Kalman Filter를 이용한 mouse tracking

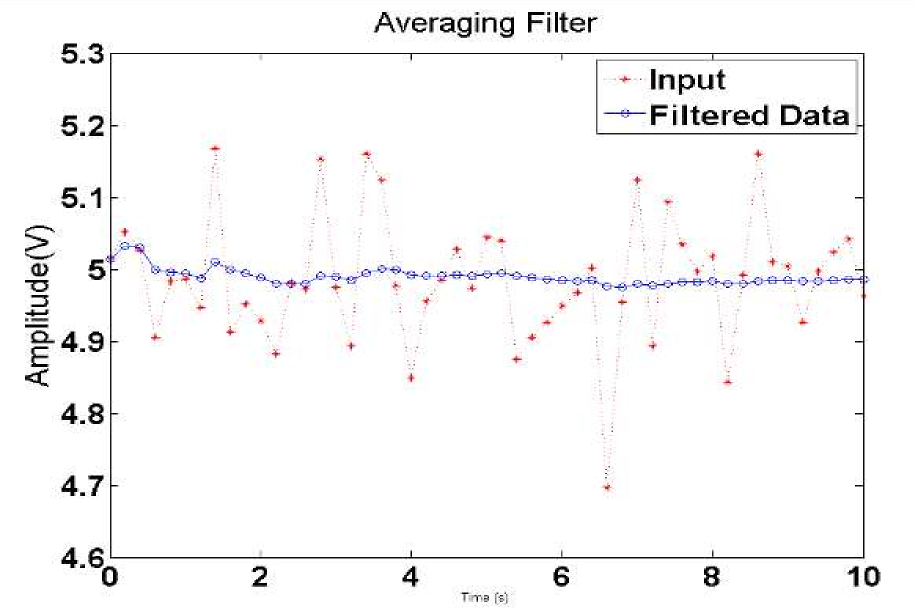
홍익대학교
전자전기공학과

목차

1 서론) average filter

- 평균 : 데이터의 총 합을 데이터의 갯수로 나눈 값
- ex) k개의 데이터 $(x_1, x_2, \cdots, x_k)$ 가 있을 때 평균 : $\overline{x_k} = \frac{x_1 + x_2 + \cdots + x_k}{k}$
- 여기에 데이터가 하나 더 추가된다면? 평균을 어떻게 계산해야 하는가?
  => 모든 데이터를 다시 더해서 K+1로 나눠야한다.
  만약 데이터가 백만개라면, 백만개의 데이터를 모두 갖고 있어야하고, 연산을 다시 해야만한다. (비효율적)
- 이는 위의 평균 식이 recursive expression이 아니기 때문이다.
  recursive expression은 이전의 결과를 재사용하기 때문에 계산 효율이 좋다.
- 평균의 recursive expression은 다음과 같다. $\overline{x_k} = \frac{k-1}{k}\overline{x_{k-1}} + \frac{1}{k}x_k$

1 서론) average filter



- 평균을 내면 측정 데이터에서 잡음을 제거 할 수 있다.

- 하지만 측정하려는 물리량이 시간에 따라 변하면 평균필터는 적절한 대안이 아니다.
  평균은 데이터의 동적인 변화는 모두 없애버리고, 과거의 모든 데이터를 망라하여 하나의 값만을 내놓기 때문이다.

1 서론) moving average filter

- moving average filter는 모든 측정 데이터가 아니라, 지정된 개수의 최근 측정값만 가지고 계산한 평균이다.

