Multi-Sensor Data Fusion

Class 2: Algorithms

December 10, 2019

Louis W. Giokas

Presented by:

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

DesignNews

This Week's Agenda

Monday	The Sensor Fusion Problem
Tuesday	Algorithms
Wednesday	Sensor Types
Thursday	Sensor Fusion
Friday	Applications

Course Description

The use of multiple, heterogeneous sensors is often necessary. This is the case in areas such as robot control, autonomous vehicles and military aviation. Different skills are required including electrical engineering, computer science and statistics. These systems can be complex and include many control theory concepts. In this class we will go over the problem, describe the types of algorithms and sensors used and finally will give some examples.

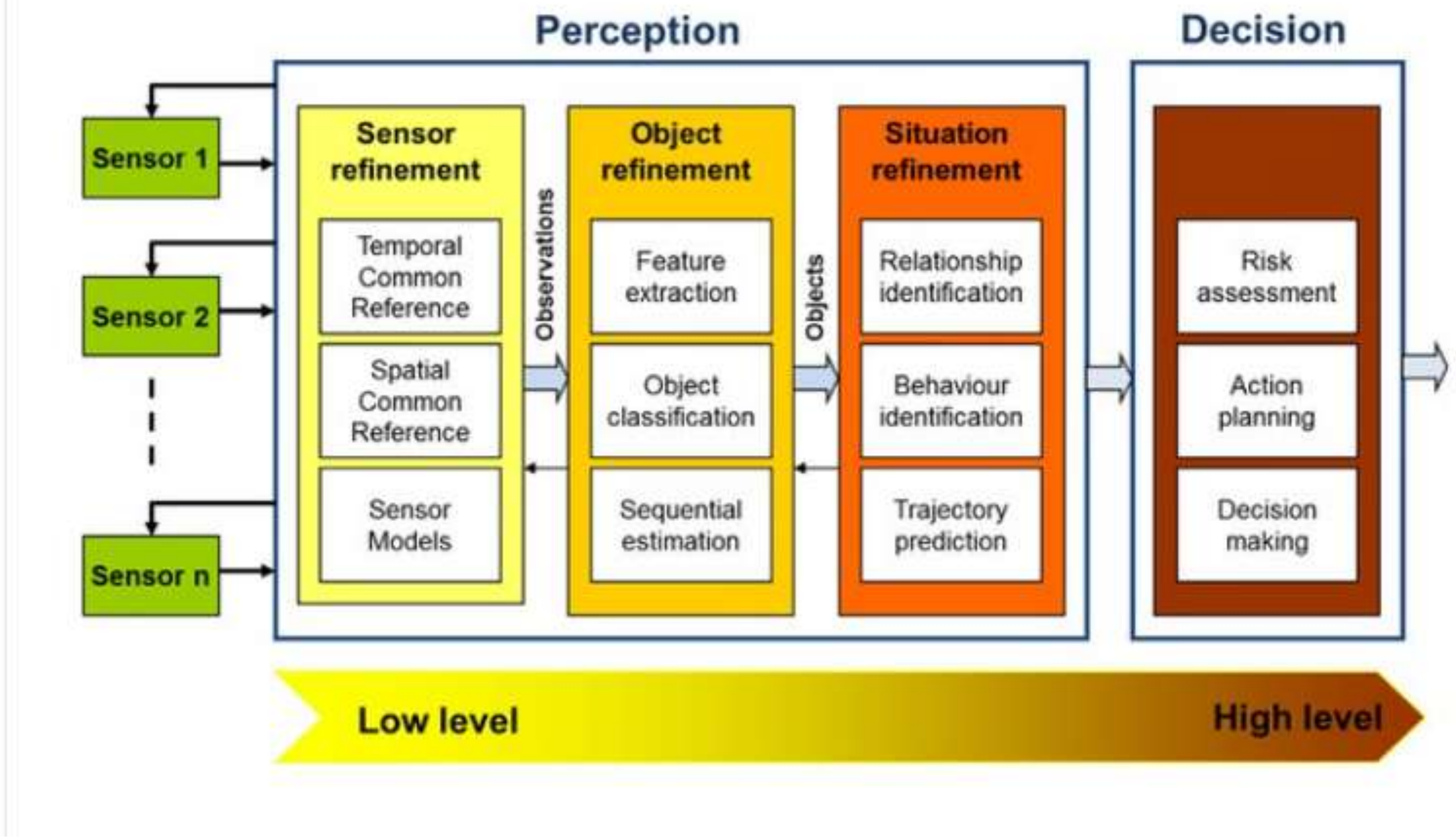
Today's Agenda

- Classes of Algorithms
- Sensor Processing
- Fusion Algorithms
- Conclusion/Next Class

Classes of Algorithms

- There are two types of algorithms we need to consider
  - Sensor processing algorithms
  - Global processing algorithms
- Sensor processing varies widely depending upon the type of sensor and the rate required
- Global processing integrates the sensors to create the predictions needed to get an overall picture of the situation and determine the action required

Classes of Algorithms



Classes of Algorithms

- As we see from the previous slide, we have three levels of processing, each requiring a different set of algorithms
  - Sensor refinement: raw sensor data to observations
  - Object refinement: extract the relevant objects from the observations
  - Situation refinement: objects combined into a global model
- The model is then used to determine some action

