

fortiss

Parking Team

Sensor Data Fusion

July 4, 19

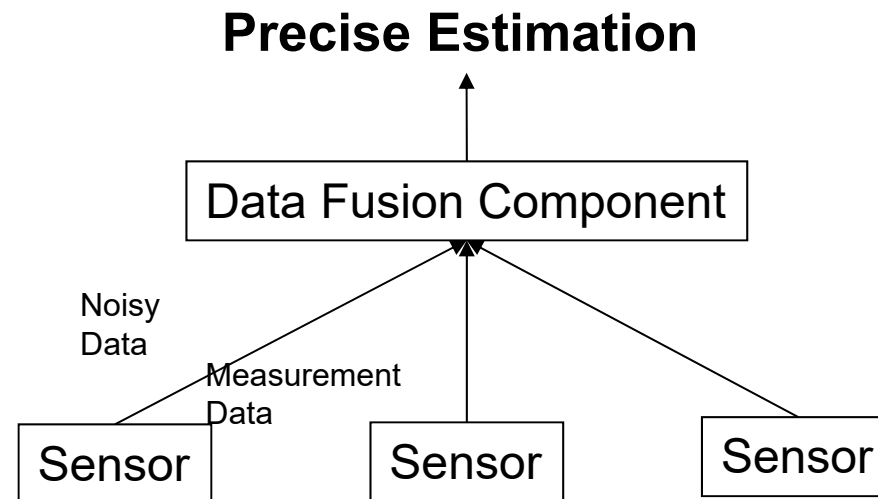
Content

- Introduction to Sensor Data Fusion
- Data Fusion Architecture
- Sensor Fusion Functions
- Relation to our Project

Introduction to Sensor Data Fusion

Definitions:

Sensor fusion is combining of sensory data or data derived from sensor sources such that the resulting information has less uncertainty.



Introduction to Sensor Data Fusion

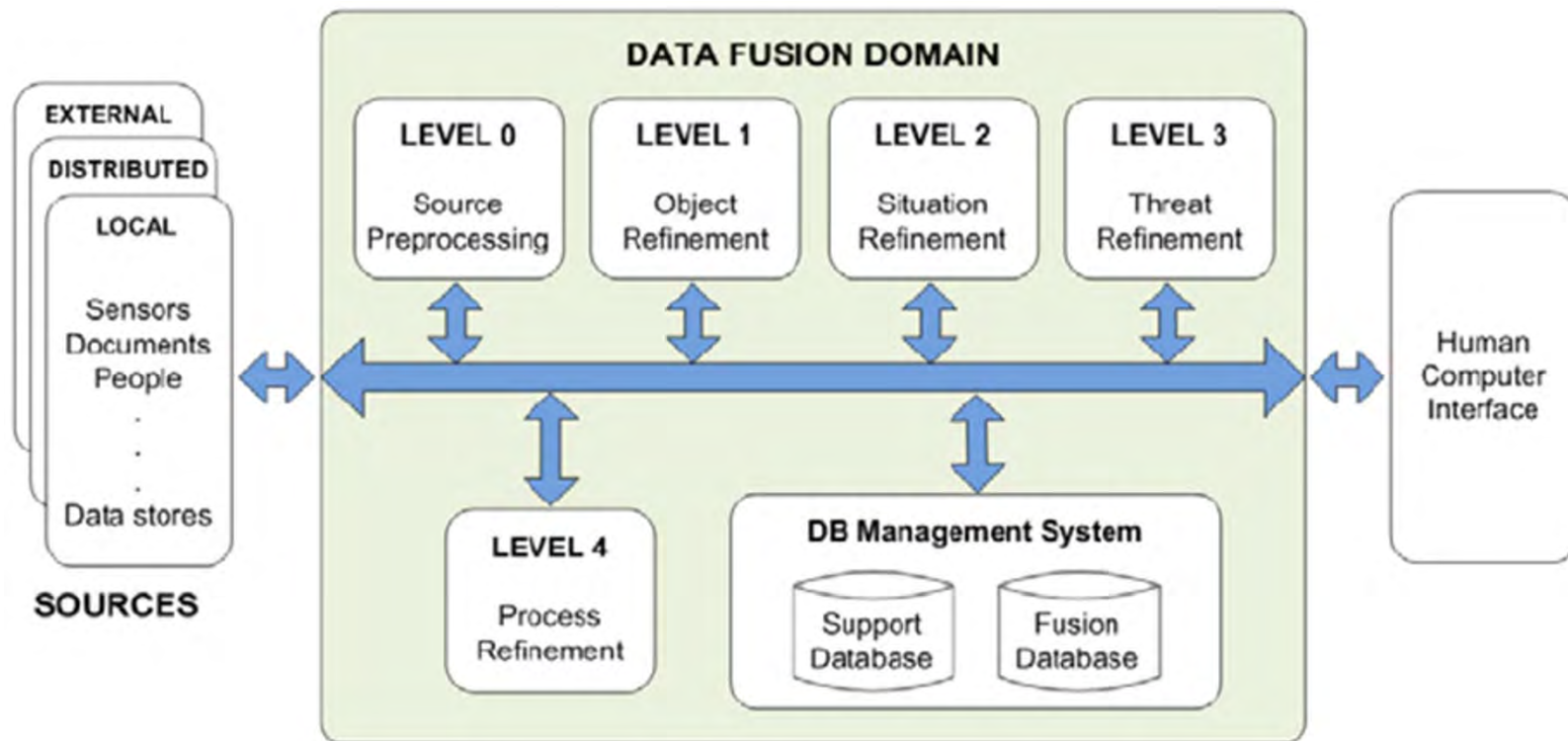
Advantages:

- Improves accuracy
- Improves precision
- Improves availability
- Reduces uncertainty
- Supports effective decision making

Challenges:

- There is no guarantee for the raw data quality
- The fused answer may be worse than the sensor
- There are no magic algorithms
- There will never be enough training data
- It is difficult to quantify the value of data fusion

Data Fusion Architecture

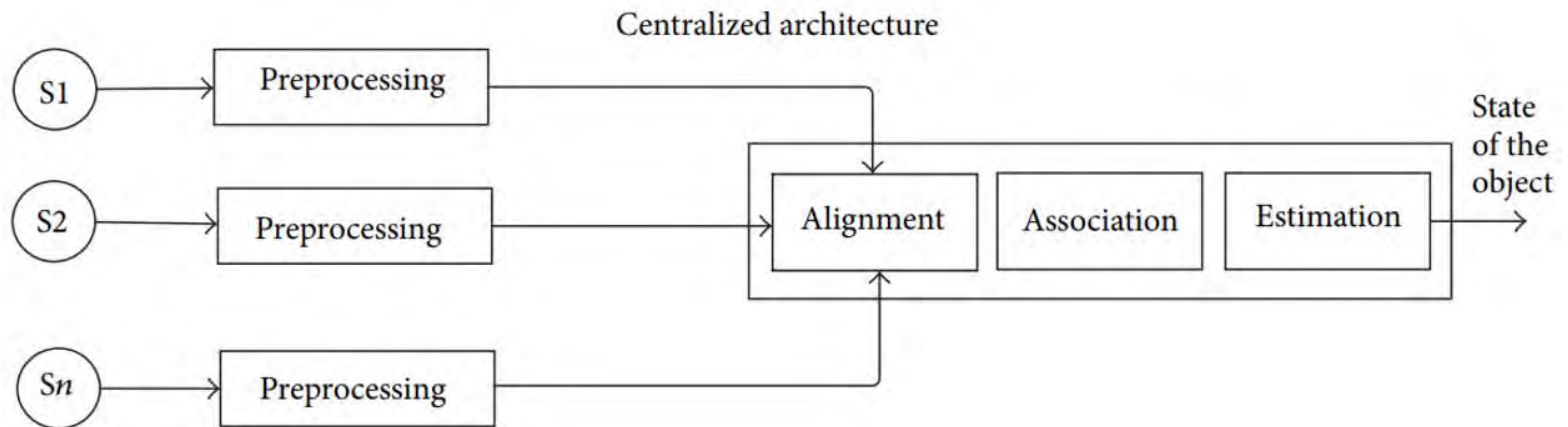


Data Fusion Architecture

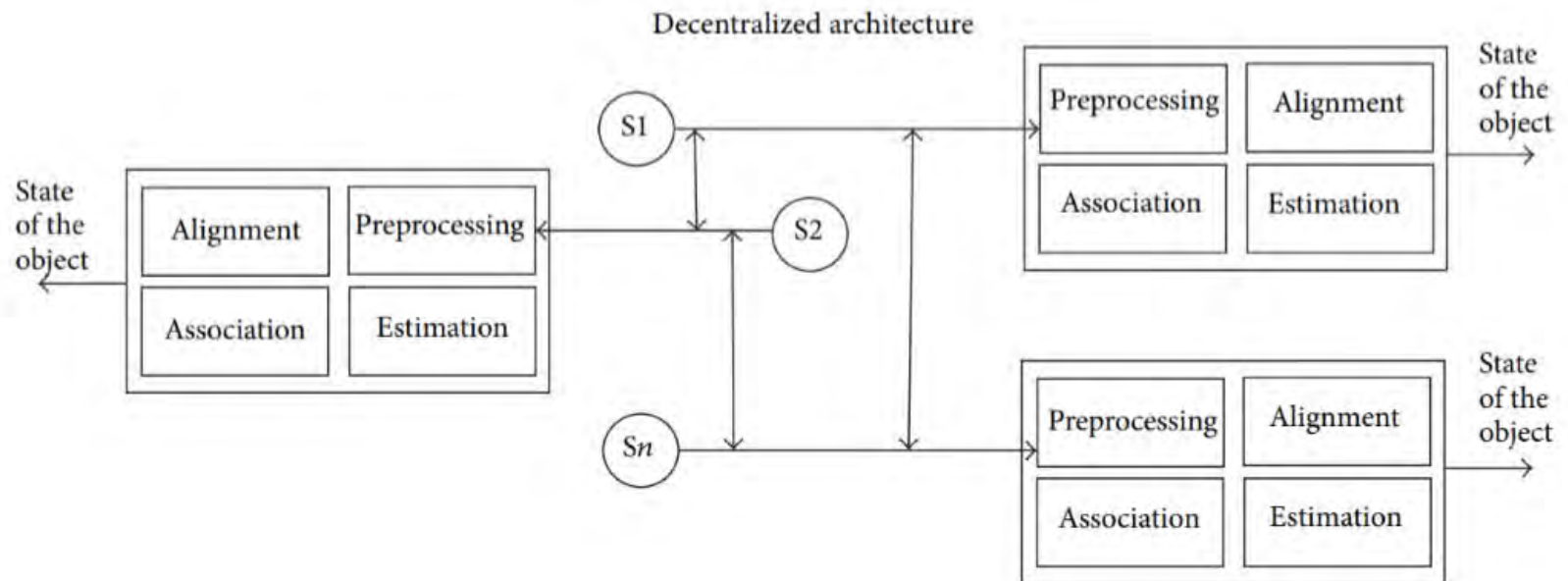
- Level 0—Sub-Object Data Association and Estimation
- Level 1—Object Refinement
- Level 2—Situation Refinement
- Level 3—Significance Estimation
- Level 4—Process Refinement