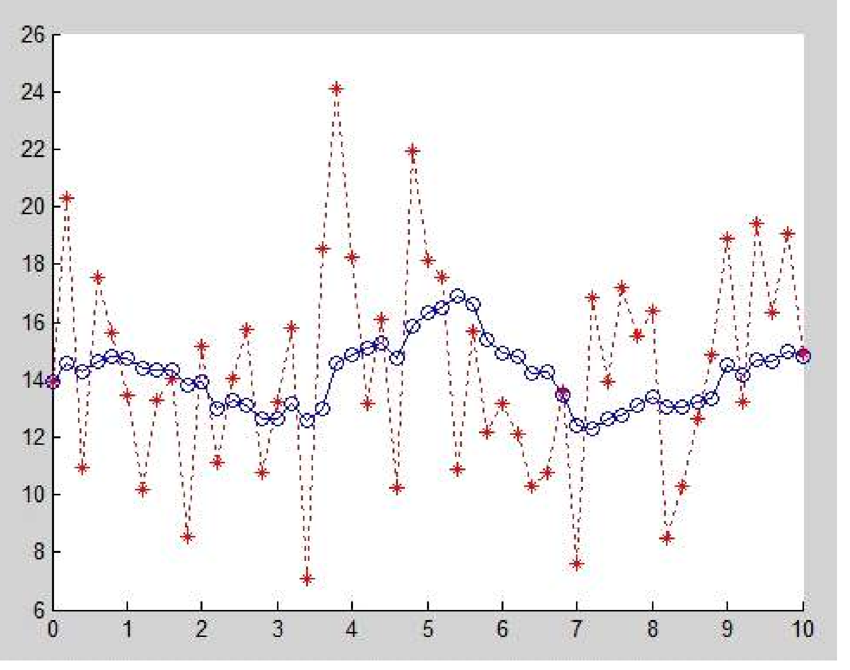
$$\overline{x_k} = \frac{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}{n}$$

- 즉, 새로운 데이터가 들어오면 가장 오래된 데이터는 버리는 방식으로, 데이터 개수를 일정히 유지시키면서 평균을 구한다.

- moving average filter의 recursive expression은 다음과 같다.

$$\overline{x_k} = \overline{x_{k-1}} + \frac{x_k - x_{k-n}}{n}$$

1 서론) moving average filter



- 평균을 내는 데이터의 개수가 많으면 잡음 제거 성능은 좋아지지만, 측정 신호의 변화는 제 때 반영되지 않고 시간지연된다.

- 반대로 평균을 내는 데이터의 개수가 적으면 측정 신호의 변화는 잘 따라가지만 잡음이 잘 제거 되지 않는다.

1 서론) exponentially weighted moving average filter

- moving average filter를 실제로 사용해 보면 잡음을 제거하면서 변화추이를 반영하는게 쉽지 않다. 이유는 다음과 같다.

$$\overline{x_k} = \frac{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}{n} \qquad \overline{x_k} = \frac{1}{n}x_{k-n+1} + \frac{1}{n}x_{k-n+2} + \cdots + \frac{1}{n}x_k$$

- 모든 데이터에 동일한 가중치 (1/n)을 부여하기 때문이다.
다시말해서 가장 최근의 데이터 (x_k)와 가장 오래된 데이터 (x_k-n+1)를 같은 비중으로 평균에 반영하는 것이다.

- EWMAF recursion expression은 다음과 같다.

$$\overline{x_k} = \alpha\overline{x_{k-1}} + (1-\alpha)x_k \qquad 0 < \alpha < 1$$

- 평균필터에서는 k의값을 임의로 지정할 수 없고, 데이터의 갯수에 따라 자동으려 결정되는 값이었다.
반면, EWMAF는 알파값이 평균과는 직접적 관련이 없고, 설계자가 직접 설정하는 값이다.

$$\overline{x_k} = \frac{k-1}{k}\overline{x_{k-1}} + \frac{1}{k}x_k$$

참고 : average filter recursion expression

1 서론) exponentially weighted moving average filter

$$\overline{x_k} = \alpha \overline{x_{k-1}} + (1-\alpha)x_k \qquad 0 < \alpha < 1 \qquad 0 < 1-\alpha < 1$$

$$\overline{x_{k-1}} = \alpha \overline{x_{k-2}} + (1-\alpha)x_{k-1} \qquad \alpha(1-\alpha) < 1-\alpha$$

$$\overline{x_{k-2}} = \alpha \overline{x_{k-3}} + (1-\alpha)x_{k-2} \qquad \alpha^2(1-\alpha) < \alpha(1-\alpha) < 1-\alpha$$

$$\overline{x_k} = \alpha^2 \overline{x_{k-2}} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k$$

$$\overline{x_k} = \alpha^3 \overline{x_{k-3}} + \alpha^2(1-\alpha)x_{k-2} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k$$

