칼만필터 대 파티클 필터: K-NN 옥내 측위의 경우

임재걸*, 도재수*, 김진석**
*동국대학교 컴퓨터공학과
**동국대학교 정보통계학과
e-mail : {yim, dojesu}@dongguk.ac.kr, jinseog.kim@gmail.com

Kalman Filter vs. Particle Filter: in the case of K-NN Indoor Positioning

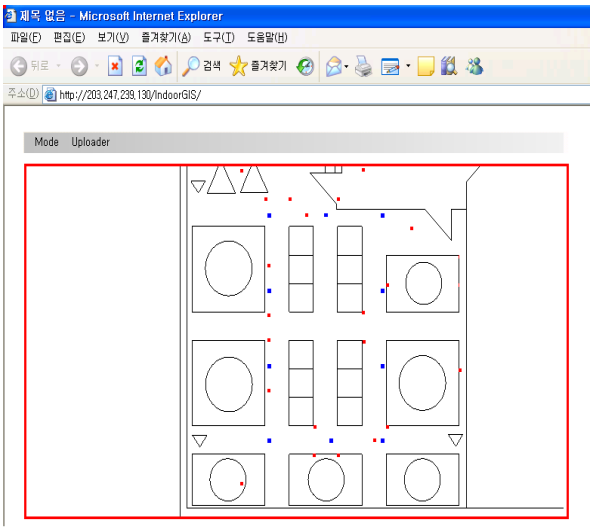
Jaegeol Yim*, Jaesu Do*, Jinseog Kim**
*Dept. of Computer Science, Dongguk University
**Dept. of Statistics and Information Science, Dongguk University

요 약

다양한 옥내 측위 방법이 연구 발표되었다. 그 중에서 무선근거리통신망을 이용하는 방법은 측위를 위한 별도의 특수 장비를 요구하지 않기 때문에 실용적이다. 무선근거리통신망 기반 옥내 측위에서는 지문방식과 신호세기를 거리로 환산하여 사용하는 방법이 가장 흔히 사용되는 두 가지 방법이다. 지문방식은 시간과 노력이 많이 소요되는 준비단계가 필요하지만 신호세기를 거리로 환산하여 사용하는 방법보다 정확한 반면, 신호세기를 거리로 환산하여 사용하는 방법은 구현하기가 용이하지만 오차가 심하다. 때때로 지문방식 조차도 실제 응용에 적용되기에 부적절할 만큼 오차가 커, [1]은 일종의 지문방식인 K-NN (K nearest neighbors) 방법의 오차를 개선하기 위한 파티클 필터를 소개하였다. 칼만 필터도 역시 K-NN 옥내 측위의 정확도 개선을 위하여 사용된 바 있다. 본 논문은 K-NN 방법의 정확도 개선에 있어 칼만 필터와 파티클 필터의 성능을 비교하는 실험 결과를 소개한다.

1. 서론

<그림 1>에 보이는 통로를 걸어가면서 K-NN (K Nearest Neighbors) 옥내 측위 방법을 수행하였다. 이 통로를 4 바퀴 돌면서 실험한 결과의 일부가 <표 1>에 보인다. 측정치의 단위는 mm 이며, K-NN 방법의 평균 오차는 6.42 미터로 나타났다.



<Figure 1> Our test bed and a typical test result of K-NN indoor positioning

<Table 1> A part of the K-NN results

	K-NN 1				
	True X	True Y	Measured X	Measured Y	Error
Point 1	79626.1	17657.8	85175.4	12225.4	7765.6
Point 2	80414.9	17657.8	77187.9	7444.0	10711.4
Point 3	81203.8	17657.8	82190.5	10207.9	7515.0
	…		…		
Point 29	78069.1	17657.8	76187.3	19217.9	2444.4
Point 30	78847.6	17657.8	79172.2	18222.9	651.7
				Average Error:	6240.0
				Standard Deviation:	2732.1

칼만 필터는 동적 시스템의 상태를 예측하고 이를 측정치로 수정하는 작업을 반복함으로써 동적 시스템의 상태를 개산한다. 칼만 필터의 성질들 중에는 오차평균을 최소화한다는 장점이 있기 때문에, 칼만 필터를 사용하는 논문이 수백 개 이상 출판된 바 있는데 대부분은 자동 혹은 반자동 항법에 칼만 필터를

