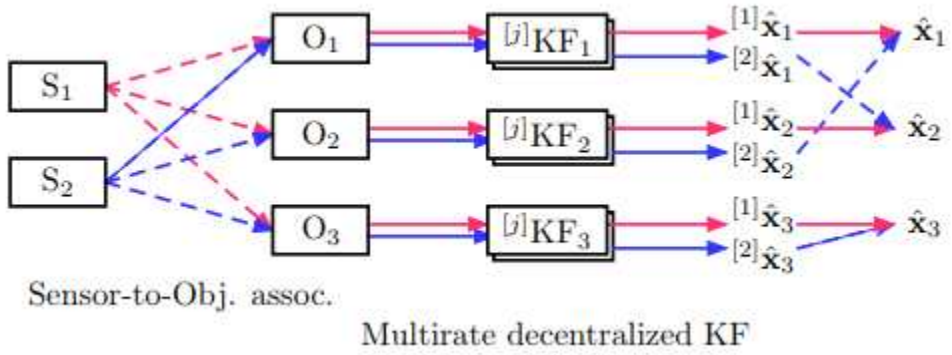
The problem we are to consider is the driving environment modeling around the ego vehicle.
Let us introduce a simple scenario. We assume the vehicle has two sensors to measure the relative distances to each object vehicle. Also assume there are three object vehicles around. Your mission to to track object vehicles using measurements by the sensors for driving environment modeling.
The situation we consider is depicted in the following figure:



Here, Sj denotes sensor-j with j = 1, 2, Oi denotes object-i with i = 1, 2, 3, [j]KFi denote Kalman filters for the objects Oi using the associated measurements [j]yi by Sj, [j]x^i denote the estimates of states xi for the objects Oi using the measurements of sensors Sj . Dashed lines denote unknown 'association'.

You want to have state estimates xi at every Tc using the sensor measurements [j]yi at every [j]Ts = [j]RTc. You will need develop 'Sensor-to-Object association', 'Multirate decentralized KF', and 'Obj-to-Obj association & Fusion'. Simply use the CV model for Oi in {xyz} in terms of the state vector xi = [xi yi ẋi ẏi]^T :

$$\mathbf{x}_i(k+1) = \Phi_{cv}\mathbf{x}_i(k) + \mathbf{w}_i(k) = \begin{bmatrix} 1 & 0 & T_c & 0 \\ 0 & 1 & 0 & T_c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_i(k) + \begin{bmatrix} T_c \\ T_c \\ 1 \\ 1 \end{bmatrix} w_i(k)$$

where wi(k) denotes process disturbance associated with Oi's CV motion with wi ~ N (0, αi). Assume αi > 0 at your preference. Assume yi =[xiyi]'is updated at each sampling instant Tskv, kv = 1, 2 ...
In our notation, the subscript '[j]' denotes associated sensors: [j]xi denotes the state of object vehicle - i associated with the sensor - j.
[j]yi denotes the observation of object vehicle - i associated with the sensor - j, and so on.