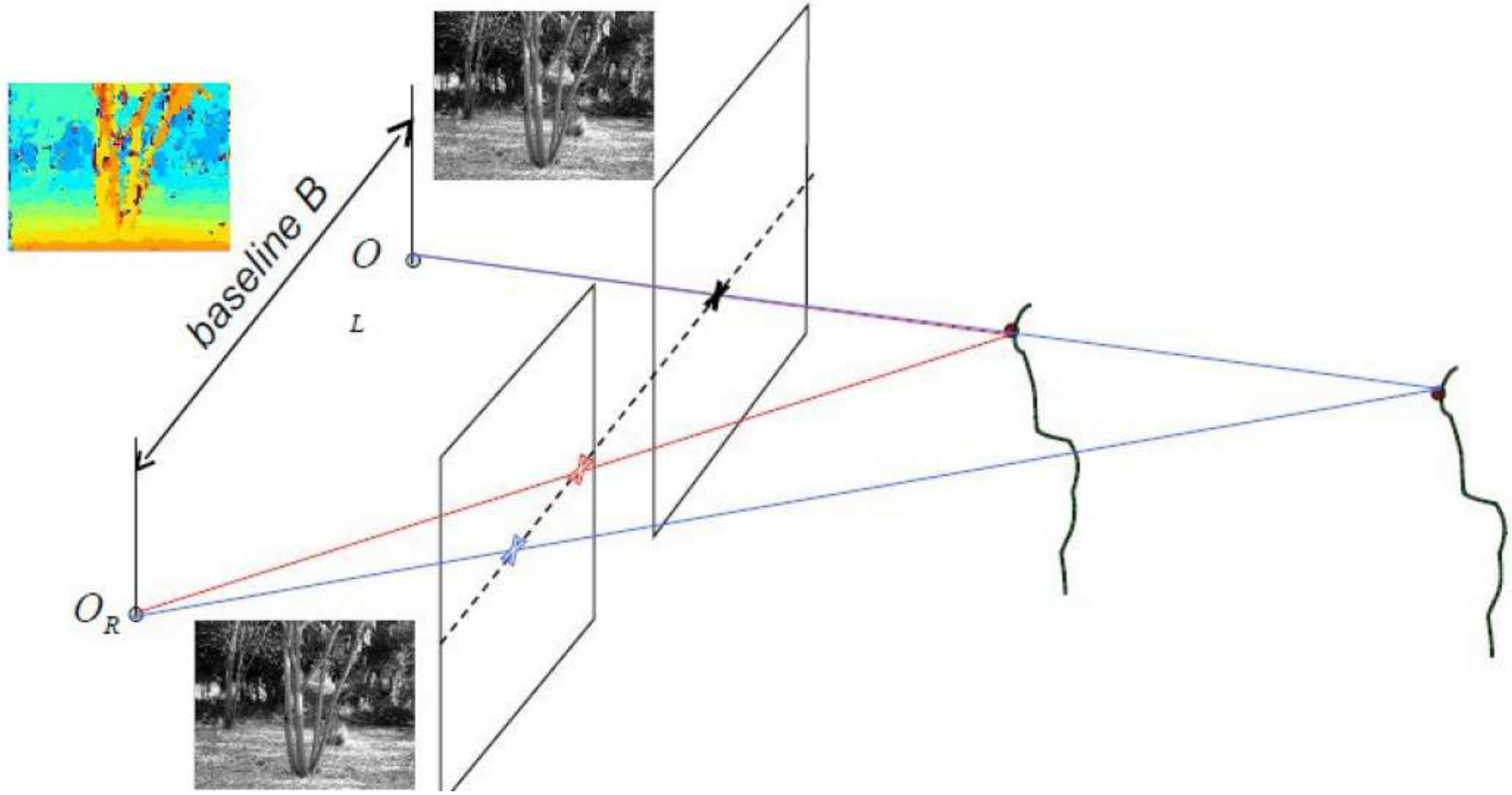
Sensor Processing

- Sensor processing algorithms are unique to each sensor and to the information required of each
  - As such we will not go into great detail for each, but will outline the types of processing typical to some
  - Sensors vary widely in many applications, from image sensors to simple state sensors
  - Rates for sensors can also vary widely; this issue must be handled at the global processing level