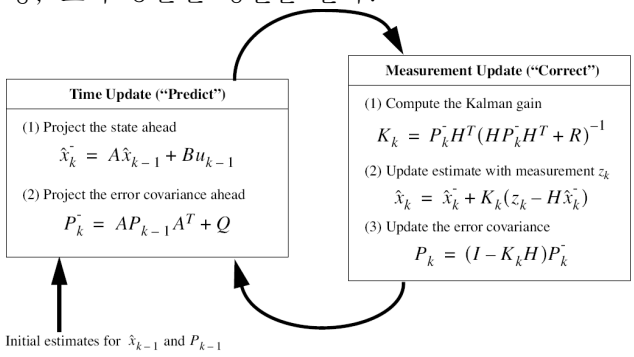
응용하는 것이다 [2-5]. 옥내 측위나 항법에 칼만 필터를 적용한 사례는 비교적 적지만, [6]과 [7]이 옥내 측위에 칼만 필터를 적용한 사례를 보인다[8].

파티클 필터는 동적 시스템의 상태를 확률밀도함수로 모델하고, 칼만 필터처럼 예측과 교정을 반복한다. 그러나 파티클 필터는 칼만 필터와 달리 상태 자체를 예측 및 교정하는 게 아니라 상태의 확률밀도함수를 예측 및 교정한다. [1]은 RSSI 의 불안정성을 억제하여 K-NN 방법의 오차를 줄이는 파티클 방법을 소개하고, 나아가서 위치의 제약 조건을 고려하여 파티클 필터의 성능을 더욱 개선하는 방법이 소개되었다.

본 논문은 칼만 필터와 파티클 필터를 모두 구현하여 두 가지 필터의 성능을 비교하는 실험을 수행한다.

2. 칼만 필터

칼만 필터 처리 과정은 <그림 2>와 같이 요약된다. 이 그림에서 이동객체의 상태는 벡터 x 로 표현되며, <그림 3>에 보이는 바와 같이 x, y 좌표와 속도로 구성된다. 행렬 A 는 현재 상태로 다음 상태의 예측치를 구하기 위한 행렬임으로 <그림 3>에 보이는 바와 같은 값을 가져야 하며, Δt 는 현재 상태에서 다음 상태 간에 흘러간 시간을 나타낸다. 벡터 z 는 측정치를 나타내며 이 실험에서는 x, y 좌표로 구성된다. 이 실험에서, 행렬 H 는 <그림 3>에 보이는 바와 같은 값으로 구성되어 상태 벡터 x 에서 x, y 좌표만 추출할 수 있어야 한다. 칼만 필터 처리 과정은 예측과 측정치를 이용한 교정을 반복한다. 예측 단계에서는 다음 상태와 오차 공분산을 예측하고, 교정 단계에서는 Kalman gain 구하기, 측정치를 반영한 예측치 교정, 오차 공분산 갱신을 한다.



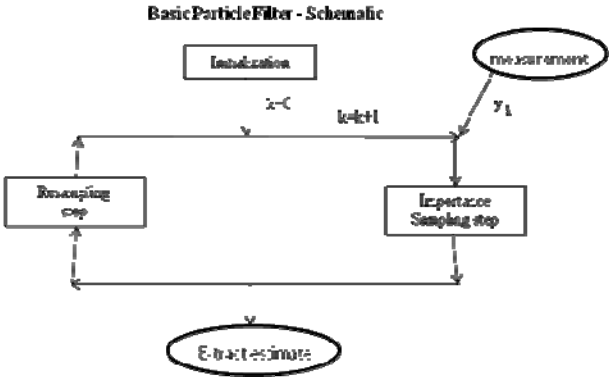
<Figure 2> The Kalman filter process

$$\hat{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{xk} \\ v_{yk} \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

<Figure 3> The matrices used in the Kalman filter

3. 파티클 필터

파티클 필터도 역시 반복 처리한다. 즉, <그림 4>에 보이는 바와 같이, 초기화한 후에 importance sampling 단계와 resampling 단계를 반복한다 [9]. 초기화에서는 동적 시스템의 초기 상태를 나타내는 확률밀도함수를 따르는 초기 파티클을 (샘플) 생성한다. Importance sampling 단계에서는 그 당시의 측정치를 이용하여 파티클에 가중치를 부여하고, 파티클들의 가중치 합으로 그 당시의 상태를 결정한다. Resampling 단계에서는 차기 파티클을 생성한다.