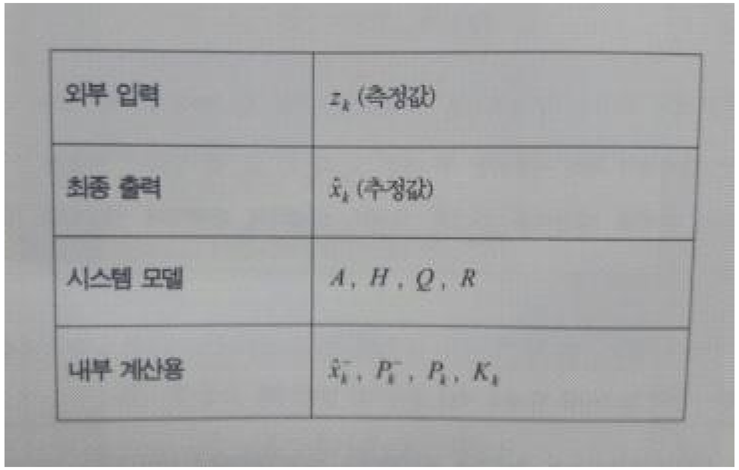
- 위의 식을 보면 이전 측정 데이터일수록 더 작은 계수가 곱해진다는 사실을 확인할 수 있다.

- 따라서 EWMAF는 잡음 제거와 변화 민감성이라는 상충된 요구를 moving average filter보다 잘 충족한다.

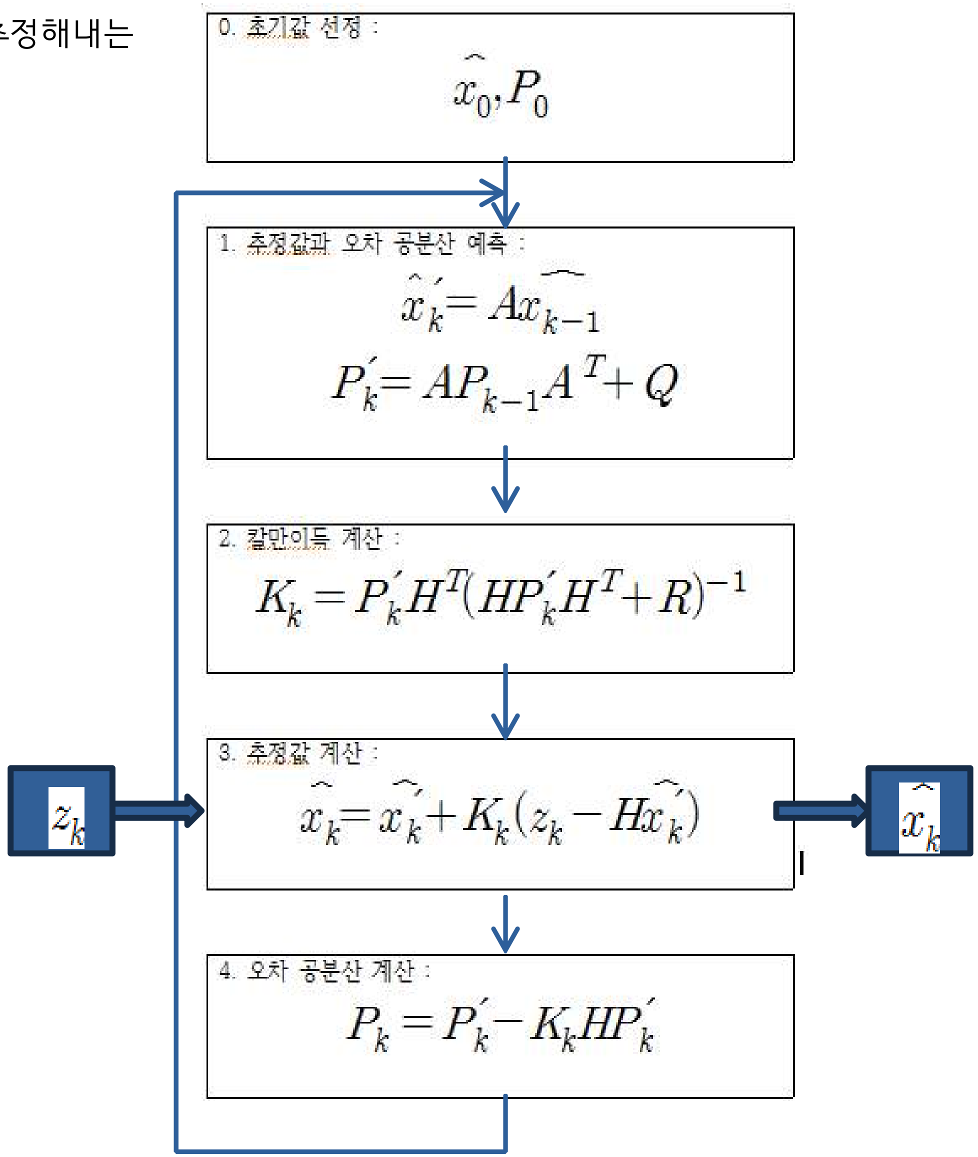
- 알파값이 작으면 추정값에 측정값이 더 많이 반영된다.
  반면, 알파값이 커지면 측정값의 비중은 작아지고, 직전 추정값의 비중이더 커진다.

