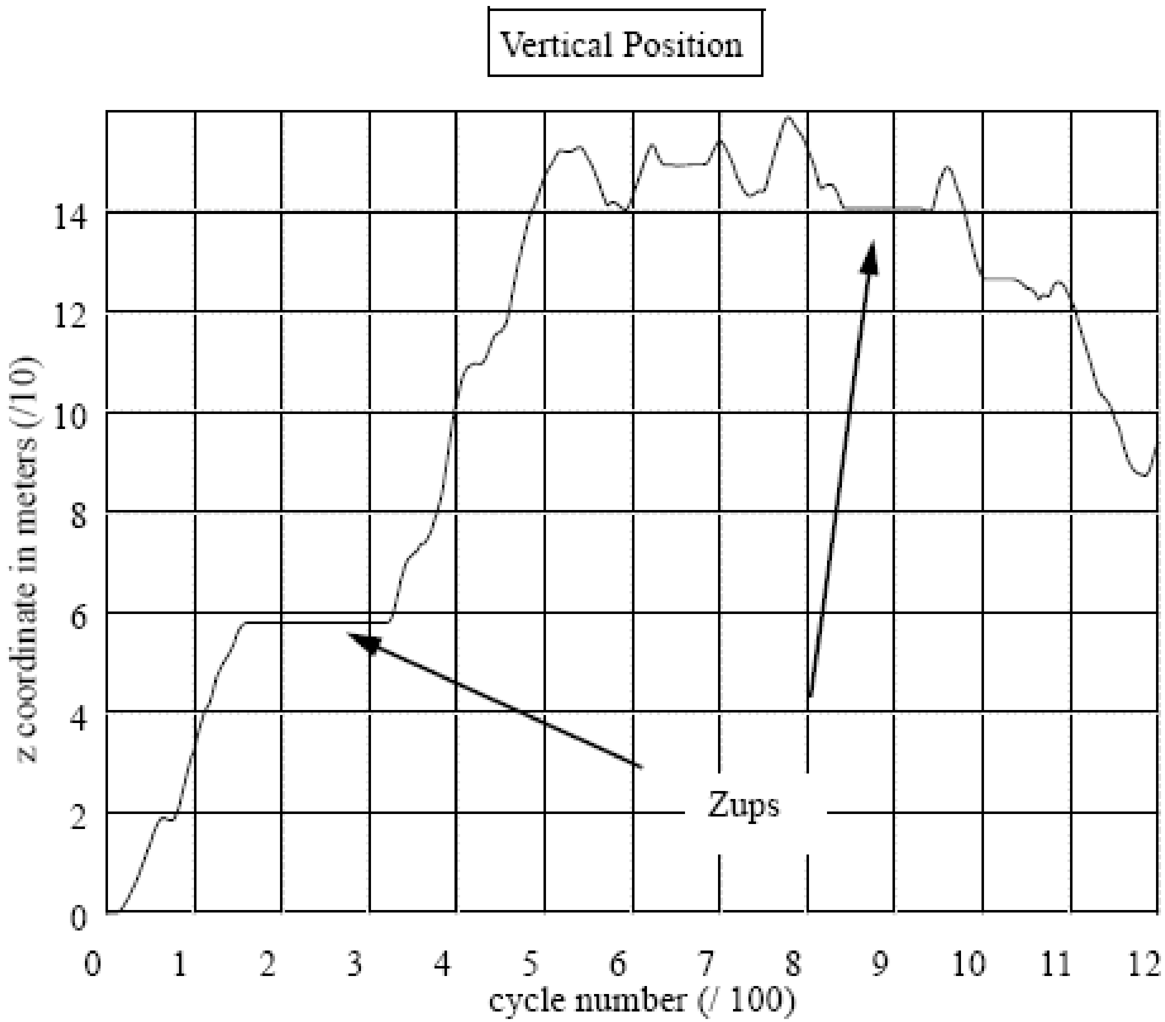
Just Kidding!

- Here are some graphs of a 3D filter.