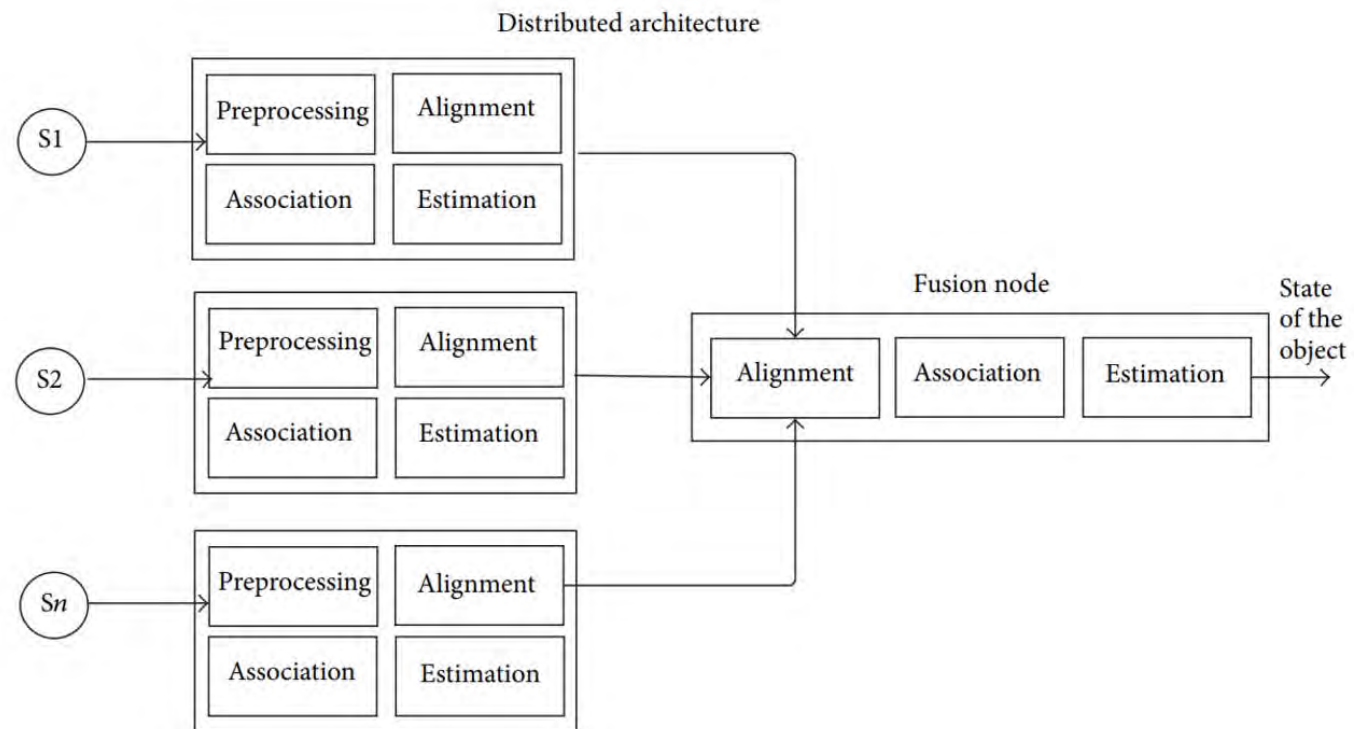
Data Fusion Architecture



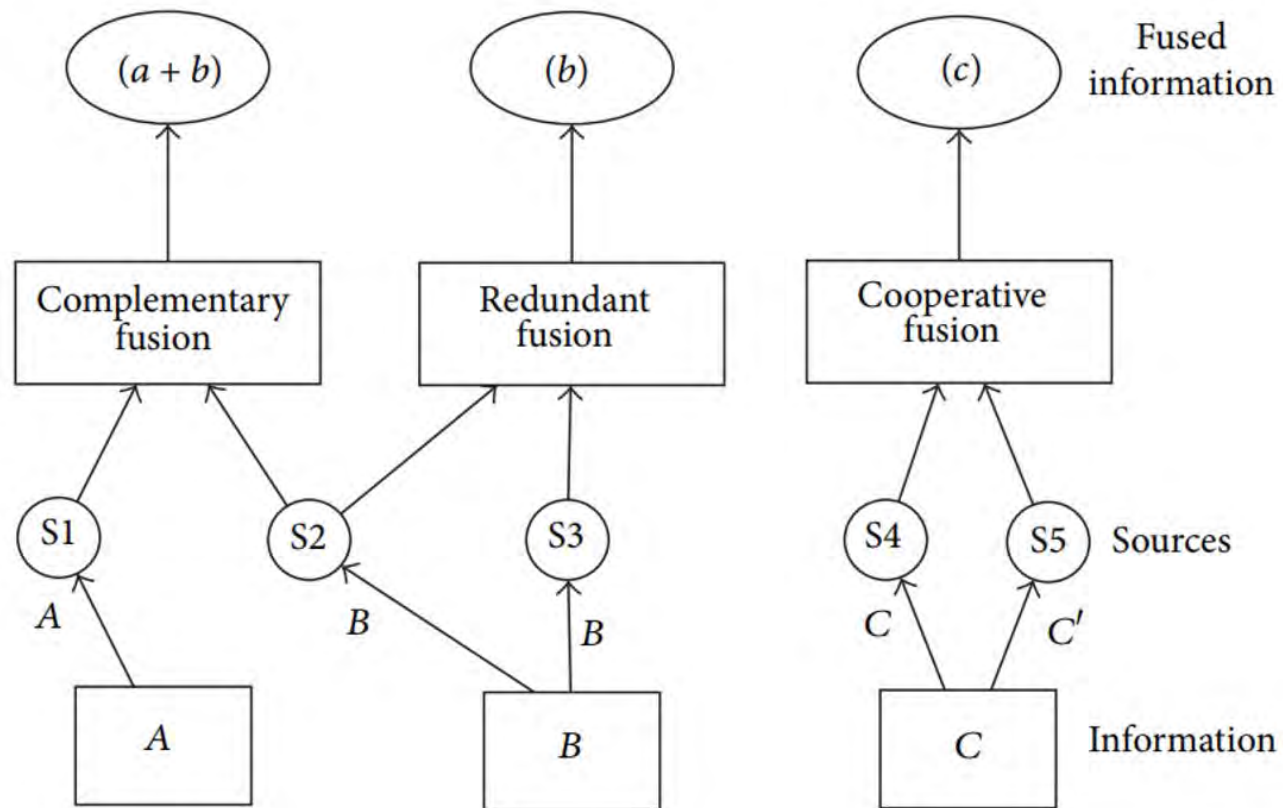
Data Fusion Architecture



Data Fusion Architecture



Data Fusion Architecture



Sensor Fusion Functions

- By splitting Sensor Data Fusion into these subtasks, they can be studied individually

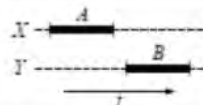
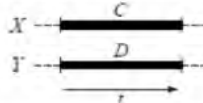

