Journal of Korea Multimedia Society Vol. 12, No. 2, February 2009(pp. 300-312)

지도 정보를 반영한 옥내 측위 보정 방안

임재걸[†], 심규박[††], 박찬식[†††], 정승환[††††]

요 약

옥내 위치 기반 서비스를 실현하려면 옥내 측위 문제가 해결되어야 한다. 경제적으로 구현이 용이한 옥내 측위 시스템은 비교적 오차가 크다. 본 논문은 지도 정보를 반영한 옥내 측위 시스템용 보정 방법을 제안한다. 제안하는 방법은 지도 정보를 이용하여 구한 적당한 인수 값을 사용하는 칼만 필터를 이용하여 이동 객체의 궤적을 구하고, 프레쉐 거리를 이용한 지도 정합을 수행한 다음, 실시간 보정 방법을 적용한다. 제안하는 방법의 효율성을 보이는 실험 결과도 제공한다.

Location Correction Based on Map Information for Indoor Positioning Systems

Jaegeol Yim[†], Kyu-Bark Shim[††], Chansik Park[†††], Seunghwan Jeong[††††]

ABSTRACT

An indoor location-based service cannot be realized unless the indoor positioning problem is solved. However, the cost-effective indoor positioning systems are suffering from their inaccurateness. This paper proposes a map information-based correction method for the indoor positioning systems. Using our Kalman filter with map information-based appropriate parameter values, our method estimates the track of the moving object, then it performs the Frechet Distance-based map matching on the obtained track. After that it applies our real time correction method. In order to verify efficiency of our method, we also provide our test results.

Key words: Indoor Positioning(옥내측위), Trajectory(궤적), Map Matching(지도 정합), Kalman Filter (칼만 필터), Real-Time Correction(실시간보정)

1. 서 론

위치 기반 서비스(LBS: Location Based Service)를 개발하려면 필수적으로 사용자의 위치를 파악해야 한다. 사용자의 위치를 파악하는 측위 방법은 옥외용과 옥내용으로 구분되며, 옥외용은 범지구 위치 결정 시스템(GPS: Global Position -ing System) 방법[1]이 일반적으로 사용되고 있다. 옥내용으로 크리켓[2], 액티브 뱃지[3], 무선 근거리 통신망에서 수신 신호의 세기를 이용하는 방법[4-7]과 신호의 도착 시각(TOA: Time of Arrival)을 이용하는 방법[8,9]등 다양한 시스템이 소개된 바 있다. 이들 중, 무선 근거리 통신망 환경의 옥내 측위는 측위를 위한 특수 장치를 설비할 필요가 없기 때문에 매력적이다.

※ 교신저자(Corresponding Author) : 심규박, 주소 : 경북 경주시 석장동 707(780-714), 전화 : 054)770-2245, FAX : 054)770-2520, E-mail : gpshim@dongguk.ac.kr
접수일 : 2008년 9월 12일, 완료일 : 2009년 2월 9일
[†] 종신회원, 동국대학교 과학기술대학 컴퓨터멀티미디어학과 교수
(E-mail : yim@dongguk.ac.kr)
[††] 정회원, 동국대학교 과학기술대학 정보통계학과 교수
[†††] 정회원, 충북대학교 전기전자컴퓨터공학부 교수
(E-mail : chansp@chungbuk.ac.kr)
[††††] 준회원, 동국대학교 전자계산학과 석사과정
(E-mail : honourj@dongguk.ac.kr)
※ 이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-314-D00362)

범지구 위치 결정 시스템은 미국이 제공하는 인공위성 기반 측위 데이터 제공 시스템이다. 측위 위성은 항상 지구를 향하여 전파를 발사하고 있으며, 범지구 위치 결정 시스템 수신 장치는 이 전파를 참조하여 위성에서 송신한 신호가 수신 장치에 도달하는데 걸리는 시간을 계산하고, 이 시간에 광속을 곱하여 수신 장치까지의 거리를 계산한다. 지구상의 어떠한 수신 장치라도 4 개의 위성으로부터 동시에 신호를 받을 수 있도록 미국 국방성은 24개의 측위 위성을 항상 작동시킨다. 위성의 위치와 수신 장치까지의 거리를 바탕으로 삼각 측량법을 적용하여, 수신 장치는 자신의 위치를 계산한다. 그러나 범지구 위치 결정 시스템 신호는 옥내에서 수신이 불가능하여 옥내 위치 기반 서비스에는 사용될 수 없다.

크리켓[2] 시스템에서는 고정 지점에 부착된 발신기(beacon)들이 고주파 신호와 초음파 신호들을 동시에 송신하면 이동 물체에 부착된 수신기가 이 신호들을 수신한다. 그리고 신호들의 도착 시각의 차이를 바탕으로 발신기 각각에 대한 거리들을 구한다. 그리고 발신기들의 좌표와 거리에 삼각 측량법을 적용하여 자신의 좌표를 구한다.

액티브 뱃지[3] 시스템에서는 이동 물체들에 적외선을 발사하는 송신기가 부착되어 있고, 방마다 여러 개의 수신기가 고정 지점에 설치되어 있다. 적외선은 벽을 통과하지 못하기 때문에 이동 물체가 어느 방에 들어오면 그 방에 설치된 고정 수신기들만 그 이동 물체의 적외선을 감지할 수 있다. 수신기는 중앙 컴퓨터에 연결되어 있어서 수신기에 감지된 적외선을 바탕으로 이동 물체가 어느 방 안에 있다는 것을 판정한다.

위에서 살펴본 옥내 측위 방법들은 측위를 위한 특수 장비와 환경을 갖추어야 한다. 특수 장비를 갖추려면 상응하는 비용을 투자해야 할 뿐만 아니라 장비가 설치되어 있지 않은 곳에서는 적용할 수 없다는 단점이 있다. 그래서, 측위를 위한 특수 장비가 전혀 필요 없는, 무선 근거리 통신망을 이용하는 방식이 이미 여러 곳에서 발표된 바 있다[4-9]. 그러나 이러한 방법들은 비교적 오차가 크다는 단점이 있다.

그래서 지금까지 옥내 측위 관련 연구는 측위의 정확도를 제고하는 목적에 초점을 맞추었다. 그럼에도 불구하고, 무선 네트워크 신호의 굴절, 반사, 간섭 등의 이유로 무선 근거리 통신망을 이용한 옥내 측위 방법은 오차가 아직도 비교적 크다. 따라서 본 논문은 지도 정보를 이용하여 무선 근거리 통신망을 이용한 측위 결과를 보정하는 방법을 제안한다. 제안하는 방법은 일련의 측위 결과에 칼만 필터[9,10]를 적용하여 보행자의 궤적을 구한 다음, 지도정합 방법으로 궤적을 보행자 통로로 수정한다. 이후로는 선분의 교차를 검사하는 함수를 이용하여 이동단말기가 장애물을 통과하였다는 측위 결과가 감지되는 즉시 이를 수정한다. 선분 교차 검사 과정의 처리 시간을 단축시키기 위하여 측위 영역을 작은 영역으로 구분함으로써 검사할 장애물의 수를 줄이는 전략도 적용한다.