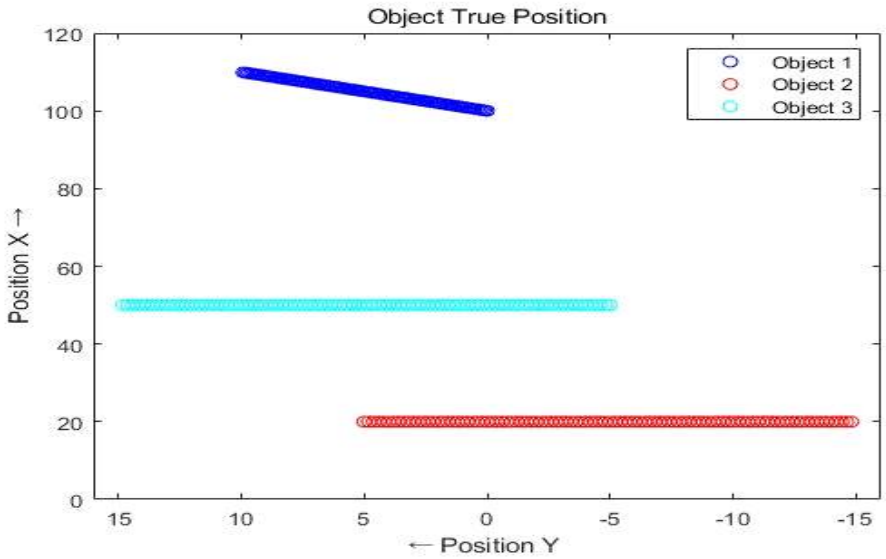
1) Object의 초기 위치와 속도 설정

Initial condition

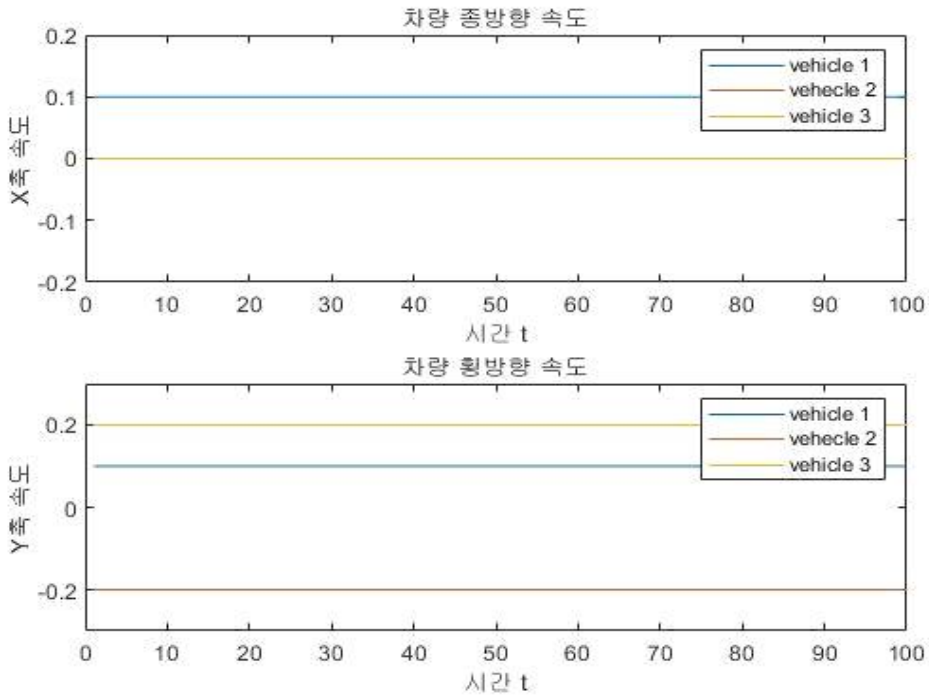
x(1) = [x y $\dot{x}$ $\dot{y}$] = [80 0 0.1 0.1]

x(2) = [x y $\dot{x}$ $\dot{y}$] = [20 5 0 -0.2]

x(3) = [x y $\dot{x}$ $\dot{y}$] = [50 -5 0 0.5]