<Figure 4> A schematic diagram of the particle filter process.

4. 칼만 필터 구현

마이크로소프트 비주얼 스튜디오의 C#으로 <표 2>와 같이 칼만 필터 처리과정을 구현하였다. 이 구현에서 행렬 Q, R, P 그리고 X 를 각각 <표 3>처럼 초기화하였다. 행렬 R 과 Q 는 각각 측정치의 공분산과 시스템 모델의 공분산을 나타내는데 이 값들을 조절하여 칼만 필터의 성능을 최적화 할 수 있다. 본 실험에서는, R 값과 Q 값을 다양하게 바꾸어 가면서 구현한 칼만 필터 처리과정을 <표 1>에 보이는 K-NN 결과에 적용하여 본 결과 <표 3>에 보이는 값이 적당함을 알았다. 본 실험에서 벡터 X 는 처음 측정치로 하였고, 행렬 P 는 초기 X 의 공분산이 크다는 가정하에 큰 값으로 초기화하였다.

<Table 2> Our Kalman Filter Process

```
1. initialize
For ( i = 2; not EOF; i++) {
   2. Step (1) of Time Update
   3. Step (2) of Time Update
   4. Step (1) of Measurement Update
   5. Step (2) of Measurement Update. Use the i-th
measured location as z. Print x̂_k .
   6. Step (3) of Measurement Update
}
```

<Table 3> Initial values of our Kalman filter process

$$Q=\begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}, \quad R=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$P_0=\begin{bmatrix} 300 & 0 & 0 & 0 \\ 0 & 300 & 0 & 0 \\ 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 300 \end{bmatrix}, \quad \hat{x}_k=\begin{bmatrix} 85175.4 \\ 12225.5 \\ 0 \\ 0 \end{bmatrix}$$

실험 결과가 <표 4>에 요약되었다. <표 1>로부터 실험 데이터인 측정치의 공분산은 7025691.4 임을 알 수 있으나 시스템 모델의 공분산은 구하기가 어렵다. 많은 실험 끝에, 칼만 필터의 성능은 R 값과 Q 값의 비율에 의하여 결정됨을 알았다. 즉, 정확한 측정치 공분산과 시스템 모델의 공분산을 구하기 어려울 때에는 R 을 1 로 고정시키고 Q 값을 변경하면서 성능을 최적화하는 Q 값을 찾을 수 있다. 본 실험에서는 R=1, Q=0.0001 일 때 가장 좋은 결과를 보였다. 실험 데이터는 <그림 1>에 보이는 통로를 4 바퀴 돌면서 측정한 측정치다. <표 4>의 첫 행은 "For all"이라는 레이블이 의미하는 바와 같이 4 바퀴 전체의 평균 오차를 나타내며, "first lap"이라는 레이블이 붙은 두 번째 행은 첫 번 바퀴만의 평균오차를 나타낸다. 칼만 필터 실험을 요약하면, 칼만 필터로 K-NN 결과를 6421/3323 = 1.93 배 개선할 수 있다. 즉, 칼만 필터를 적용하여 오차를 반으로 줄일 수 있다. 실험 데이터에서 3323 은 3.323 미터와 같다.

5. 파티클 필터 구현

마이크로소프트 비주얼 스튜디오 C#으로 <표 5>와 같이 파티클 필터를 구현하였다. 이동객체가 X 축을 따라 초속 788㎜ 로 이동한다는 가정하에 θ와 V를 각각 0 과 788 로 초기화하였다. <표 1>에 보인 측정치의 범위는 다음과 같다:
77290 <= X <= 81993 and
9748 <= Y <= 17657.
그래서 초기 파티클의 범위를 다음과 같이 정했다:
74800 <= X <= 84490 and
7500 <= Y <= 19850.
[1]에 의하면 파티클의 수는 400 이 적당하다고 한다. 그래서 파티클 간의 거리는 X(Y)축을 따라 각각 510 (650)으로 정했다. 이동객체는 어느 방향으로도 향할 수 있다는 가정하에 그리고 속도도 매우 유동적이라