제안하는 방법의 특징은 사용하는 칼만 필터의 인수 값이 이동 단말기의 위치에 따라 알맞게 조절된다는 것과 지도 정합을 옥내 측위 시스템에 최초로 적용하였다는 것이다. 나아가서 실시간 보정 방법이 옥내 측위 시스템에 적용되는 것도 본 논문이 최초로 제안하는 것이다.

기존의 옥내 측위 방법[7]으로 얻는 전형적인 일련의 측위 결과와 여기에 제안하는 방법을 적용하여 얻은 궤적을 비교하여 제안하는 방법의 타당성을 보인다.

본 논문은 다음과 같이 구성된다. 2절에서는 칼만 필터와 지도 정합 관련 연구를 조사하고, 3절에서는 제안하는 방법을 소개하고, 3절에서는 제안하는 방법을 구현한 결과를 소개하며, 4절에서는 성능 테스트 결과를 다룬다.

2. 기존의 연구

본 논문이 제안하는 보정 방안은 일련의 측위 결과에 칼만 필터를 적용하여 궤적을 구하고, 이 궤적을 지도상의 보행자 통로에 정합한 다음, 이후부터는 측위 결과를 실시간으로 보정한다. 따라서 기존의 연구로 칼만 필터와 지도 정합 방법을 살펴본다.

2.1 칼만 필터

상태 X(정지 상태의 위치는 2차원 벡터, 움직이는 상태는 위치와 가속도)가 식 (1), (2)와 같은 동역학 식과 조건으로 표현되면, 그림 1과 같은 과정으로 측정치를 구할 수 있다는 것이 칼만 필터이다.

$$X_k = AX_{k-1} + Bu_{k-1} + w_{k-1} \quad (1)$$

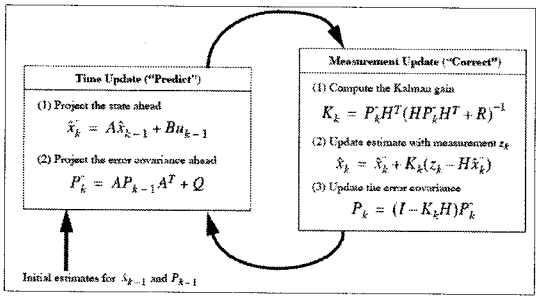


그림 1. 칼만필터 처리 과정

$$z_k = HX_k + v_k \tag{2}$$

단, k는 시간을 나타내고, w_k와 v_k는 각각 시스템 모델의 잡음과 측정 잡음을 나타내는 무작위 변수이며, 평균 0, 분산이 각각 Q와 R인 백색 가우시안 잡음으로 가정한다. 식 (1)은 이전 상태로부터 현재 상태를 구하는 함수이며, 식 (2)는 현재 상태와 측정치를 연관 짓는 함수이다[11].

2차원 평면에서 이동 객체의 상태를 측정할 때, 그림 1의 X, A, H는 각각 다음과 같다[12]. 단, 그림에서 $\hat{x}_k$, $\hat{x}_k^-$ 등 소문자 x는 X의 k번째 예측치, 예측치를 측정치로 갱신한 X와 같이 모두 X의 값이다.

$$X_k = \begin{bmatrix} x_k \\ y_k \\ v_{xk} \\ v_{yk} \end{bmatrix},\ A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

2.2 지도 정합

지도 정합 방법은 자동차 항법 시스템 분야에서 범지구 위치 결정 시스템 데이터로 구한 자동차의 궤적을 지도상의 도로로 정합하기 위하여 널리 사용된다. 가장 많이 사용되는 지도 정합 방법은 프레쉐 거리 (Frechet distance) 방법이다. 주어진 두 개의 복합선(polyline)의 프레쉐 거리는 다음과 같이 정의된다. 어떤 사람이 강아지를 데리고 산보를 하는데 사람은 하나의 복합선을 따라 걷고, 강아지는 다른 복합선을 따라 걷는다. 단, 진행 반대방향으로 걷는 것은 허용되지 않는다. 이러할 경우에 필요한 강아지 목 끈의 최소 길이를 프레쉐 거리라 한다.

참고문헌 [13]에 의하면, 곡선(curve)은 다음과 같은 연속 매핑으로 정의된다.

$f: [a,b] \to V$, 단 $a, b \in R$, $a \le b$ 그리고 (V,d)는 계량 공간으로 대표적인 예로 곡선(V)와 거리(d)를 들 수 있다. 두 곡선 $f: [a,b] \to V$와 $g: [a',b'] \to V$가 주어질 때, 이들 간의 프레쉐 거리는 식(3)과 같이 정의된다.

$$\delta_F(f,g) := \inf_{\alpha,\beta:[0,1]\to[0,1]} \max_{t\in[0,1]} \|f(\alpha(t)) - g(\beta(t))\| \tag{3}$$

단, α와 β는 각각 [0,1]에서 $[a,b]$와 $[a',b']$으로 가는 임의의 연속 증가함수이다.

참고문헌 [14]에는 두 복합선과 한계 거리가 주어질 때, 두 복합선의 어느 부분이 한계 거리 이내에 있는지 찾아주는 방법과, 이를 이용하여 두 경로간의 거리를 구하는 방법이 소개된다. 이 방법을 그림으로 설명하면 그림 2와 같다. 그림 2에서 하나의 복합선은 6개의 선(L1, ..., L6)으로 구성되었고, 다른 복합선은 단 하나의 선(L)이며, 한계 거리는 점선으로 주어졌다. 그림 2의 오른쪽에 보이는 사각형은 차례로 L1, L2, ..., L6를 나타내며, 높이는 다른 복합선 L을 나타낸다. 흰 부분은 L과 L1, L2, ..., 중 한계거리 이내에 있는 부분(자유공간이라 함)을 표시하며, 검은 부분은 한계거리 밖의 부분을 나타낸다.

지도 정합에서는 하나의 복합선이 궤적에 해당하고, 다른 복합선(L)이 도로를 구성하는 한 구간에 해당한다(두 교차로를 연결하는 길을 구간이라 한다.) 또한, 한계 거리는 측위 모듈의 오차 범위로 정하는 것이 적당하다. 도로망의 각 구간에 대하여 그림 2와 같은 자유공간을 구하여 이웃한 구간끼리 연결하면 좌하 단점에서 우상 단점을 연결하는 단조 증가 경로가 주어진 궤적에 가장 가까운 도로가 된다.