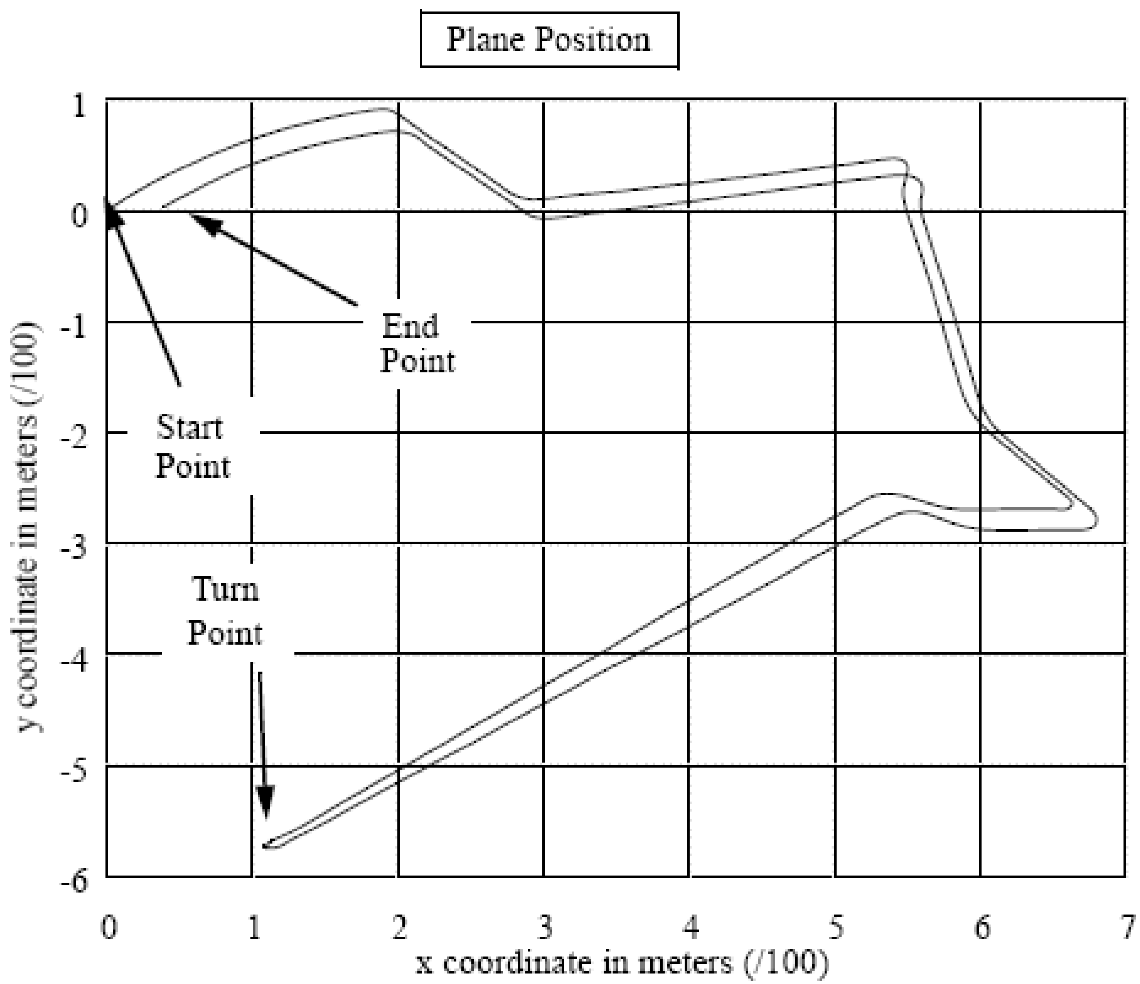
3D AHRS Filter Results



3D AHRS Filter Results



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
  - <u>5.3.5 Pragmatic Information for Kalman Filters</u>
  - 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.

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

if( Doppler measurement available)
   run Kalman() on Doppler;
if(  Encoder  measurement  avail

   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:

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$$

- Suppose only one direct measurement: $R = [r]$

1 at index s

- 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:

$$\boxed{K = \left(\frac{1}{p + r}\right) P_{is}}$$

Sth column of P

Uncertainty Propagation

- The formula: $P = [I - (KH)]P$
- takes n2(1+m)+n3 flops
  - [1200 for n=10,m=1]
- Can be computed more efficiently as:

$$P = P - K(HP)$$

- which takes n2(1+m) + mn2 flops
  - [300 for n=10,m=1]

Uncertainty Propagation

- For a scalar measurement, recall:

1 at index s

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- KHP is just a constant times the outer product:

$$(KHP)_{ij} = \left(\frac{1}{p + r}\right) P_{is} P_{sj} \qquad \forall i \forall j$$

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 wierd 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
  - <u>Summary</u>

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.