<Table 5> Our particle filter process

1. Initialization. Generate initial particles (1 particle/m^2). For each particle, let 1 be its weight.
$$\theta = 0 \text{ (in rad)}, \text{V} = 788 \text{ (mm/s)}$$
2. for (i = 1; not EOF; i++) {
2.1 z_i = the i-th measured location;
2.2 for each particle x_i, compute its weight w_i as follows:

$$w_i = w_i \cdot p[z_i \mid x_i], \textit{where } p[z_i \mid x_i] = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(z_i - x_i)^2}{2\cdot\sigma^2}\right],$$

where $\sigma = 20{,}000$.
2.3 TotalOfWeight = $\sum_{j=1}^{n} w_j$, where n is the number of particles
2.4 Normalize the weights so that the total of them is 1.
2.5 Print the weighted sum, (X, Y, θ, V), of all the particles
2.6 Particle = Propagation(X, Y, θ, V)
}

Propagation(X, Y, θ, V, TotalOfWeight) {
if (TotalOfWeight < 0.000001) return(1 particle/m^2. For each particle, let 1 be its weight)
else { // generate numberOfParticles (400, for example) particles
for (i=0; I < numberOfParticles; i++) {
Tempθ = θ +RandomNormal(0, 1.5π);
TempV = V+RanddomNormal(0.0, 400);
Particle[i].X = X + TempV*cos(Tempθ);
Particle[i].Y = Y + TempV*sin(Tempθ);
Particle[i]. θ = Tempθ;
Particle[i].V = TempV;
} // end of for
return(Particle);
} // end of else
}

는 가정하에 방향각과 속도를 구하는 식에 각각 "RandomNormal(0, 1.5 π);"과 "RandomNormal(788, 400);" 을 사용한다. 단, RandomNormal 은 Normal(mean, standard deviation) 분포의 난수를 생성하는 함수다. 이제 남은 것은 2.2 행에 보이는 가중치 함수다. 가중치로 파티클과 측정치 간의 거리의 역수를 사용한다.

<Table 4> A summary of our Kalman filter test results

Parameter / Mean	R=1 Q=0.1	R=1 Q=0.01	R=1 Q=0.001	R=1 Q=0.0001	R=1 Q=0.000055	R=1 Q=0.00001	R=7025691.4 Q=700	R=7025691.4 Q=1.0
For all	4975	4233	3459	3323	3355	3435	3376	4071
first lap	4855	4304	4207	4160	4156	4152	4028	5883

실제 사용한 가중치 함수로 다음을 들 수 있다:
particles[j].weight = 1 / (Math.Sqrt((x - particles[j].X) * (x - particles[j].X) + (y - particles[j].Y) * (y - particles[j].Y)));.
처음 실험으로 위에 보이는 가중치 함수를 다음에 보이는 두 가지와 비교하였다:
1/(distance ** 2) and 1/(distance ** 4).
이 실험의 결과는 <표 6>과 같이 1/(distance ** 2)을 가중치 함수로 사용할 때가 가장 좋았다. K-NN 의 오차 평균이 6240 임으로 1/(distance ** 2)을 가중치 함수로 사용할 때 약 14% 정도 정확도가 개선된다. 그러나 칼만 필터의 오차 평균이 3323 임으로 이 파티클 필터는 더욱 개선될 여지가 많이 있음을 알 수 있다.

<Table 6> Test results for simple weight functions

Weight / error	K-NN	1/distance	1/(distance**2)	1/(distance**4)
Average error	6240	5854	5534	12415

그래서 <표 5>에 보이는 가중치 함수를 다양한 값을 σ 와 RandomNormal 의 표준편차로 대입하면서 사용하였다. 그 결과 <표 5>에 보이는 값이 적당함을 알았고, 이러한 실험 결과가 <표 7>에 보인다.