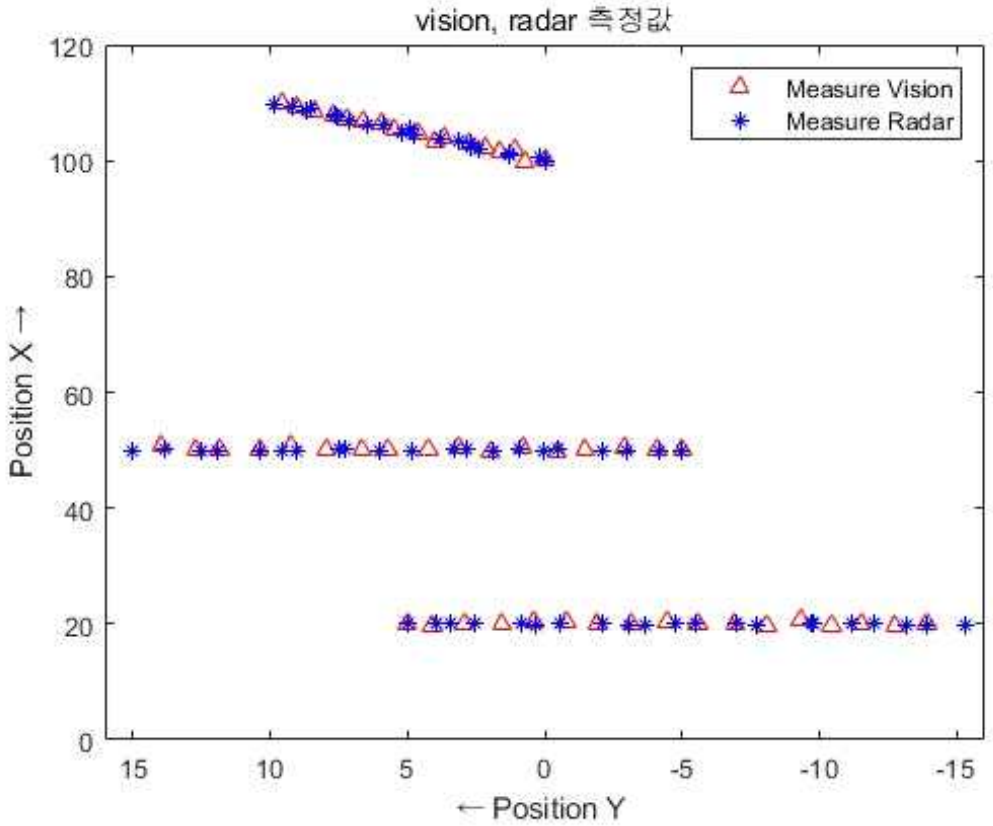


2) Object의 상대 속도(횡방향/종방향)