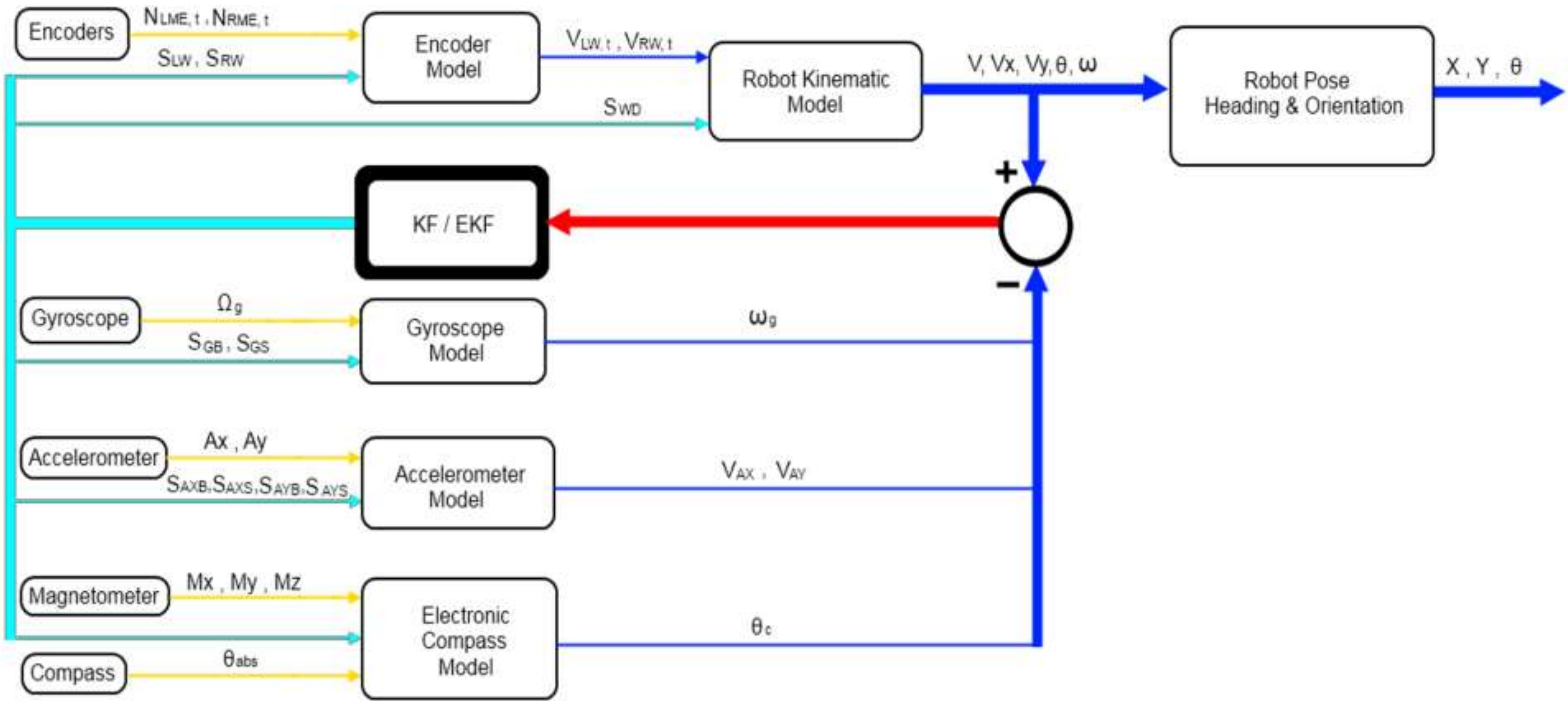
Sensor Processing

- Image type sensors
  - May be used for single image processing or stereo vision
  - Issue with stereo is calibration
  - Once an image has been captured, further processing, such as feature extraction may be performed
    - From features we then perform object classification
    - We may also want information such as depth (relative or absolute)

Sensor Processing



Sensor Processing



Presented by:

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

DesignNews

Sensor Processing

- Processing execution models may be distributed or centralized
  - Many smart sensors have on-board processors and give sensor output in a “digestible” form
    - This is the preferred model for many applications
  - High bandwidth sensors may utilize a dedicated separate processor (say a FPGA)
  - Integration is easier in a distributed system, since each sensor can be programmed and calibrated individually

Fusion Algorithms

- The fusion methods we will look at are statistical methods based on Bayes Rule
- The main methods used are:
  - Kalman Filters
  - Extended Kalman Filters
  - Monte Carlo methods
- Other non-statistical methods can also be used