3. 제안하는 방법

무선 근거리 통신망 환경을 이용한 옥내 측위 방법과 같이 경제적으로 구현이 용이한 옥내 측위 방법은 오차가 크다는 단점이 있다. 따라서 본 논문은 옥내 측위 결과를 보정하여 오차를 줄이는 방법을 제안한다. 제안하는 방법은 그림 3에 보이는 바와 같이 옥내

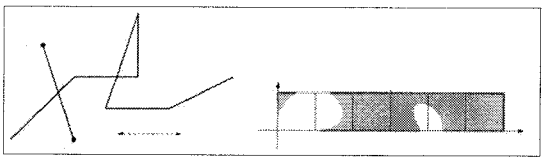


그림 2. 오른쪽 사각형에서 흰 부분이 한계거리 이내인 자유공간임.

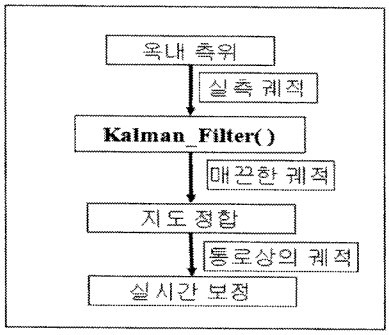


그림 3. 제안하는 방법의 요약

측위 결과에 칼만 필터를 적용하여 궤적을 구한 다음 지도상의 통로로 정합하고, 실시간으로 보정한다.

일련의 옥내 측위 결과로 실측 궤적을 얻게 되는데 전형적인 실측 궤적은 그림 4에 보이는 바와 같이 잡음이 심하다. 그림 4는 왼쪽으로부터 오른쪽으로 초속 0.5 미터의 속도로 걸어가면서 매초마다 옥내 측위 프로그램을 실행하여 얻은 실측 궤적으로 평균 오차는 3.53 미터이다.

본 논문은 실측값에 칼만 필터를 실시간으로 적용하여 매끈한 궤적을 구한다. 예를 들어 실측값이 그림 4와 같은 경우에 초기 40개의 실측치에 칼만 필터를 적용하면 그림 5와 같이 수렴한다. 칼만 필터 과정에서는 인수 값을 검사하여 수렴여부를 판정할 수 있다.

다음은 지도 정합 방법을 적용하여 칼만 필터로 구한 매끈한 궤적을 사람이 실제로 걸어 다닐 수 있는 공간 즉 지도 상의 통로로 정합시킨다. 지도상에는 일반적으로 수많은 통로가 있기 마련이다. 이들과 칼만 필터로 구한 매끈한 궤적과의 프레쉐 거리를 구하여 가장 가까운 통로로 그림 6과 같이 정합한다.

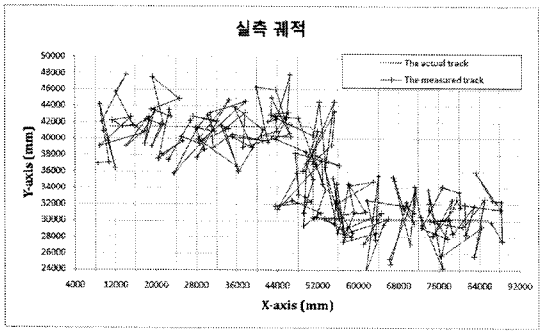


그림 4. 전형적인 실측 궤적

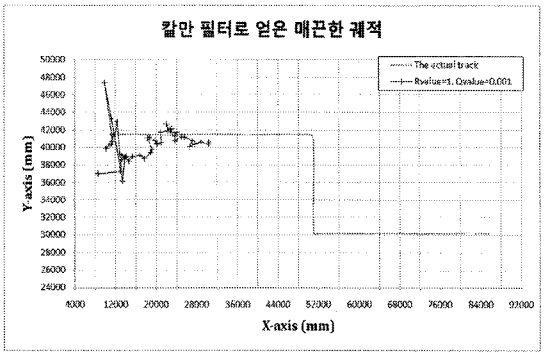


그림 5. 칼만 필터로 얻은 매끈한 궤적

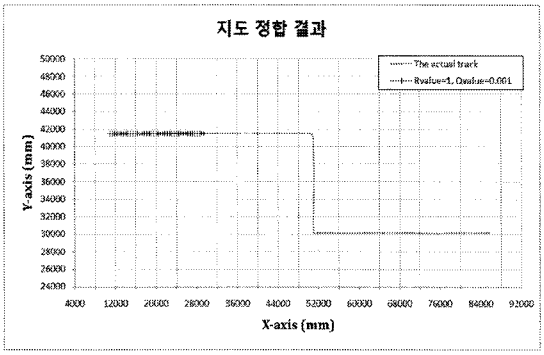


그림 6. 지도 정합 결과

프레쉐 거리가 실측 궤적의 오차 범위보다 작은 통로가 존재하지 않을 경우에는 더 많은 실측치에 칼만 필터를 적용하여 더욱 매끈한 궤적을 구하여 지도 정합을 다시 한다.

지도 정합이 성공적으로 완수된 이후에는 측정치를 실시간으로 보정한다. 즉, 측정치가 전자지도 상의 벽이나 장애물을 뚫고 지나가는 현상을 나타낼 때, 통로상의 위치로 보정하는 작업을 실시간으로 실행한다.

4. 제안하는 방법 구현

제안하는 방법은 일련의 측위 결과에 칼만 필터를 적용하여 칼만 필터 궤적을 구한다. 칼만 필터 궤적은 측위 결과보다 더 매끄럽다. 그리고 칼만 필터 궤적을 지도상의 통로에 정합한다. 다음부터는 측위 결과를 실시간으로 보정한다. 본 절은 각 과정의 구현 결과를 소개한다.

4.1 칼만 필터 구현

본 논문은 지도 정보를 이용하여 칼만 필터의 정확도를 제고하는 방안을 제안한다. 칼만 필터의 변수 Q와 R은 각각 시스템 모델의 잡음과 측정 잡음을 나타내는 무작위 변수이다. 따라서 시스템이 안정되어 있을 경우에는 Q 값이 작아야 계산 결과가 정확하고, 그렇지 않을 경우에는 Q 값이 커야한다.

이동 객체의 상태를 측정하는 경우에는 경로가 외길인 경우에는 시스템 모델이 안정적인데 반하여 교차로 영역에서 진행 방향을 바꿀 때에는 불안정해진다. 따라서 제안하는 방법의 기본 전략은 그림 7에 보이는 바와 같이, 이동 객체의 현재 위치가 교차로이면 Q값을 크게 하고 아니면 작게 하여주는 것이다.