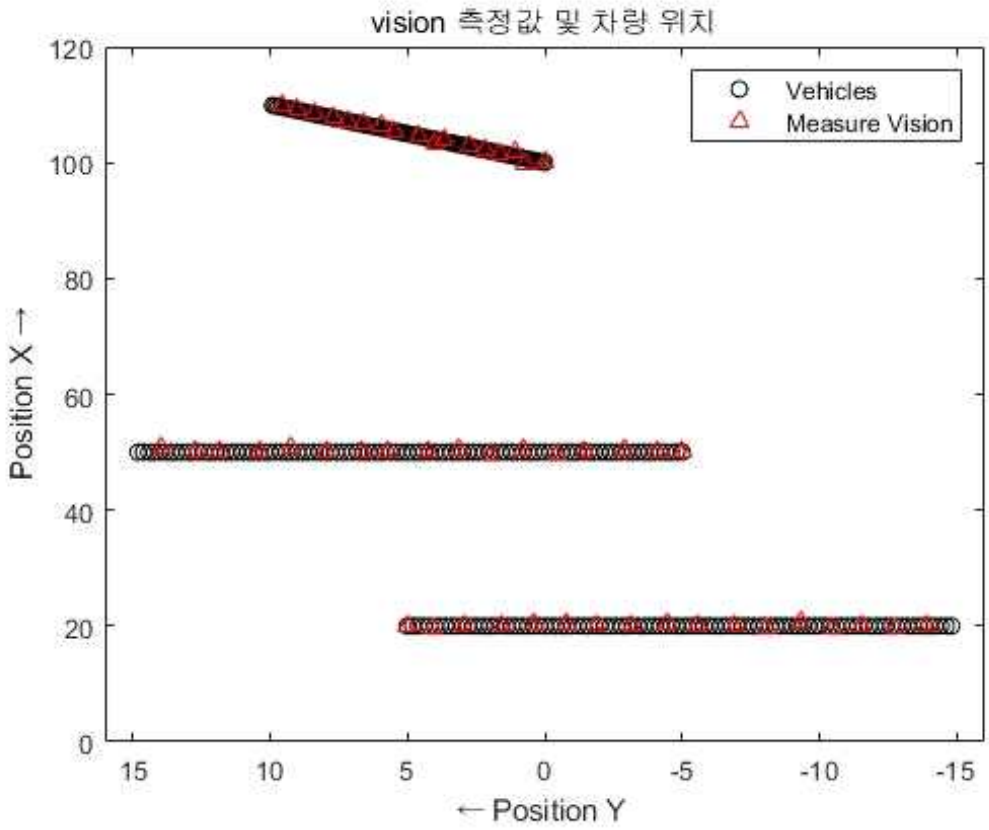


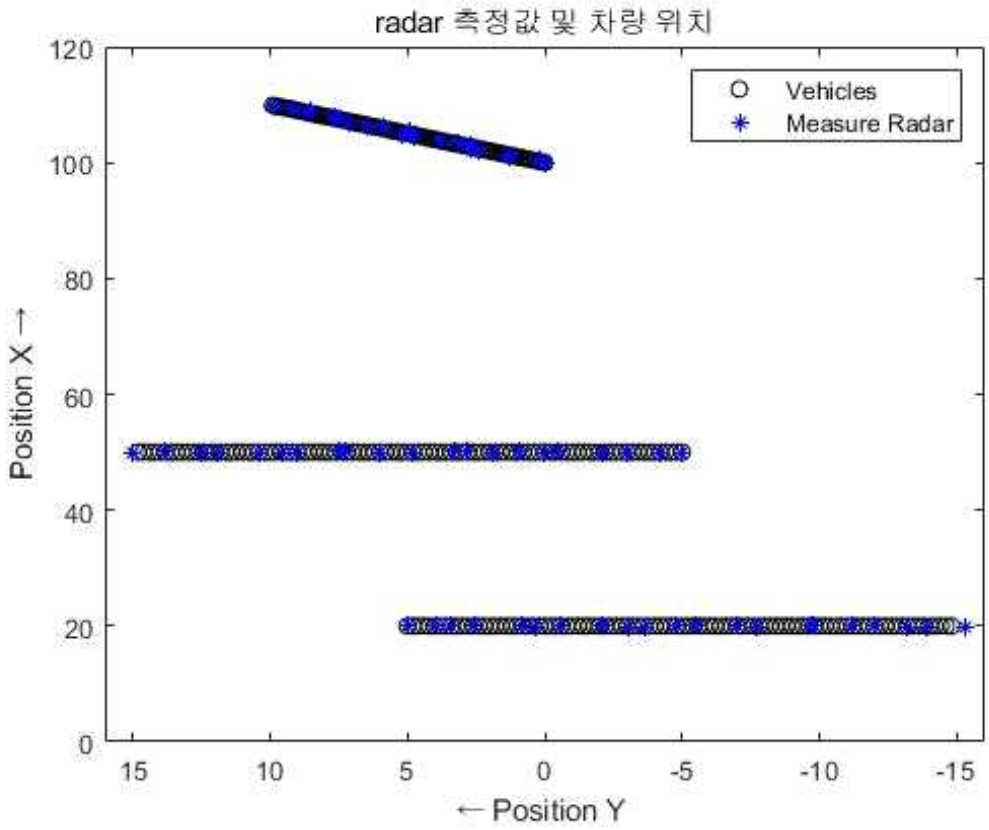
3) Sampling Time = 10[ms] = Tc
Rv(Camera) = 6, Rr(Radar) = 5
Camera와 Radar 는 각각 60ms, 50ms 마다 data가 업데이트 되며, 이를 고려하여 Measure data를 측정.
Random으로 index를 부여하여 Mahalanobis distance를 통해 Classification 수행

$$d^2_{M_q}(\mathbf{y}^{[m_q]}, \bar{\mathbf{y}}^{[l_q]}) = \left(\mathbf{y}^{[m_q]} - \bar{\mathbf{y}}^{[l_q]}\right)^T R_q^{-1} \left(\mathbf{y}^{[m_q]} - \bar{\mathbf{y}}^{[l_q]}\right)$$