  - Interval Calculus
  - Fuzzy Logic
  - Evidential Reasoning (Dempster-Shafer)

Fusion Algorithms

- Bayes Rule
  - Inferences about a state x given an observation z
  - Based on the chain-rule of conditional probabilities

$$P(x, z) = P(x \mid z)P(z) = P(z \mid x)P(x).$$

rearranged we get Bayes Rule

$$P(x \mid z) = \frac{P(z \mid x)P(x)}{P(z)}.$$

  - Recursive form

$$P(x \mid Z^k) = \frac{P(z_k \mid x)P(x \mid Z^{k-1})}{P(z_k \mid Z^{k-1})}.$$

Fusion Algorithms

$\boldsymbol{x}_k$: The state vector to be estimated at time k,

$\boldsymbol{u}_k$: A control vector, assumed known, and applied at time $k-1$ to drive the state from $\boldsymbol{x}_{k-1}$ to $\boldsymbol{x}_k$ at time k,

z_k: An observation taken of the state $\boldsymbol{x}_k$ at time k.

The history of states:
$\boldsymbol{X}^k = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_k\} = \{\boldsymbol{X}^{k-1}, \boldsymbol{x}_k\}$.
The history of control inputs:
$\boldsymbol{U}^k = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_k\} = \{\boldsymbol{U}^{k-1}, \boldsymbol{u}_k\}$.
The history of state observations:
$\boldsymbol{Z}^k = \{z_1, z_2, \cdots, z_k\} = \{\boldsymbol{Z}^{k-1}, z_k\}$.