그림 8은 이동 객체가 갈림길 영역 내부에 위치하는지 판단하는 방법을 그림으로 보인다. IntersectionList[]는 갈림길 영역에 대한 꼭지점의 좌표가 저장된 배열로서 도면의 갈림길 층에서 내용을 입력받는다. 그림에서 GetPos(300,300)은 GetPos라는 함수로 구한 현재 위치의 좌표가 (300, 300)이라는 뜻이고, (300, 0)은 현재 위치의 x좌표와 0으로 구성된 좌표이다. 현재 위치의 좌표인 (300, 300)과 (300, 0)가 결정하는 선분이 IntersectionList[]로 형성되는 선분들과 교차하는 회수가 짝수이면 현재 위치는 갈림길 영역 밖이며 홀수면 갈림길 영역 내부이다.

이와 같이, 현재 위치가 갈림길 영역 내부인지를 판단하는 알고리즘이 표 1에 보인다. Discriminate_

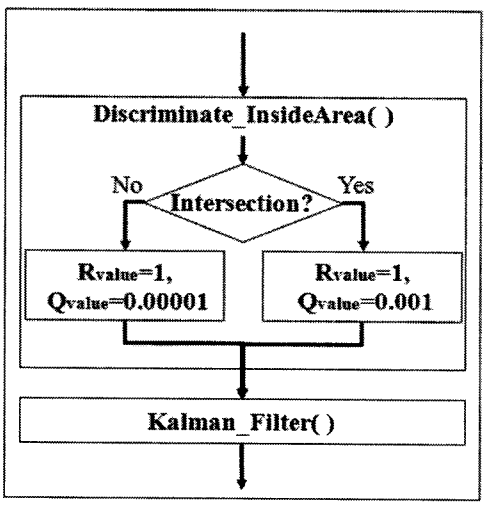


그림 7. 제안하는 칼만필터 방법

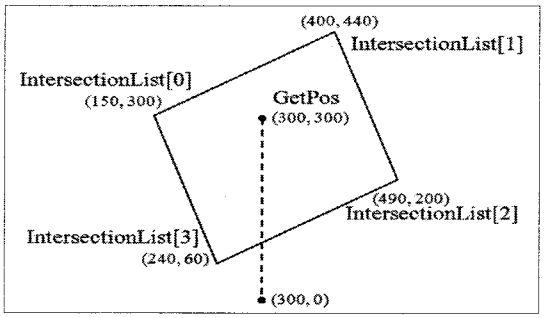


그림 8. 이동객체가 갈림길 영역 내부에 위치하는지 판단하는 사례

InsideArea() 알고리즘에서 IntersectionList[]는 갈림길 영역에 대한 꼭지점의 좌표가 저장된 배열이다. GetPos는 사용자의 위치를 반환함으로 GetPos가 IntersectionList[] 배열의 갈림길 영역 안에 위치하는지 판단한 후 영역 안에 위치하면 참 아니면 거짓을 반환한다.

이동 객체가 갈림길 영역 내부에 있는지 판단하는 방법은 이동 객체가 갈림길 경계를 홀수 번 교차했는지로 판단한다. 예를 들어 GetPos와 IntersectionList[] 배열의 값이 그림 8과 같을 경우 표 1의 1번 문장은 GetPos 좌표(300, 300)와 GetPos′ 좌표(300, 0)를 연결(그림 8의 점선)하는 직선의 방정식을 구한다. 2번 문장의 cntIntercept는 직선 (GetPos, GetPos′)가 갈림길 영역의 외곽선과 교차하는 횟수를 의미하며 초기 값은 0으로 설정한다. 문장 3, 4, 5에서는 직선 (GetPos, GetPos′)가 IntersectionList[] 배열의 각 요소에 대하여 교차여부를 판별한 후 교차하는 총 횟수를 cntIntercept에 저장한다. 문장 6, 7, 8에서는 교차하는 횟수(cntIntercept)가 홀수이면 참을 반환하고 짝수이면 거짓을 반환한다. 여기서 교차된 횟수가 홀수일 경우는 GetPos가 IntersectionList[] 배열(갈림길 영역) 안에 위치함을 의미한다.

본 알고리즘에 사용된 함수들의 실행 시간은 모두 상수이다. 따라서 본 알고리즘의 실행 복잡도는 IntersectionList[]의 크기에 비례한다. 일반적으로 지도에 나타나는 갈림길 영역은 4개의 선분으로 결정됨으로 본 알고리즘의 실행 복잡도는 상수라고 볼 수 있다.

4.2 지도정합

2절에서 소개한 프레쉐 거리는 라인의 끝점뿐만

표 1. 사용자의 위치가 갈림길 영역 내부에 위치하는지 판단하는 알고리즘

```
Algorithm Discriminate_InsideArea
 (GetPos, IntersectionList[])
GetPos는 사용자의 위치이다.
IntersectionList[]는 도면의 layer들 중에서
갈림길 영역의 꼭지점의 좌표가 저장된 배열이다.
GetPos가 IntersectionList[] 배열의 갈림길 영역 안에
위치하면 True, 아니면 False를 반환한다.
1. GetPos (X, Y)좌표를 시작점으로, GetPos' (X, 0)좌표
   를 끝점으로 하는 직선방정식을 구한다.
2. cntIntercept = 0 // cntIntercept는 직선 (GetPos,
   GetPos')가 갈림길 영역의 외곽선과 교차하는 횟수
   를 의미한다.
3. index=0
4. loop (index < size of List)
  4.1 직선 (IntersectionList[index-1], IntersectionList
      [index])를 구한다.
       // 직선 (GetPos, GetPos')와 직선
          (IntersectionList[index-1], IntersectionList
          [index])가 교차할 경우
4.2 if (직선 (GetPos, GetPos')와 직선 (IntersectionList
    [index-1], IntersectionList[index])의 기울기가 다르
    다.)
4.2.1 직선 (GetPos, GetPos')와 직선 (IntersectionList
      [index-1], IntersectionList[index])의 교점을 구함.
    4.2.2 if ((교점의 X 좌표가 IntersectionList[index-
          1]의 X 좌표와 IntersectionList[index]의 X 좌
          표 사이에 있다.) and (교점의 Y 좌표가
          GetPos의 Y 좌표와 0 사이에 있다.))
       4.2.2.1 cntIntercept 증가
    4.2.3 end if
  4.3 end if
  4.4 index 증가
5. end loop
6. if (cntIntercept가 홀수이다.)
  6.1 return (True) // GetPos가 갈림길 영역 안에 위치
      한다고 판정.
7. else
  7.1 return (False) // GetPos가 갈림길 영역 밖에 위치
      한다고 판정.
8. end if
End Algorithm선방정식을 구한다.
```

아니라 중간점에서도 거리를 구해야 함으로 계산이 복잡하여 본 논문은 이산적인 프레쉐 거리를 사용한다. 두 복합선의 이산적인 프레쉐 거리는 다음과 같이 정의된다.