4) Object의 True값과 Radar, Camera의 측정 값





위에 Radar와 Camera의 측정 값 데이터를 True값을 비교하여 보았을 때 센서의 특성을 볼 수 있었습니다. Radar는 종방향으로 정확하지만 횡방향 오차가 존재하였으며, Camera는 횡방향은 정확하지만 종방향은 오차가 있는 것을 확인하였습니다.
각 센서들의 장단점으로 인해 한가지의 센서를 사용하는 것 보다는 각각의 센서들의 장점을 살려 Sensor-Fusion을 하는 것이 좋은 결과를 낼수있다고 생각하게 되었습니다. (Camera에서는 횡방향, Radar에서는 종방향)

5) Multirate decentralized filter

$$\begin{cases} \bar{\mathbf{x}}^{[l_q]}(k_q, i+1) = \Phi_{ov}\hat{\mathbf{x}}^{[l_q]}(k_q, i) \\ \hat{\mathbf{x}}^{[l_q]}(k_q, i+1) = \bar{\mathbf{x}}^{[l_q]} + L_q\left(\mathbf{y}^{[l_q]}(k_q, 0) - C_{ov}\bar{\mathbf{x}}^{[l_q]}(k_q, 0)\right) \end{cases}$$

Step 1 예측값 계산

$$\overline{X}^{[lv]} = \Phi_{ov}\hat{X}^{[lv]}, \qquad \overline{X}^{[lr]} = \Phi_{ov}\hat{X}^{[lr]}$$

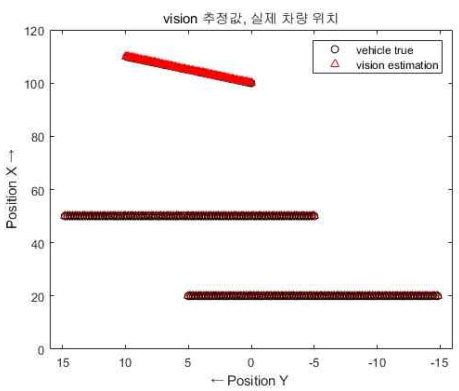
Step 2 gain
칼만 gain은 matlab place 함수 사용.

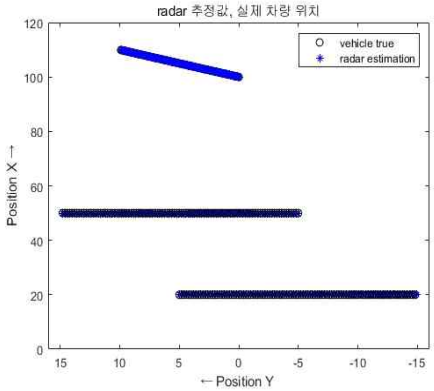
$$A = \begin{bmatrix} 1&0&1&0 \\ 0&1&0&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} B = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} L = \begin{bmatrix} 1&0 \\ 0&1 \\ 0&0 \\ 0&0 \end{bmatrix}$$

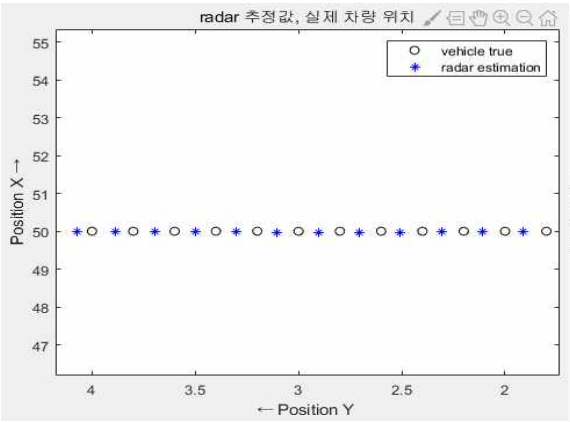
$$P_v = \left[10^{-0.02}\,10^{-0.03}\,10^{-0.04}\,10^{-0.05}\right] P_r = \left[10^{-0.03}\,10^{-0.02}\,10^{-0.04}\,10^{-0.05}\right]$$