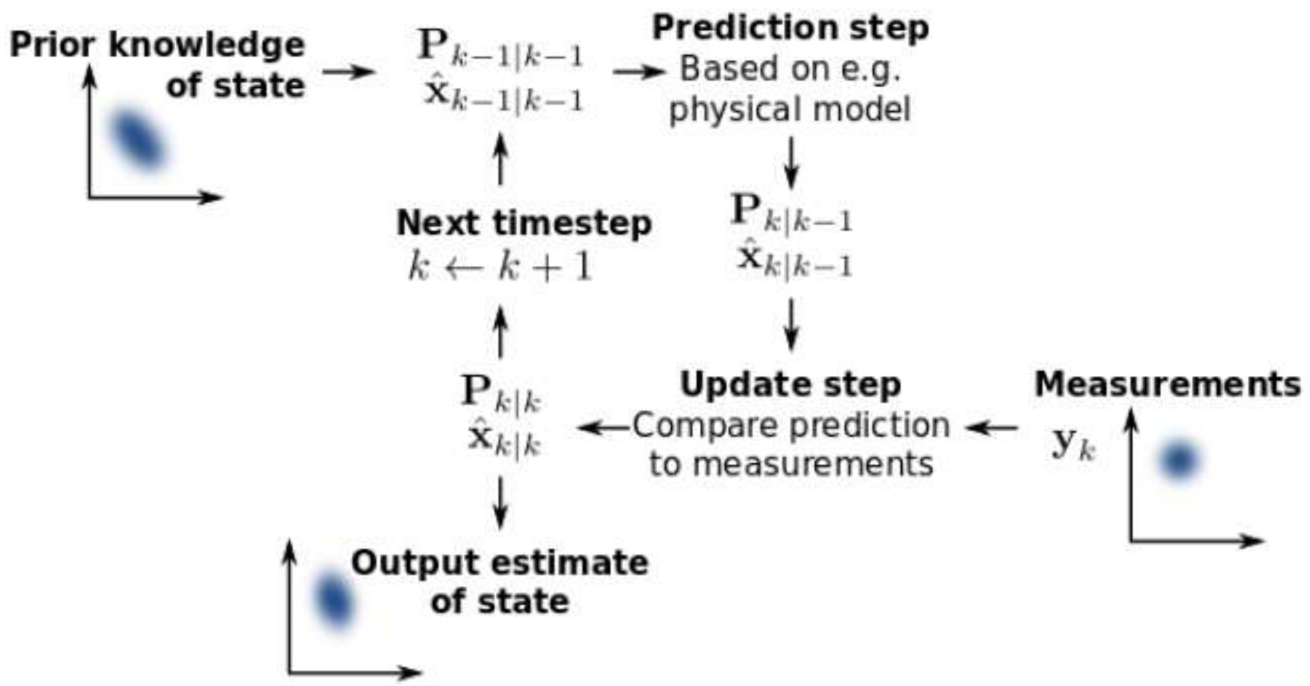
$$P(\boldsymbol{x}_k \mid \boldsymbol{Z}^k, \boldsymbol{U}^k, \boldsymbol{x}_0) = \frac{P(z_k \mid \boldsymbol{x}_k) P(\boldsymbol{x}_k \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k, \boldsymbol{x}_0)}{P(z_k \mid \boldsymbol{Z}^{k-1}, \boldsymbol{U}^k)}.$$

Fusion Algorithms

- Kalman Filter
  - A recursive linear estimator
  - Relates the time evolution of a parameter of interest, x(t), to observations, z(t)
  - We have an explicit model of how these observations are related to x(t)
  - Our goal is to minimize the mean squared error

$$\hat{x}(t) = \mathrm{E}[x(t) \mid Z^t]$$

Fusion Algorithms



Fusion Algorithms

$$\dot{x}(t) = F(t)x(t) + B(t)u(t) + G(t)v(t)$$

$$z(t) = H(t)x(t) + D(t)w(t)$$

Standard continuous state-space form; x is the state; u is a known control input; v and are random variables describing uncertainty; F, B, G, H, D are matricies describing the contribution of states

$$x(k) = F(k)x(k-1) + B(k)u(k) + G(k)v(k)$$

$$z(k) = H(k)x(k) + D(k)w(k)$$

Discrete-time version

$$\hat{x}(k \mid k-1) = F(k)\hat{x}(k-1 \mid k-1) + B(k)u(k)$$

Prediction

$$P(k \mid k-1) = F(k)P(k-1 \mid k-1)F^{\mathrm{T}}(k) + G(k)Q(k)G^{\mathrm{T}}(k) .$$