$P:[0,n]\rightarrow V$를 복합선이라 하자. 이 복합선을 구성하는 각 라인의 끝점들 $(P(0), P(1), ..., P(n))$을 $\sigma(P)$로 표기하자. 그러면 두 개의 복합선 P와 Q의 끝점들을 각각 $\sigma(P)=(u_1,...,u_p)$와 $\sigma(Q)=(v_1,...,v_q)$로 표기할 수 있다. P와 Q의 짝 L은 다음과 같은 성질을 만족하는 $\sigma(P)\times\sigma(Q)$의 부분집합으로 구성된 열 (u_{a_1},v_{b_1}), $(u_{a_2},v_{b_2}),...,(u_{a_m},v_{b_m})$이다.

$a_1=1, b_1=1, a_m=p, b_m=q$이고, 모든 $i=1,...,q$에 대하여 $a_{i+1}=a_i$이거나

$a_{i+1}=a_i+1$이고 $b_{i+1}=b_i$이거나 $b_{i+1}=b_i+1$이다.

다시 말하면 P와 Q의 짝 L은 P와 Q의 점들의 순서를 준수한다. 순서쌍 (u_{a_i},v_{b_j})이 주어질 때, 점 u_{a_i}와 점 v_{b_j}의 거리를 $|(u_{a_i},v_{b_j})|$로 표기하며, P와 Q의 짝 L의 길이는 L을 구성하는 모든 순서쌍들의 거리 중 가장 큰 것으로 정의하며 $\|L\|$로 표기한다. 그러면 두 복합선 P와 Q의 이산적 프레쉐 거리는 P와 Q의 모든 짝 L 중에서 가장 짧은 길이로 정의한다.

본 연구에서 구현한 두 복합선 $P=(u_1,...,u_p)$와 $Q=(v_1,...,v_q)$의 이산적 프레쉐 거리를 구하는 알고리즘은 표 2와 같다. 이 알고리즘은 동적프로그래밍 전략을 채택하여 i가 1인 경우와 j가 1인 경우부터 이산적 프레쉐 거리를 구하여 $p\times q$ 배열 D에 저장한다. D[i,j]는 다음과 같이 이미 결정된 주변항의 최소값과 $|u_i,v_j|$중 큰 값으로 결정한다.

$$D[i,j]=\max(\min(D[i-1,j], D[i-1,j-1], D[i,j-1]), |u_i,v_j|).$$

따라서 본 알고리즘의 시간 복잡도는 $p\times q$이다.

칼만 필터 궤적을 하나의 복합선 P로하고 오차 범위내의 지도상의 통로를 다른 복합선 Q로하여 이들 간의 이산적 프레쉐 거리를 구한다. 오차 범위 내의

표 2. 두 복합선 P, Q의 이산적 프레쉐 거리를 구하는 알고리즘

```
Algorithm 이산적프레쉐거리(P, Q)
// P=(u1, ..., up), Q=(v1, ..., vq)
D[p,q]: real // 배열
for(i=1; i<=p, i++) {
  for(j=1: j<=q, j++) {
    if (i==1 and j==1) then D[1,1]=|u1, v1|;
    elseif (i==1 and j>1)
      then D[1,j] = max(D[1,j-1],|u1, vj|);
    elseif (i>1 and j==1)
      then D[i,1] = max(|ui, v1|, D[i-1, 1]);
    else D[i,j]= max(min(D[i-1,j],
           D[i-1, j-1], D[i,j-1]), |ui,vj|);
  }
}
return D[p,q];
End Algorithm
```

통로가 여럿 있으면 이들 각각에 대한 이산적 프레쉐 거리를 구하여 거리가 가장 가까운 것을 이동 객체의 경로로 정하여준다.

4.3 실시간보정

4.3.1 실시간 위치 보정 방법

지도 정합 이후에는 표 3의 Self_Adjusting() 메소드로 직전 측정 위치와 현재 측정 위치 사이에 장애물이 존재하는 비현실적인 측정 결과를 보정한다. 표 3의 알고리즘에서 PrePos는 직전 측정 위치이고 GetPos는 현재 측정 위치이다. 그리고 ObstacleList[] 배열은 도면의 층들 중에서 복도, 계단, 연구실 등의 장애물(벽)에 대한 시작점과 끝점의 좌표가 저장된 배열이다. 표 3의 알고리즘은 직선 (PrePos, GetPos)가 ObstacleList[] 배열의 요소(장애물)와 교차할 경우 보정된 위치를 계산하여 최종 위치로 반환하고, 교차하지 않을 경우 현재 측정한 위치(GetPos)를 최종 위치로 반환한다.

예를 들어 인수 PrePos, GetPos, ObstacleList[] 배열 값이 그림 9와 같은 경우에 표 3의 알고리즘을 적용하면, 문장 1은 직선 (PrePos(330, 100), GetPos(450, 400))를 구한다. 문장 2의 bIntercept는 직선 (PrePos, GetPos)와 ObstacleList[] 배열의 요소(장애물)와의 교차 여부를 의미하며, 최초 False (교차하지 않은 상태)로 설정한다. 문장 3, 4, 5에서는 직선 (PrePos, GetPos)가 ObstacleList[] 배열의 각 요소에 대하여 교차여부를 판별하며, 교차할 경우 문장 4.2.2.2에서 gCntIntercept 값을 1 증가시킨다. 여기서 gCntIntercept는 연속적인 교차 횟수를 의미하는 전역 변수로 표 3의 알고리즘이 수행될 때 직선 (PrePos, GetPos)와 ObstacleList[] 배열의 요소가 교차할 경우 값을 증가시키며, 교차는 하지만 연속된 교차 횟수가 임계치(Threshold)를 넘을 경우(문장 6.2), 그리고 전혀 교차하지 않는 경우(문장 7)엔 값을 0으로 초기화한다.

문장 6, 7, 8에서는 보정이 필요한 경우를 판별하여 문장 6.1처럼 보정된 위치를 계산하여 최종 위치로 반환하며, 보정이 필요 없을 경우 GetPos를 최종 위치로 반환한다. 여기서 보정된 위치는 그림 8의 점 P4로써 장애물로부터 Distance (300mm) 만큼 떨어진 지점이다.