<Table 7> Test results for the weight function shown in <Table 5>

Weight / error	K-NN	σ = 12,000	σ = 20,000	σ = 30,000
Average error	6240	3,716	3,648	3,703

[1]에는 벽 넘어와 같은 불가능한 곳에 위치한 파티클의 가중치를 0 으로 하는 전략이 소개되었다. 이 전략을 적용하여 <표 8>과 같은 결과를 얻었다.

<Table 8> Test results for location-constrained weight function

Weight / error	K-NN	Kalman filter	Particle filter σ = 20,000	Location-constrained weight
Average error	6240	3,323	3,648	3,572.4

[1]은 파티클 생성시에도 불가능한 곳에 위치한 파티클을 제거하는 전략을 제안하였다. 타당한 공간의 폭을 다양하게 변하면서 이 전략을 적용하는 실험을 하였는데, <표 9>에 보이는 바와 같이, 이 전략은 정확도 개선에 전혀 기여하는 바가 없었다.

<Table 9> Test results for location-constrained particle generation

Weight / error	K-NN	Kalman filter	Particle filter	Size= 3m	Size= 2m	Size= 0.6m
Average error	6240	3,323	3,648	3,980	4,000	4,052

6. 결론

K-NN 옥내 측위 방법의 정확도를 개선화는 방법으로 [1]과 [8]은 각각 파티클 필터와 칼만 필터를 제안하였다. 본 논문은 실험을 통하여 이들 두 가지 방법의 성능이 거의 비슷하다는 것을 보였다. [1]은 또한 이동객체가 위치하기에 불가능한 곳이라는 정보를 이용하여 파티클 필터의 정확도를 두 배로 개선할 수 있다고 하였다. 그러나 본 실험에서는 이러한 정보를 이용하여도 성능 개선에 기여하는 바가 거의 없음을 보였다. 더구나 타당한 영역의 폭을 줄이면 파티클 생성에 소요되는 시간이 현저하게 늘어나고 정확도도 오히려 떨어지는 실험 결과를 보였다. 이는 파티클 필터의 성능이 실험 데이터의 성질에 따라 크게 달라짐을 의미한다. 향후 연구로 확장칼만필터를 파티클 필터와 비교하려고 한다.

References

[1] Chih-Hao Chao; Chun-Yuan Chu; An-Yeu Wu, "Location-Constrained Particle Filter human positioning and tracking system," Proceedings of IEEE Workshop on Signal Processing System, 2008, pp. 73-76

[2] T. Teo, J Chai, W. Yao, Design of a positioning system for AGV navigation, Proc. of the 7th International Conference on Control, Automation, Robotics and Vision, 2002 (ICARCV 2002) 2 (2-5 Dec. 2002) 637-642.

[3] Z. Peroutka, Design considerations for sensorless control of PMSM drive based on extended Kalman filter, Proc. of 2005 European Conference on Power Electronics and Applications (11-14 Sept. 2005) 1-10.

[4] M. Boussak, Implementation and experimental investigation of sensorless speed control with initial rotor position estimation for interior permanent magnet synchronous motor drive, IEEE Transactions on Power Electronics 20 (6) (2005) 1413–1422.

[5] S. Bolognani, L. Tubiana, M. Zigliotto, Extended Kalman filter tuning in sensorless PMSM drives, IEEE Transactions on Industry Applications 39 (6) 2003 1741–1747.

[6] A. Kotanen, M. Hannikainen, H. Leppakoski, T.D. Hamalainen, Experiments on local positioning with bluetooth, Proceedings of International Conference on Information Technology: Coding and Computing [Computers and Communications] (ITCC 2003) (April 2003) 297-303.

[7] H. Qasem, L. Reindl, Unscented and Extended Kalman Estimators for non Linear Indoor Tracking Using Distance Measurements, Proc. of the 4th Workshop on Positioning, Navigation and Communication, WPNC '07 (March 2007) 177-181

[8] Jaegeol Yim, Chansik Park, Jaehun Joo, Seunghwan Jeong, "Extended Kalman Filter for Wireless LAN Based Indoor Positioning," Decision Support Systems 45, Nov. 2008, pp. 960-971

[9] Keith Copsey, "Tutorial on Particle filters," Pattern and Information Processing Group DERA Malvern, K.Copsey@signal.dera.gov.uk.