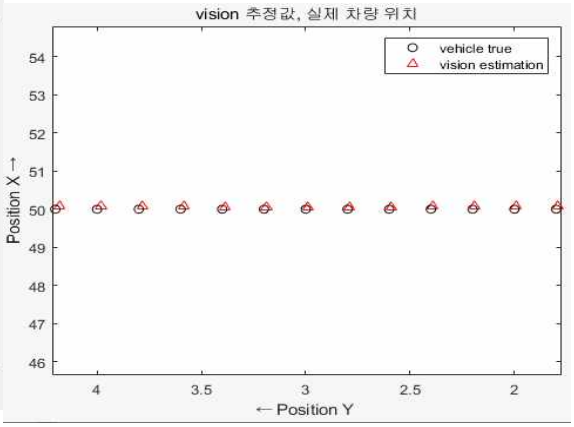
Step 3 추정값 계산

$$\widehat{X}^{[lv]} = \overline{X}^{[lv]} + L_v(y^{[lv]}(i_0) - C_{ov}\overline{X}^{[lv]}(i_0))$$
$$\widehat{X}^{[lr]} = \overline{X}^{[lr]} + L_r(y^{[lr]}(i_0) - C_{ov}\overline{X}^{[lr]}(i_0))$$









Camera와 Radar의 추정값을 비교해보면 Camera는 횡방향은 정확하지만 종방향의 정확도가 낮으며 Radar는 종방향은 정확하지만 횡방향은 정확도가 낮음을 확인.
==> 따라서 Sensor-Fusion시에 Camera에 횡방향 가중치를 높게하고 Radar에는 종방향 가중치를 높게 설정하여야 한다.

given : $\hat{X}_{fusion}$, $\hat{X}_{radar}$, $\hat{X}_{radar}$, $weigh$

$\hat{X}x_{radar}$, $\hat{X}y_{radar}$ = radar sensor 추정 값의 x,y좌표

$\hat{X}x_{vision}$, $\hat{X}y_{vision}$ = vision sensor 추정 값의 x,y좌표

$\hat{X}x_{fusion}$, $\hat{X}y_{fusion}$ = sensor fusion 결과 값 x,y좌표

$weigh_{high}$ = 0.8 , $weigh_{low}$ = 0.2

N = all samples

iteration = 0

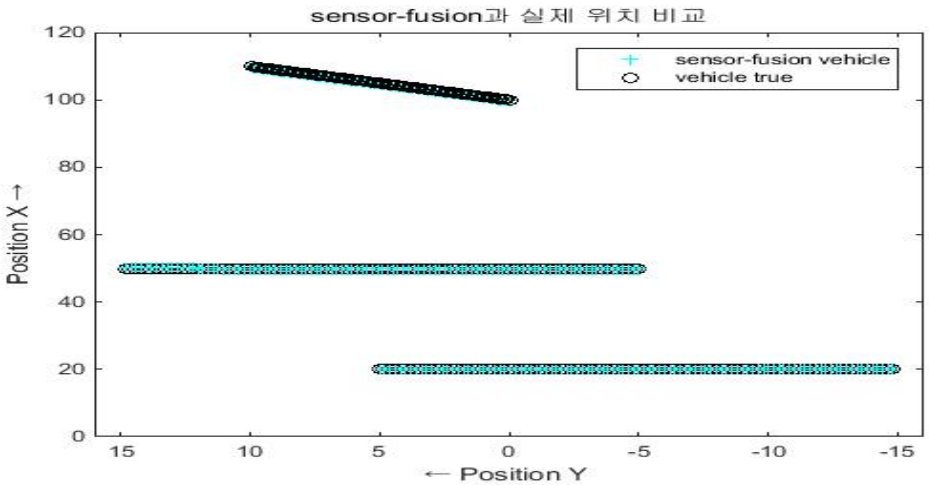
while interation < N {

for all index {

$$\hat{X}x_{fusion} = \hat{X}x_{vision} * weigh_{low} + \hat{X}x_{radar} * weigh_{high}$$