표 3. 실시간 위치 보정 알고리즘

```
Algorithm Self_Adjusting (PrePos, GetPos,
                          ObstacleList[])
PrePos는 직전에 측정한 위치이고 GetPos는 현재 측
정한 위치이다.
ObstacleList[] 배열은 도면의 층들 중에서 직선
(PrePos, GetPos)가 내포되는 영역의 장애물이 저장
된 배열이다. 이 때 두 개 이상의 영역에 겹칠 경우
도면 전체의 장애물이 저장된다.
직선 (PrePos, GetPos)가 ObstacleList[] 배열의 요소
(장애물)와 교차할 경우 보정된 위치를 계산하여 반환
하고, 교차하지 않을 경우 GetPos를 반환한다.
1. 직선 (PrePos, GetPos)를 구한다.
2. bIntercept = False
   // bIntercept는 직선 (PrePos, GetPos)와 직선
   (ObstacleList[index-1], ObstacleList[index])의
   교차 여부를 의미한다. 최초 False(교차하지 않은
   상태)로 설정한다.
3. index=0
4. loop ((index < size of ObstacleList)
          and (bIntercept == False))
   4.1 직선 (ObstacleList[index-1],
          ObstacleList[index])를 구한다.
   4.2 if (직선 (PrePos, GetPos)와 직선
          (ObstacleList[index-1], ObstacleList[in-
          dex])의 기울기가 다르다.)
     4.2.1 직선 (PrePos, GetPos)와 직선
          (ObstacleList[index-1],
          ObstacleList[index])의 교점을 구한다.
     4.2.2 if (교점의 X 좌표가 PrePos의 X 좌표와
          GetPos의 X 좌표 사이에 있다.)
       4.2.2.1 bIntercept = True
       4.2.2.2 gCntIntercept 증가
// 두 직선방정식이 교차할 경우 값 증가.
     4.2.3 end if
   4.3 end if
   4.4 index 증가
5. end loop
6. if (bIntercept == True)
   6.1 if ((gCntIntercept != 0) and
          (gCntIntercept <= 임계치))
     6.1.1 return (보정 위치를 구한다.)
   6.2 else
     6.2.1 gCntIntercept = 0
     6.2.2 return (GetPos) // 최종 위치가 됨.
   6.3 end if
7. else
   7.1 gCntIntercept = 0
   7.2 return (GetPos) // 최종 위치가 됨.
8. end if
End Algorithm
```

본 알고리즘의 시간복잡도는 ObstacleList의 크기 즉 장애물의 수에 비례한다. 그래서 전체 지도를 그림 4와 같이 분할함으로써 ObstacleList의 크기를 작게한다.

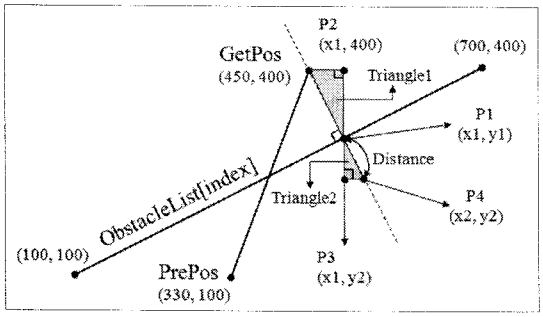


그림 9. 실시간 위치 보정 알고리즘에 사용된 인수들의 예

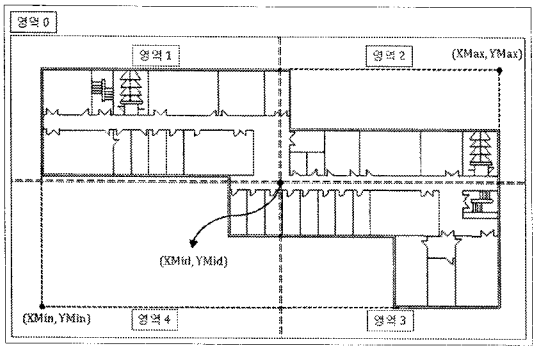


그림 10. 도면 지도의 중심점(XMin, YMin)을 기준으로한 각 영역 구분

처음 DXFImporter Class에서 DXF 파일에 저장된 실내 도면 지도를 읽어올 때, 도면 지도에 대한 X축의 최대(XMax), 최소(XMin) 좌표와 Y축의 최대(YMax), 최소(YMin) 좌표를 기억하게 된다. 이 ObstacleList[] 배열은 그림 10과 같이 각 구분된 영역마다 장애물의 시작점과 끝점 좌표가 저장되어 있다. 여기서 장애물은 도면 지도에서 직선으로 표현되는 것으로, 콘크리트 벽처럼 사용자가 통과할 수 없는 구조물을 의미한다.

각 영역을 구분하는 방법은 다음과 같으며 보정 알고리즘에 사용되는 인수 중 ObstacleList[] 배열은 직선 (PrePos(X_1, Y_1), GetPos(X_2, Y_2))가 속하는 영역의 장애물이 저장된 배열을 의미한다.

※ $XMid = XMin + (XMax - XMin)/2$,

$YMid = YMin + (YMax - YMin)/2$

영역 0 : 도면의 전체 영역.

영역 1 : $((X_1 \le XMid)\ And\ (X_2 \le XMid))$
$And\ ((Y_1 \ge YMid)\ And\ (Y_2 \ge YMid))$

영역 2 : $((X_1 > XMid)\ And\ (X_2 > XMid))$
$And\ ((Y_1 \ge YMid)\ And\ (Y_2 \ge YMid))$

영역 3 : $((X_1 > XMid)\ And\ (X_2 > XMid))$
$And\ ((Y_1 < YMid)\ And\ (Y_2 < YMid))$

영역 4 : $((X_1 \le XMid)\ And\ (X_2 \le XMid))$
$And\ ((Y_1 < YMid)\ And\ (Y_2 < YMid))$

4.3.2 식 클래스

표 1과 표 3의 알고리즘에서는 두 점을 지나는 직선방정식을 구하는 작업과 두 직선방정식이 교차할 경우의 교점을 구하는 작업, 그리고 그림 9처럼 보정된 위치를 계산하는 작업들이 사용되었다. 본 연구에서는 이러한 작업들을 처리하기 위해 그림 11과 같은 식 클래스(Equation Class)를 작성하였다.