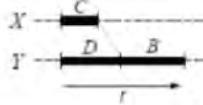
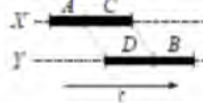
- Data Fusion functions include:

Data Alignment, Data Association, State estimation



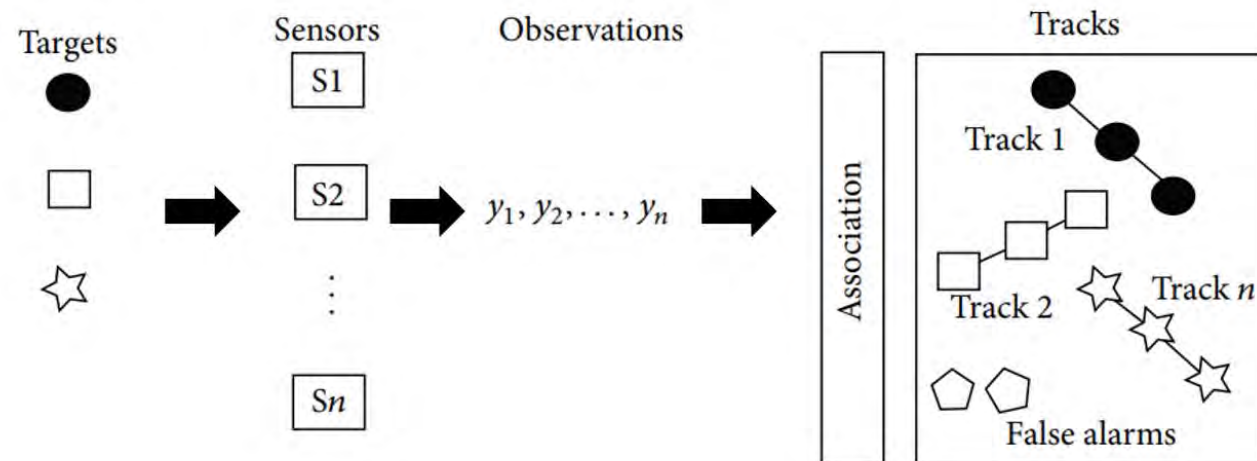
Sensor Fusion Functions

- Data Alignment
- Detect temporal differences
- Adjust the collected data
- Synchronize using shared time.
- Synchronize via communication protocol.

Logical mappings	Scenario example
<i>precedes:</i> $A \rightarrow B$	
<i>simultaneous:</i> $C D$	
<i>ends:</i> $A \rightarrow (C D)$	
<i>starts:</i> $(C D) \rightarrow B$	
<i>overlaps:</i> $A \rightarrow (C D) \rightarrow B$	

Sensor Fusion Functions

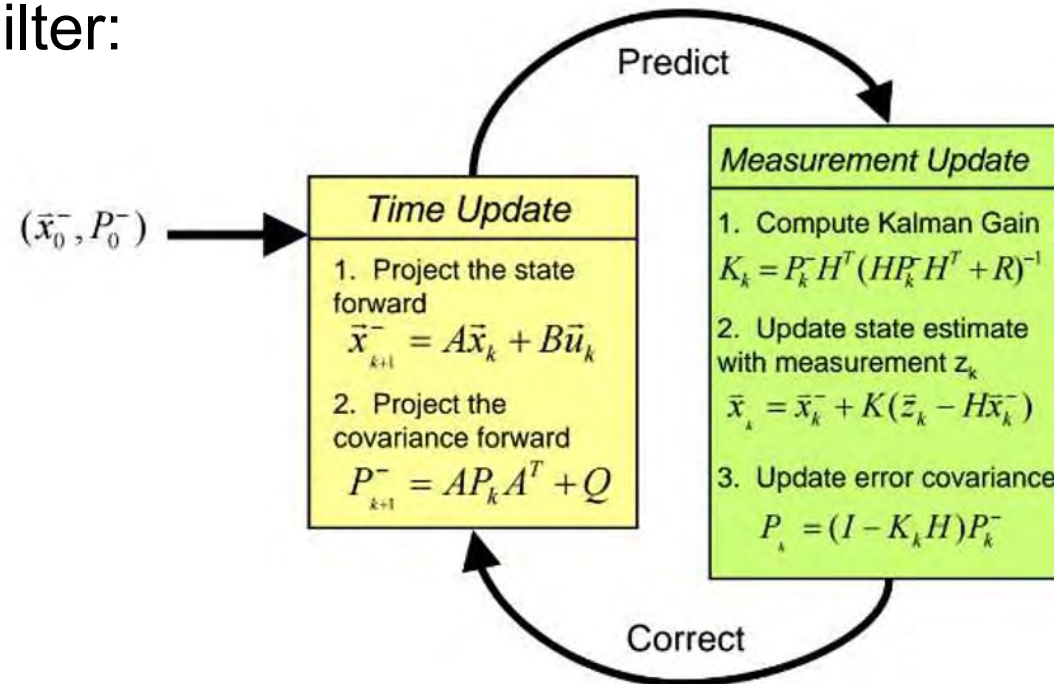
- Data Association
- Clustering: Nearest Neighbors and K-Means, Probabilistic Data Association