Fusion Algorithms

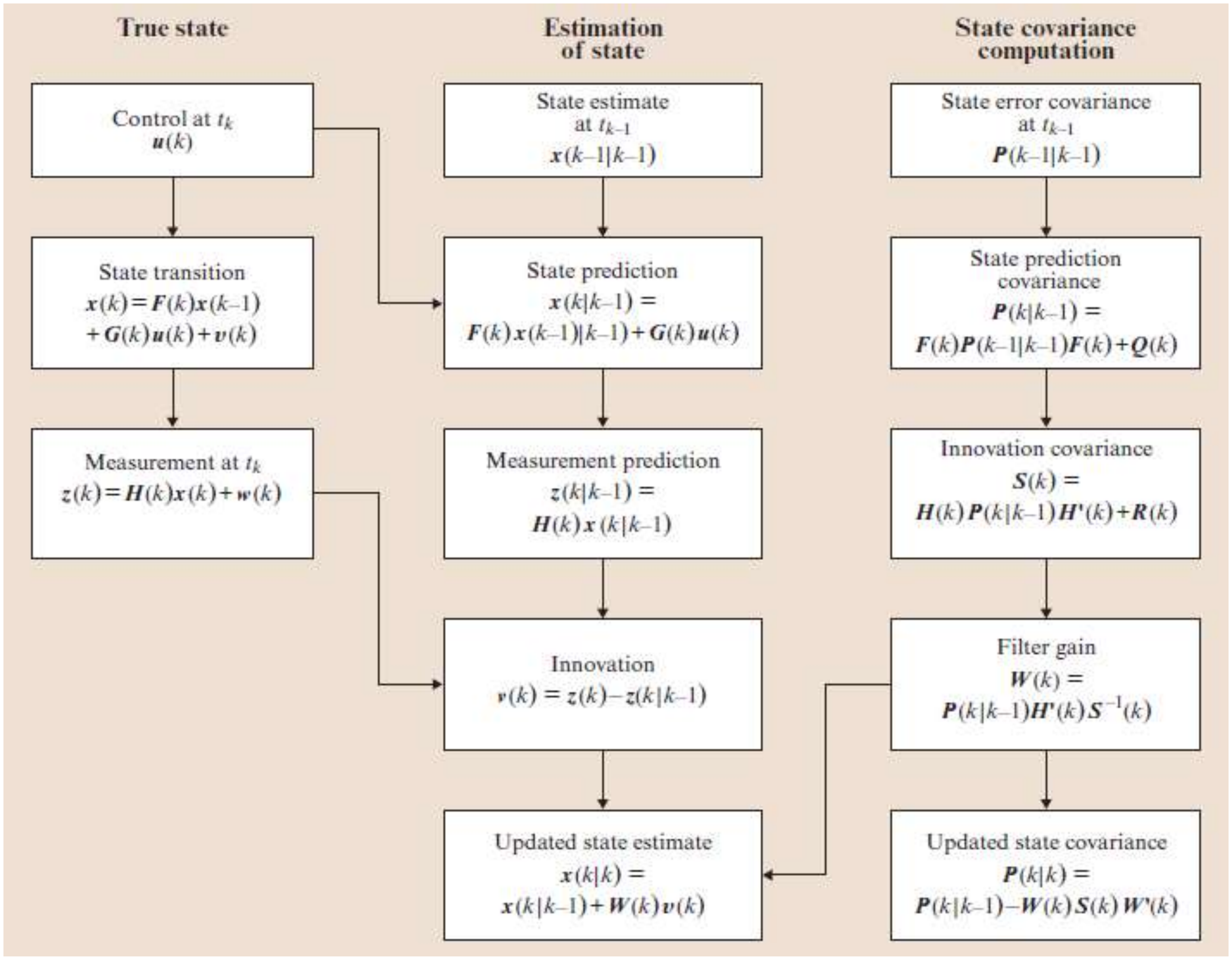
- Extended Kalman Filter is used when one or both of the state model and the observation model are non-linear
- The state space is then described by a first order nonlinear vector differential equation