2 본론) kalman filter

- 칼만필터는 잡음을 제거하고, 측정하지 않은 값을 추정해내는 필터이다.
- 칼만필터 알고리즘은 다음과 같다.
- 1. 예측과정, 2~4 추정 과정



외부 입력	z_k (측정값)
최종 출력	$\hat{x}_k$ (추정값)
시스템 모델	A, H, Q, R
내부 계산용	$\hat{x}_k^-$, P_k^-, P_k, K_k



추정과정 (1/3)

- 추정과정의 목표는 칼만 필터의 최종 결과물인 추정값을 계산해 내는 것이다.
- exponentially weighted moving average filter는 직전 추정값과 측정값에 가중치를 주고 더해서 추정값을 계산

$$\overline{x_k} = \alpha \overline{x_{k-1}} + (1-\alpha) x_k \qquad 0 < \alpha < 1$$

- 칼만필터또한 직전 예측값과 측정값에 적절한 가중치를 곱한 다음, 두 값을 더해서 최종 추정값을 계산

$$\hat{x}_k = \hat{x}'_k + K_k (z_k - H \hat{x}'_k)$$

$$\hat{x}_k = (I - K_k H) \hat{x}'_k + K_k z_k \qquad \hat{x}'_k = A \hat{x}_{k-1}$$

$$\hat{x}_k = (I - K_k H) A \hat{x}_{k-1} + K_k z_k$$

- 즉, 칼만필터는 EWMAF의 직전 추정값이 아닌 예측값을 사용한다는 점이 다르다.