Sensor Fusion Functions

- State Estimation Methods

Kalman Filter:



Sensor Fusion Functions

- State Estimation Methods

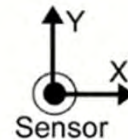
A standard Kalman Filter can only handle linear models


- Non-linear process and sensor models:

Extended Kalman Filter (EKF) Unscented Kalman Filter (UKF)

- Extended Kalman Filter uses the Jacobian matrix to linearize non-linear functions
- Unscented Kalman Filter takes representative points from a Gaussian distribution

Sensor Fusion Functions



- ☒  Lidar Measurement
- ☒  Radar Measurement

RMSE
X: 0.1067
Y: 0.165928
VX: 0.276554
VY: 0.634738

☒ Record Data

Run

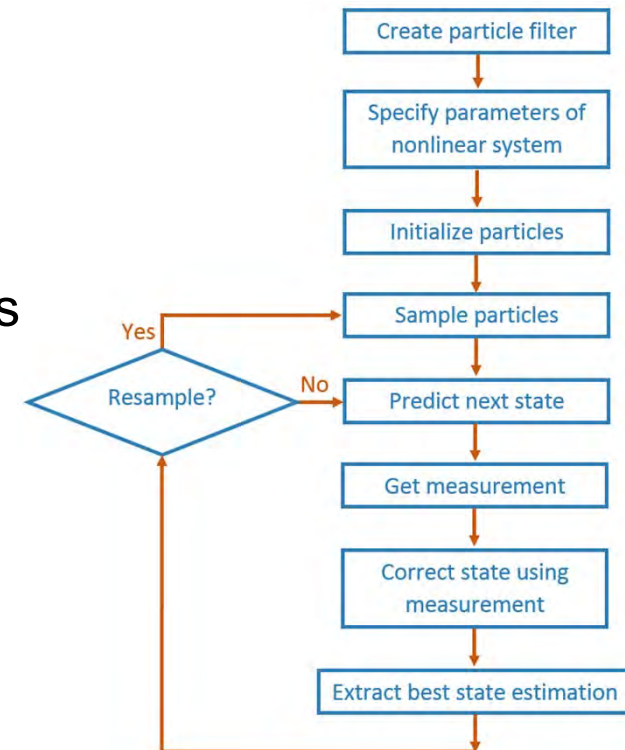
Sensor Fusion Functions

- State Estimation Methods

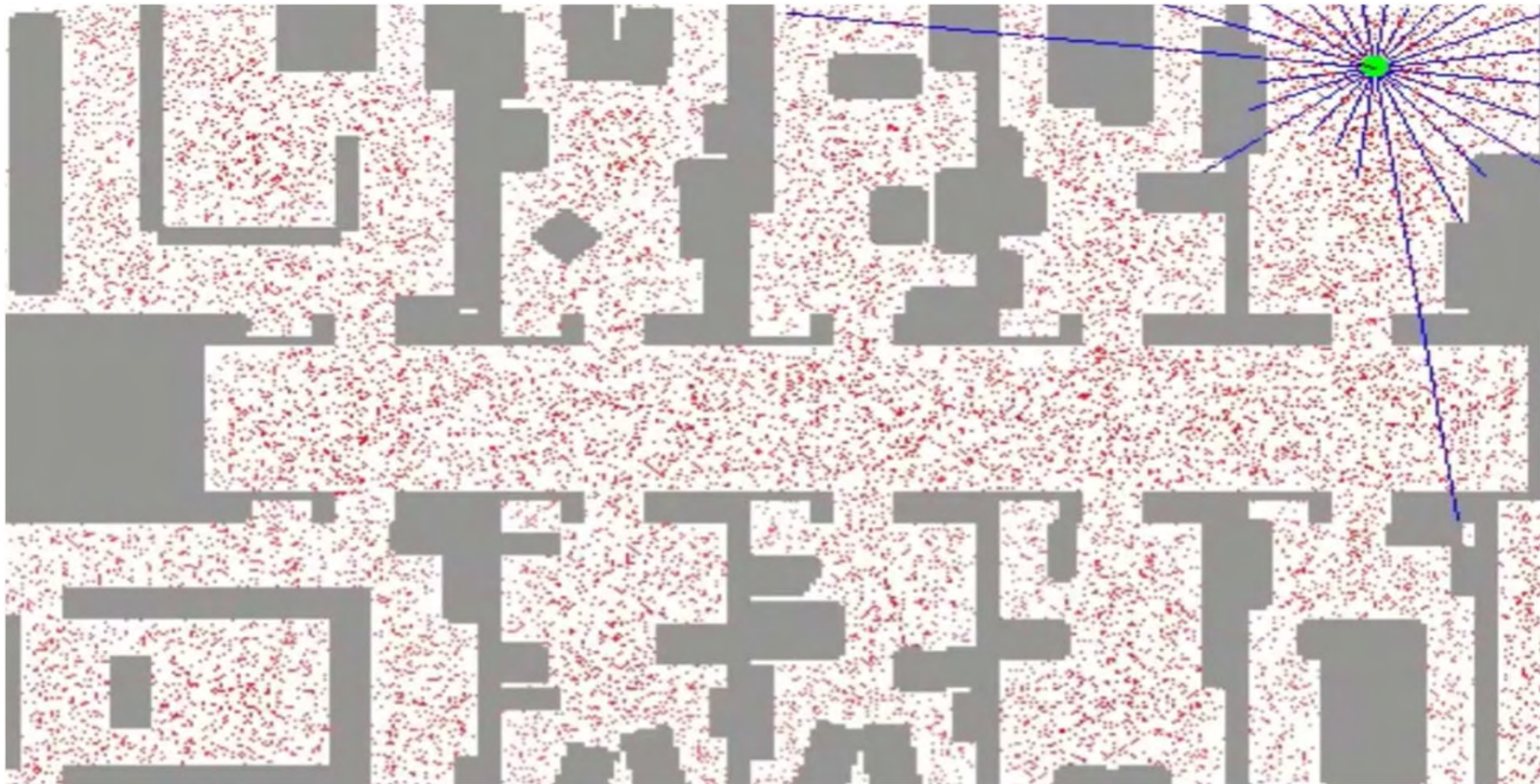
Particle Filter:

Particle filters is a set of Monte Carlo algorithms used to solve filtering problems