$$\dot{x}(t) = f[x(t), u(t), v(t), t]$$

The observations are described by a nonlinear vector function

$$z(t) = h[x(t), u(t), w(t), t]$$

Fusion Algorithms


True state
Estimation of state
State covariance computation
Control at t_k $\boldsymbol{u}(k)$
State transition $\boldsymbol{x}(k)=\boldsymbol{F}(k)\boldsymbol{x}(k-1)+\boldsymbol{G}(k)\boldsymbol{u}(k)+\boldsymbol{v}(k)$
Measurement at t_k $\boldsymbol{z}(k)=\boldsymbol{H}(k)\boldsymbol{x}(k)+\boldsymbol{w}(k)$
State estimate at t_{k-1} $\boldsymbol{x}(k-1|k-1)$
State prediction $\boldsymbol{x}(k|k-1)=\boldsymbol{F}(k)\boldsymbol{x}(k-1)|k-1)+\boldsymbol{G}(k)\boldsymbol{u}(k)$
Measurement prediction $\boldsymbol{z}(k|k-1)=\boldsymbol{H}(k)\boldsymbol{x}(k|k-1)$
Innovation $\boldsymbol{v}(k)=\boldsymbol{z}(k)-\boldsymbol{z}(k|k-1)$
Updated state estimate $\boldsymbol{x}(k|k)=\boldsymbol{x}(k|k-1)+\boldsymbol{W}(k)\boldsymbol{v}(k)$
State error covariance at t_{k-1} $\boldsymbol{P}(k-1|k-1)$
State prediction covariance $\boldsymbol{P}(k|k-1)=\boldsymbol{F}(k)\boldsymbol{P}(k-1|k-1)\boldsymbol{F}(k)+\boldsymbol{Q}(k)$
Innovation covariance $\boldsymbol{S}(k)=\boldsymbol{H}(k)\boldsymbol{P}(k|k-1)\boldsymbol{H}'(k)+\boldsymbol{R}(k)$
Filter gain $\boldsymbol{W}(k)=\boldsymbol{P}(k|k-1)\boldsymbol{H}'(k)\boldsymbol{S}^{-1}(k)$
Updated state covariance $\boldsymbol{P}(k|k)=\boldsymbol{P}(k|k-1)-\boldsymbol{W}(k)\boldsymbol{S}(k)\boldsymbol{W}'(k)$

Fusion Algorithms

- Monte Carlo Method
  - Simulates the recursive Bayes update equations using sample support values and weights to describe the underlying probability distributions
  - For low dimensional state spaces this can be a look up table using interpolation. For higher dimensional cases, the preferred method is to provide the representation as a function
  - Resampling is used to update support values to keep the weights equal

Fusion Algorithms

$$\{x^i_{k-1}, w^i_{k-1|k-1}\}_{i=1}^{N_{k-1}}$$

Support values and weights

$$P(x_{k-1} \mid Z^{k-1}) = \sum_{i=1}^{N_{k-1}} w^i_{k-1}\delta(x_{k-1} - x^i_{k-1})$$

Posterior probability density

$$q_k(x^i_k) = P(x^i_k \mid x^i_{k-1})$$

Importance density

$$w^i_k = \frac{w^i_{k-1} P(z_k = z_k \mid x_k = x^i_k)}{\sum_{j=1}^{N_k} w^j_{k-1} P(z_k = z_k \mid x_k = x^j_k)}$$

Updated set of normalized weights

$$P(x_k \mid Z^k) = \sum_{i=1}^{N_k} w^i_k \delta(x_k - x^i_k)$$

Posterior probability density function with weights changed by observation update

Conclusion/Next Class

- Today we have discussed the algorithms used for multi-sensor data fusion
- We briefly discussed the processing of specific sensors
- We reviewed in detail the main statistical methods of global sensor processing (fusion)
- Tomorrow we will go over a sampling of the types of sensors that might be encountered