추정과정 (2/3)

- EWMAF와는 구별되는, 칼만 필터만의 독특한 특징은 다음과 같다.
- EWMAF에서는 추정값 계산에 사용하는 가중치 알파가 상수였다. 그래서 매번 가중치를 새로 계산할 필요가 없었다. 또한, 이 값은 필터 설계자가 임의로 적절한 값을 선정하는 값이다.

$$\overline{x_k} = \alpha \overline{x_{k-1}} + (1-\alpha) x_k \qquad 0 < \alpha < 1$$

- 반면 칼만필터는 알고리즘을 반복하면서 K_k값을 새로이 계산한다.
즉 추정값을 계산하는 가중치를 매번 다시 조정한다는 뜻이다. 이점이 바로 칼만필터와 EWMAF와 다른 점이다.

$$\hat{x}_k = \hat{x}'_k + K_k (z_k - H \hat{x}'_k)$$

$$\hat{x}_k = (I - K_k H) A \hat{x}_{k-1} + K_k z_k$$

- 칼만이득을 계산하는 수식에 대해서는 자세히 설명하지 않겠다.

$$K_k = P'_k H^T (H P'_k H^T + R)^{-1}$$

추정과정 (3/3)

- 오차 공분산이란 칼만 필터의 추정값이 참값에서 얼마나 차이가 나는지를 나타내는 지표이다.
- 다시말해서 오차 공분산은 추정값의 정확도에 대한 척도가 된다.
- Pk가 크면 추정 오차가 크고
- Pk가 작으면 추정 오차가 작다.

$$P_k = P_k' - K_k H P_k'$$

예측과정 (1/2)

$$\hat{x}'_{k+1} = A\hat{x}_k$$

$$P'_{k+1} = AP_kA^T + Q$$

$\hat{x}_k$	상태변수[6] 추정값
$\hat{x}_k^-$	상태변수 예측값
P_k	오차 공분산 추정값
P_k^-	오차 공분산 예측값

- 예측 과정에서는 시각이 t_k에서 t_k+1로 바뀔 때, 추정값 X_k+1을 예측한다.
- 또한 X_k+1뿐만 아니라, 오차 공분산 P_k+1을 예측한다.

예측과정 (2/2)

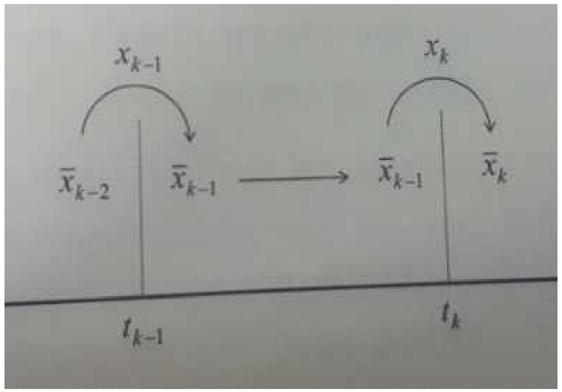
- [예측과 추정의 차이]
- EWAMF의 경우 중간에 별도의 단계를 거치지 않고, 새로운 추정값 계산에 직전 추정값을 바로 사용한다.
  다시말해서, t_k-1의 추정값 X_k-1이 다음 시각 t_k에서 그대로 사용된다.

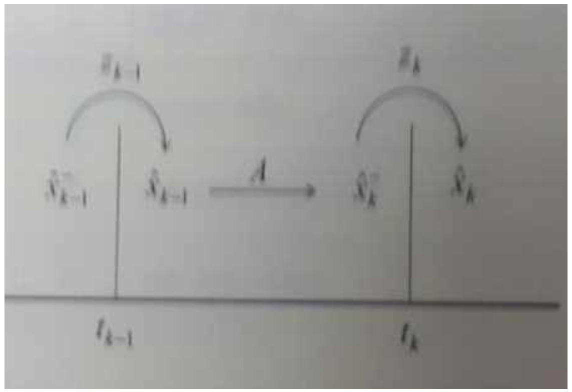
$$\overline{x_k} = \alpha \overline{x_{k-1}} + (1-\alpha) x_k$$

- 칼만필터의 경우 직전 추정값은 보이지 않고, 대신 예측값을 사용하고 있다.
  그런데 이 예측값은 직전 추정값을 이용해 구현한 값이다.
  이처럼 칼만필터는 추정값을 계산할 때 직전 추정값을 바로 쓰지 않고 예측 단계를 한 번 더 거친다.

$$\hat{x}_k = (I - K_k H)\hat{x}^-_k + K_k z_k \qquad \hat{x}^-_k = A\hat{x}_{k-1} \qquad \hat{x}_k = (I - K_k H)A\hat{x}_{k-1} + K_k z_k$$

- 이런 이유로 예측값을 priori estimate, 추정값을 posteriori estimate라고 부르기도 한다.