식 클래스에서 LinearEquation() 메소드는 두 점 (x_1, y_1), (x_2, y_2)를 지나는 직선방정식 $y = ax + b$를 구하는 작업으로 다음과 같은 식 (4), (5)로 직선의 방정식을 구할 수 있다.

$$y = ax + b = \frac{(y_2 - y_1)}{(x_2 - x_1)}x - \frac{(y_2 - y_1)}{(x_2 - x_1)}x_1 + y_1 \quad (4)$$

$$a = \frac{(y_2 - y_1)}{(x_2 - x_1)},\ b = -\frac{(y_2 - y_1)}{(x_2 - x_1)}x_1 + y_1 \quad (5)$$

InterceptPoint() 메소드는 두 점 (x_1, y_1), (x_2, y_2)를 지나는 직선방정식 $y = ax + b$와 두 점 (x_3, y_3), (x_4, y_4)를 지나는 직선방정식 $y = cx + d$에 대하여 두 직선방정식이 교차할 경우의 교점 (x, y)를 구하는 작업으로 다음과 같은 방법으로 교점을 구할 수 있다.

$$y = ax + b = \frac{(y_2 - y_1)}{(x_2 - x_1)}x - \frac{(y_2 - y_1)}{(x_2 - x_1)}x_1 + y_1 \quad (6)$$

$$y = cx + d = \frac{(y_4 - y_3)}{(x_4 - x_3)}x - \frac{(y_4 - y_3)}{(x_4 - x_3)}x_3 + y_3 \quad (7)$$

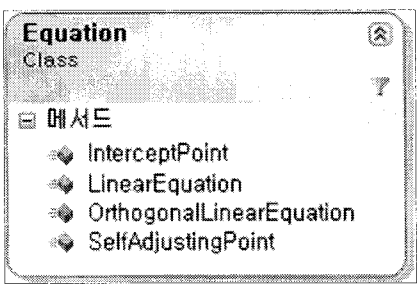


그림 11. 식 클래스에 정의된 메서드들

식 7을 식 6에 대입하여 식 8과 같이 x를 구한다.

$$ax+b=cx+d$$

$$x=\frac{(d-b)}{(a-c)} \tag{8}$$

$$=\frac{\frac{(y_2-y_1)}{(x_2-x_1)}x_1-\frac{(y_4-y_3)}{(x_4-x_3)}x_3-y_1+y_3}{\frac{(y_2-u_1)}{(x_2-x_1)}-\frac{(y_4-y_3)}{(x_4-x_3)}}$$

여기서 x를 식 6에 대입함으로써 다음과 같이 y를 구한다.

$$y=\frac{(y_2-y_1)}{(x_2-x_1)}\frac{\frac{(y_2-y_1)}{(x_2-x_1)}x_1-\frac{(y_4-y_3)}{(x_4-x_3)}x_3-y_1+y_3}{\frac{(y_2-u_1)}{(x_2-x_1)}-\frac{(y_4-y_3)}{(x_4-x_3)}}$$

$$-\frac{(y_2-y_1)}{(x_2-x_1)}x_1+y_1$$

SelfAdjustingPoint() 메소드는 <예 1>에서 그림 9의 보정된 위치 P4 (x2, y2)를 구하듯이 실시간 보정을 하고, OrthogonalLinearEquation() 메소드는 알고 있는 직선 $y=ax+b$와 직교하며 알고 있는 점 (x_1, y_1)를 지나는 직선의 방정식 $y=mx+n$을 다음과 같은 방법으로 구한다.

1) 두 직선 $y=ax+b$, $y=mx+n$이 직교하기 위한 조건은 $am=-1$ 이다. 따라서 $y=ax+b$에 직교하는 직선의 방정식은 $y=-\frac{1}{a}x+n$ 이다.
2) 알고 있는 점 (x_1, y_1)을 직선의 방정식 $y=-\frac{1}{a}x+n$에 대입하면 n을 구할 수 있다.

5. 제안하는 방법의 타당성 검증

기존의 연구에서 무선 근거리 통신망 환경의 옥내 측위는 오차가 상당히 큰 것으로 밝혀졌다. 그림 4의 점선을 평균 속도 0.5 미터/초로 걸으면서 그림 12와 같이 구현된 참고문헌 [7]에 소개된 옥내 측위 방법을 매초 한 번씩 적용하여 얻은 실측 궤적이 3절의 그림 4에 보인다. 이 실측 궤적의 평균 오차와 표준 편차는 각각 3.53미터와 1.8미터이다.

칼만 필터의 초기 인수 값을 표 4와 같이 설정하고, 이후 추적을 위한 인수인 측정치 오차 R(2×2 matrix) 행렬과 상태 오차 Q(4×4 matrix) 행렬의 적절한 값을 찾기 위해 행렬들의 값을 표 5와 같이 달리

예 1. 실시간 위치 보정 알고리즘을 그림 9에 적용하는 예

인수의 예가 그림 9와 같을 경우 보정된 위치를 계산하는 방법은 다음과 같다.

1) 두 점 (100, 100), (700, 400)을 지나는 직선방정식 Equation1을 구한다.
2) Equation1과 직교하며 점 GetPos를 지나는 직선방정식 Equation2를 구한다. 여기서 직교방정식은 OrthogonalLinearEquation() 메소드를 이용하여 구할 수 있다.
3) Equation1과 Equation2의 교점 P1 (x1, y1)를 구한다.
4) 그림 9에 보이는 것처럼 두 개의 직삼각형 Triangle1, Triangle2가 존재한다고 가정하면, 가상의 두 점 P2와 P3를 구할 수 있다.
5) Triangle1과 Triangle2는 닮음 꼴이기 때문에 보정된 위치 P4의 x2를 다음과 같은 방법으로 구한다.

   GetPos와 P2간의 거리 : P3와 P4간의 거리 = GetPos와 P1간의 거리 : Distance

   $\sqrt{(x1-450)^2}:\sqrt{(x2-x1)^2}$
   $=\sqrt{(x1-450)^2+(y1-400)^2}:Distance$

   여기서 x1은 이미 알고 있고, Distance(300mm) 역시 알고 있기 때문에 x2를 구할 수 있다.
6) 5)에서 구한 x2를 Equation2에 대입하여 y2를 구한다.

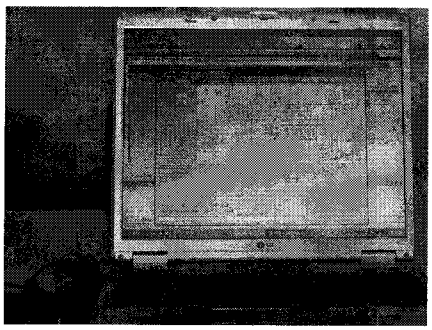


그림 12. 옥내 측위 프로그램의 사용자 인터페이스

표 4. 칼만필터에 사용될 초기 인수

Description	Value
$\vec{x}_0^-$	$[0\ 0\ 0\ 0]^T$
p_0^-	$\begin{bmatrix}100 & 0 & 0 & 0\\ 0 & 100 & 0 & 0\\ 0 & 0 & 100 & 0\\ 0 & 0 & 0 & 100\end{bmatrix}$

하며 실험하였다. 즉, 본 연구에서의 칼만 필터 실험들은 R 행렬의 값을 동일하게 유지하게 하며, 상태 오차(Q 행렬)를 변화함에 따라 이들 간의 적당한 비율을 찾는다.