- To handle non linear processes
- To handle non Gaussian Noise

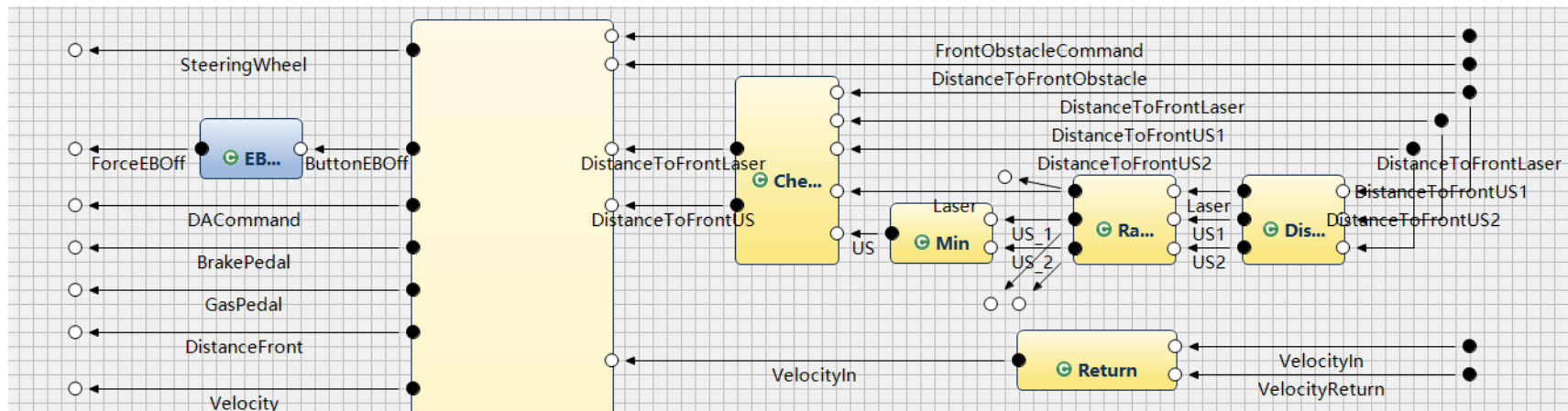


Sensor Fusion Functions



Relation to our Project

- AF3 – Sensor Data Fusion



Relation to our Project

- AF3 – Sensor Data Fusion

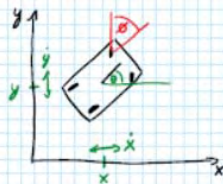
```
if (FrontObstacleCommand != NoVal && FrontObstacleCommand) {  
    DistanceFront = DistanceToFrontObstacle;  
}  
else {  
    if (DistanceToFrontLaser == NoVal && DistanceToFrontUS == NoVal) {  
        DistanceFront = NoVal;  
    }  
    else {  
        if (DistanceToFrontUS != NoVal && (DistanceToFrontLaser == NoVal || DistanceToFrontUS <= DistanceToFrontLaser)) {  
            DistanceFront = DistanceToFrontUS;  
        }  
        else {  
            DistanceFront = DistanceToFrontLaser;  
        }  
    }  
}
```


Relation to our Project

EKF for a car

Mittwoch, 3. Juli 2019 10:39

$$\vec{x} = \begin{pmatrix} x \\ y \\ \theta \\ \delta \\ \dot{x} \\ \dot{y} \end{pmatrix} \begin{array}{l} x\text{-Position} \\ y\text{-Position} \\ \text{yaw} \\ \text{steering angle} \\ x\text{-Velocity} \\ y\text{-Velocity} \end{array}$$



$$\vec{u} = \begin{pmatrix} v_c \\ \delta_c \end{pmatrix} \begin{array}{l} \text{command velocity} \\ \text{command steering} \end{array}$$

Predict

$$\vec{x}_{t|t-1} = f(\vec{x}_{t-1|t-1}, u_t)$$

$$\vec{P}_{t|t-1} = F_t \cdot \vec{P}_{t-1|t-1} \cdot F_t^T + W_t \cdot \vec{Q}_t \cdot W_t^T$$

$$\vec{f} = \begin{pmatrix} dt \cdot \dot{x} + x \\ dt \cdot \dot{y} + y \\ dt \cdot \frac{1}{L} \cdot \tan(\delta_c) \cdot v_c + \theta \\ \delta_c \\ \cos(\theta) \cdot v_c \\ \sin(\theta) \cdot v_c \end{pmatrix}$$

$$F_t = \left[\frac{\partial f}{\partial \vec{x}_t} \right] = \begin{pmatrix} 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\sin(\theta) \cdot v_c & 0 & 0 & 0 \\ 0 & 0 & \cos(\theta) \cdot v_c & 0 & 0 & 0 \end{pmatrix}$$