시스템 모델

외부 입력	z_k (측정값)
최종 출력	$\hat{x}_k$ (추정값)
시스템 모델	A, H, Q, R
내부 계산용	$\hat{x}_k^-$, P_k^-, P_k, K_k

- 칼만필터를 설계함에 있어서 어려운 부분은 시스템 모델을 유도하는 일
- 시스템 모델은 우리가 다루는 문제를 수학식으로 표현해놓은 것
- 다행히도 칼만필터가 워낙 다양한 분야에 이용되다보니 참고할 만한 시스템 모델을 쉽게 찾을 수 있다.

- 칼만필터는 다음과 같은 선형 상태 모델을 대상으로 한다.

$$x_{k+1} = Ax_k + w_k$$

$$z_k = Hx_k + v_k$$

- x_k : 상태변수, (nX1) column vector 거리, 속도, 무게 등 우리가 관심 있는 물리적 변수
- A : 상태전이행렬, (nXn) matrix 모든 성분이 상수. 시스템의 운동 방정식
- w_k : 잡음, (nX1) column vector 상태변수에 영향을 주는 잡음

- z_k : 측정값, (mX1) column vector
- H : (mXn) matrix 모든 성분이 상수. 측정값과 상태 변수의 관계를 규정
- v_k : 측정잡음, (mX1) column vector 센서에서 측정되는 잡음

잡음의 공분산

$$x_{k+1} = Ax_k + w_k$$
$$z_k = Hx_k + v_k$$

- 잡음 : 다음에 어떤 값이 나올지 예측할 수 없고 순전히 통계적인 추정만 가능한 값.
- 칼만필터는 잡음이 표준정규분포를 따른다고 가정하기 때문에,
  잡음의 분산만 알면 된다. 표준정규분포에서는 평균이 항상 0이기 때문이다.
- 이러한 이론적 기반에서 칼만 필터는 상태 모델의 잡음을 다음과 같은 공분산 행렬로 표현한다.
- **Q : w_k의 공분산 행렬, (nXn) 대각 행렬**
- **R : v_k의 공분산 행렬, (mXm) 대각 행렬**
- 공분산 행렬은 변수의 분산으로 구성된 행렬로 정의 된다.
  예를 들어 n개의 잡음 w1, w2, …, wn이 있고, 각 잡음의 분산은 $\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2$ 이라고 하자.
  그러면 공분산 행렬은 다음과 같이 쓸 수 있다.

$$\begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix}$$

- 행렬 R도 같은 방식으로 구성한다.
- Q와 R은 잡음의 특성을 정확히 해석해서 정하는게 원칙이지만,
  여러 오차가 복합적으로 적용하기 때문에 해석적으로 결정하는 데는 한계가 있다.
  이 두 행렬을 칼만 필터의 설계 인자로 보고 시행착오 과정을 통해 보정하면서 적절한 값을 찾아야 한다.

잡음의 공분산

- 공분산행렬이 칼만필터에 미치는 영향을 생각해보자.

- 먼저 칼만필터 알고리즘에 R, 과 Q행렬이 사용되는 부분을 보면 다음과 같다.

$$P'_k = AP_{k-1}A^T + Q \qquad K_k = P'_kH^T(HP'_kH^T + R)^{-1}$$

- 위의 식에서 모든 변수가 스칼라라고 가정하면 역행렬은 나누기와 같게 된다.

$$K_k = \frac{P'_kH^T}{HP'_kH^T + R}$$

- 이 식에서 R이 커지면 칼만이득은 작아진다. 그렇다면 칼만이득이 작아지면 추정값에는 어떤 영향을 줄까?

$$\hat{x}_k = (I - K_kH)\hat{x}'_k + K_kz_k$$

- 칼만이득이 작아지면 추정값 계산에 측정값이 반영되는 비율이 작아진다. 반면에 예측값의 반영비율은 높아진다. 따라서 측정값의 영향을 덜 받고 변화가 완만한 추정값을 얻고 싶다면 행렬 R을 키우면 된다.