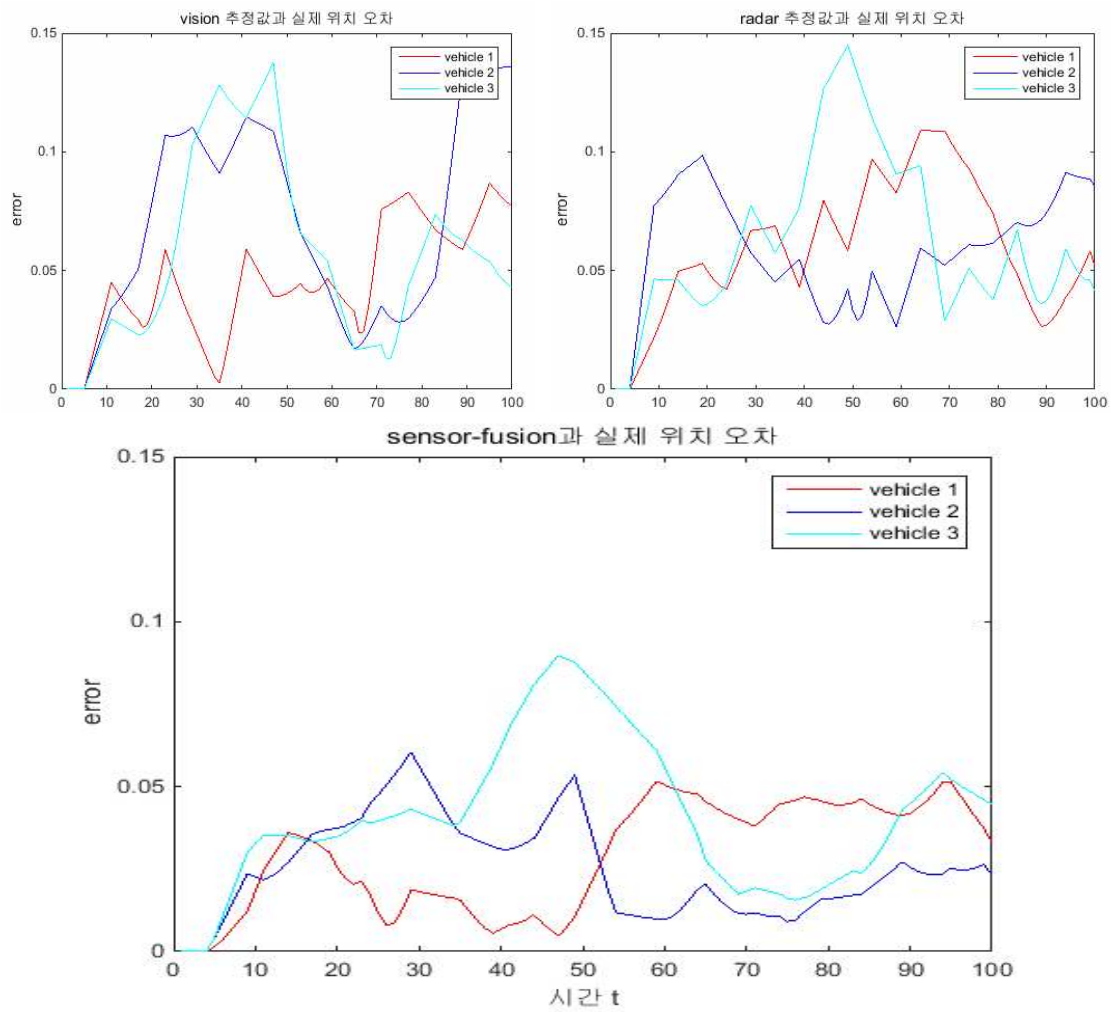
$$\hat{X}y_{fusion} = \hat{X}y_{vision} * weigh_{high} + \hat{X}y_{radar} * weigh_{low}$$

}



}

Sensor-Fusion 결과 센서를 각각 사용했던 것 보다 우수한 결과를 확인.



오차 그래프를 보았을 때 확실히 오차가 줄어듬을 확인할 수 있었습니다.