$$W_t = \left[\frac{\partial f}{\partial u_t} \right] = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \frac{dt}{L} \tan \delta_c & \frac{dt}{L} \sec^2 \delta_c \cdot v_c \\ 0 & 1 \\ \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

$$\vec{Q}_t = \begin{pmatrix} \sigma_v^2 & \sigma_{\delta v} \\ \sigma_{\delta v} & \sigma_{\delta}^2 \end{pmatrix}$$

Update

Separated for every sensor

$$y_t = z_t - h(\vec{x}_{t|t-1})$$

$$S_t = H_t \cdot \vec{P}_{t|t-1} \cdot H_t^T + R_t$$

$$K_t = \vec{P}_{t|t-1} \cdot H_t^T \cdot S_t^{-1}$$

$$\vec{x}_{t|t} = \vec{x}_{t|t-1} + K_t \cdot y_t$$

$$\vec{P}_{t|t} = (I - K_t \cdot H_t) \cdot \vec{P}_{t|t-1}$$

LIDAR Update

$$z_t = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \quad h(x) = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

$$H_t = \left[\frac{\partial h}{\partial \vec{x}_t} \right] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$R_t \in \mathbb{R}^{3 \times 3} \\ R_t = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

IMU Update

$$z_t = \theta_{\text{imu}} \quad h(x) = \theta$$

$$H_t = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$R_t \in \mathbb{R} \quad R_t = \sigma_{\theta}^2$$

actual velocity update

$$z_t = v_a \quad h(x) = \sqrt{\dot{x}^2 + \dot{y}^2}$$

$$H_t = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}} & \frac{\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}} \end{pmatrix}$$

$$R_t \in \mathbb{R} \quad R_t = \sigma_v^2$$

Resources

- Gereon Hinz, lecture Autonomous Fahren
- Darius Burschka. lecture Robot Motion Planning
- https://en.wikipedia.org/wiki/Extended_Kalman_filter
- <https://www.youtube.com/watch?v=XswKMtQBTC0>
- https://en.wikipedia.org/wiki/Expectation-maximization_algorithm
- https://en.wikipedia.org/wiki/Monte_Carlo_method