- 행렬 Q가 커지면 오차 공분산 예측값도 커진다. 오차 공분산 예측값이 커지면 어떻게 될까?

- 오차공분산 행렬이 커지면 칼만이득이 커진다. 즉, 칼만이득이 커지면 측정값의 반영 비율이 더 높아진다. 따라서 측정값의 영향을 덜받고 변화가 완만한 추정값을 얻고 싶다면 행렬 Q를 줄여야 한다. (행렬 R과는 반대)

3) 칼만필터 응용

$$x_{k+1} = Ax_k + w_k$$

$$z_k = Hx_k + v_k$$

$$x = \begin{pmatrix} 위치 \\ 속도 \end{pmatrix}$$

[위치정보만 가지고 측정하지도 않은 속도를 알아내기]

$$\begin{pmatrix} 위치_{k+1} \\ 속도_{k+1} \end{pmatrix} = \begin{pmatrix} 위치_k + 속도_k{}^{*} \Delta t \\ 속도_k + w_k \end{pmatrix}$$

$$\begin{pmatrix} 위치 \\ 속도 \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 위치 \\ 속도 \end{pmatrix}_k + \begin{pmatrix} 0 \\ w_k \end{pmatrix} \qquad A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

- 첫번째 식은 현재위치 = 직전위치 + 이동거리 라는 물리 법칙을 수식으로 표현한것.
  시스템 잡음이 관계식에 포함되지 않은 것은 이 때문.

- 두번째 식이 의미하는 바는 속도의 변화는 시스템 잡음의 영향만 받을 뿐다른 외부의 힘은 작용하지 않는다는 뜻.

$$z_k = 위치_k \qquad z_k = [1\ 0] \begin{pmatrix} 위치 \\ 속도 \end{pmatrix}_k + v_k \qquad H = (1\ \ 0)$$

- 이식이 의미하는 바는 측정하는 값은 위치이고, (속도는 측정하지 않는다.) 측정값에는 잡음이 섞여 있다는 말이다.

- 이처럼 행렬 A, H는 임의로 선정하는 것이아닌, 시스템의 물리적인 관계를 모델링한 결과이다.

3) 칼만필터 응용

- 마지막으로 잡음의 공분산행렬 Q, R만 결정하면 시스템모델의 유도는 끝난다.
- 측정잡음 (v_k)의 경우 센서 제작사에서 제공하는 경우가 많다. 그렇지 않다면 보통 실험을 통해서 결정한다.
- 시스템 잡음 (w_k)의 모델링은 어렵다.시스템에 대한 지식과 경험에 의존할 수 밖에 없기 때문이다.
- 만약 두 공분산 행렬을 해석적으로 구하기 어렵다면 이 행렬을 칼만필터의 설계 인자로 보고 시행착오를 거쳐 선정해야 한다.
- 칼만필터가 위치 정보만을 가지고 속도를 추정해낼 수 있느 이유는 무엇일까?
  시스템의 수학적 모델을 알고 있기 때문이다.
  즉 시스템이 어떤 법칙에 따라 측정값을 출력하는지를 정확히 알고 있다는 전제를 하기 때문이다.

영상속의 물체 추적

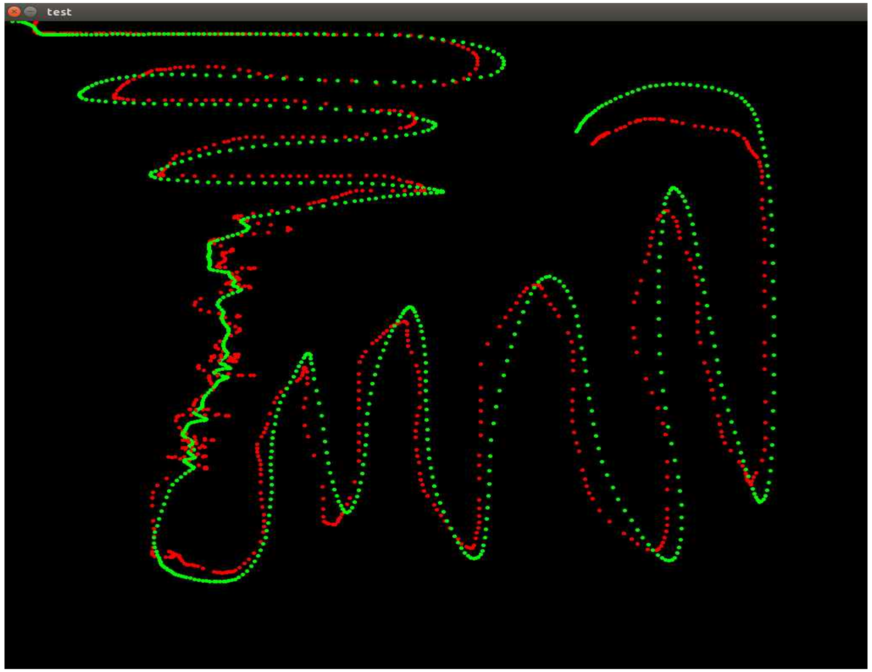
- 측정값 z_k를 구하는 방법은 영상처리 알고리즘으로 알아내야 한다.
- 칼만필터는 영상에서 물체의 위치를 찾아내는 능력이 없다.
  칼만필터는 영상처리 기법으로 찾아낸 물체의 위치를 입력 받아 정확한 위치를 추정하는 역할뿐이다.
- 앞 슬라이드에서 소개한위치-속도 모델을 2차원으로 확장하면 된다.

$$x = \begin{pmatrix} \text{위치}_x \\ \text{속도}_x \\ \text{위치}_y \\ \text{속도}_y \end{pmatrix} \qquad \begin{pmatrix} \text{위치}_{k+1} \\ \text{속도}_{k+1} \end{pmatrix} = \begin{pmatrix} \text{위치}_k + \text{속도}_k{}^* \Delta t \\ \text{속도}_k + w_k \end{pmatrix}$$

$$x_{k+1} = Ax_k + w_k \qquad A = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$z_k = Hx_k + v_k$$

- sensor noise가 큰경우 sensor noise가 작은 경우



- 실제로는 mouse pointer의 위치를 정확히 구할 수 있으므로 센서 noise는 전혀 없다고 볼 수 있습니다.

자동차 디텍션에 적용

- 마우스 포인터의 위치결과값을 가져오는 알고리즘을 영상인식 알고리즘 (자동차 디텍션) 으로 대체합니다.
- 이 경우 마우스 포인터의 P(x,y)는 자동차의 center position, P(x,y)로 적용 합니다.
- 다만 긍정적인 부분은, 칼만필터의 결과값에 의해 frame 전체를 다 찾을 필요는 없고, 칼만필터의 결과가 알려주는 곳의 주변만 찾으면 될 것 입니다.
- 또한, 칼만필터의 결과값 부근에는 분명히 자동차가 있을 것이므로, 간단한 알고리즘으로 자동차를 쉽게 찾을 것으로 기대합니다.
- 다만 어떠한 간단한 알고리즘을 찾아 낼것인가 생각해보아야할 문제점 입니다.
- 그리고 트랙킹 모드로 넘어가기전에는 detection 알고리즘을 먼저 수행해야 할것입니다.

출처

- 칼만필터의 이해 _김성필 저 page 3 ~ 117