표 5. R행렬과 Q행렬의 값 설정

$$R\text{행 렬} = \begin{bmatrix} Rvalue & 0 \\ 0 & Rvalue \end{bmatrix}$$

$$Q\text{행 렬} = \begin{bmatrix} Qvalue & 0 & 0 & 0 \\ 0 & Qvalue & 0 & 0 \\ 0 & 0 & Qvalue & 0 \\ 0 & 0 & 0 & Qvalue \end{bmatrix}$$

실험 index	*Rvalue*	*Qvalue*
실험 1	1	1
실험 2	1	0.1
실험 3	1	0.01
실험 4	1	0.001
실험 5	1	0.0001
실험 6	1	0.00001
실험 7	1	0.000001

그림 13은 표 5의 각 실험들에 대한 칼만 필터 궤적의 평균 오차를 보이고 있다. 결과를 보면 실험 4일 경우에 평균 오차가 1841.61mm로 가장 좋은 결과를 보였다. 즉, R 행렬의 *Rvalue*가 1, Q 행렬의 *Qvalue*가 0.001일 때 평균 오차가 가장 작았다. 그림 14, 그림 15, 그림16은 표 5의 실험 2, 4, 6에 대한 추적 경로를 보이고 있다. 추적 결과들을 비교해보면 *Qvalue*가 0.001보다 클수록 상태 오차가 커지기 때문에 측정치에 대한 가중치가 높아지는 결과를 볼 수 있다. 그리고 *Qvalue*가 0.001보다 작을수록 상태 오차가 작아지기 때문에 측정치에 대한 가중치가 작아지는 결과를 볼 수 있다. 여기서 점선은 실제 이동 경로를 의미하고 + 기호는 각 실험에서 얻은 칼만 필터 궤적을 의미한다.

그림 13을 보면 전체 추적 결과로는 실험 4일 때 평균 오차가 가장 작았다. 하지만 그림 14, 그림 15,

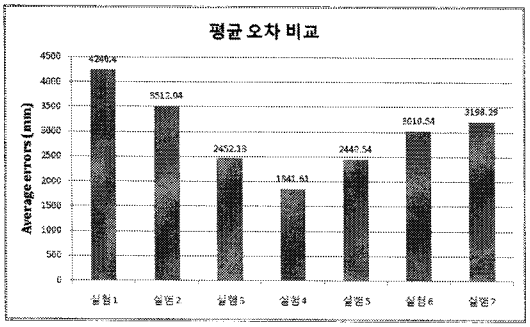


그림 13. 표 5의 실험들에 대한 KF 추적의 평균 오차 비교

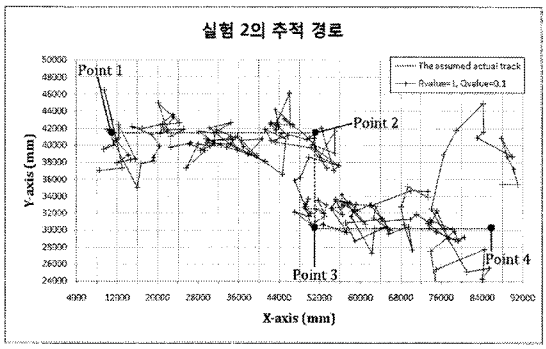


그림 14. 표 5의 실험 2에 대한 KF 추적 경로

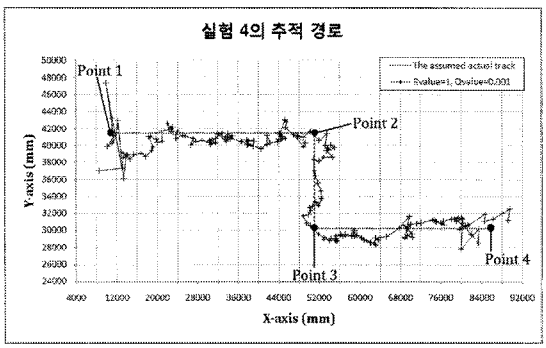


그림 15. 표 5의 실험 4에 대한 KF 추적 경로

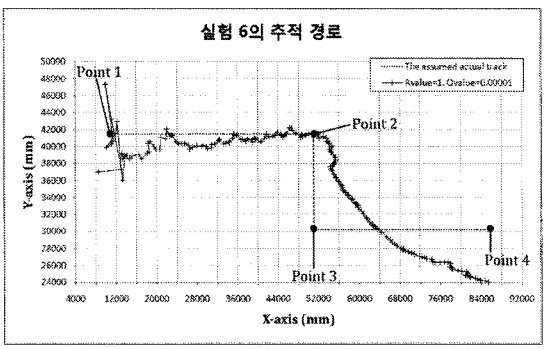


그림 16. 표 5의 실험 6에 대한 KF 추적 경로

그림 16을 보면 Point 1에서 Point 2까지(Area 1)의 부분적인 이동 경로에서는 실험 4보다 실험 6이 우수하였다. 또한 Point 2에서 Point 3까지(Area 2)의 부분적인 이동 경로에서는 실험 4가 우수하였다. 여기서 Point 2 지역과 Point 3 지역은 진행 방향이 바뀌는 영역으로 지도의 갈림길 영역이다. 이 갈림길 영역에 대한 정보는 이미 도면 지도에 저장되어 있다. 따라서 본 절에서는 칼만 필터를 위한 인수의 적절한 값을 찾기 위해 Area 1과 Area 2를 구분하여 표 5의

실험을 각기 수행하였고, 그 결과가 그림 17에 나타나 있다.

그림 17을 보면 갈림길 영역인 Area 2에서는 실험 4가 평균 오차 2259.92mm로 좋은 결과를 보였고, 갈림길 영역이 아닌 Area 1에서는 실험 6이 평균 오차 1639.9mm로 좋은 결과를 보였다. 즉, 칼만 필터의 추적을 위한 인수 값으로 갈림길 영역 에서는 *Rvalue* =1, *Qvalue*=0.001, 갈림길 영역이 아닌 곳에서는 *Rvalue*=1, *Qvalue*=0.00001 일 때 더 우수한 결과를 낳는다.

본 논문이 제안하는 옥내 측위 보정 방안을 정리하면 그림 18과 같다. 즉, 갈림길 영역에서는 *Rvalue* =1, *Qvalue*=0.001로 설정하고, 갈림길 영역이 아닐 때는 *Rvalue*=1, *Qvalue*=0.00001로 설정하여 칼만 필터를 수행한다. 또한 칼만 필터 결과가 벽과 같은 장애물을 통과하는 비현실적인 경우에는 실시간 위치 보정 작업을 수행한다. 여기서 장애물의 위치는 도면

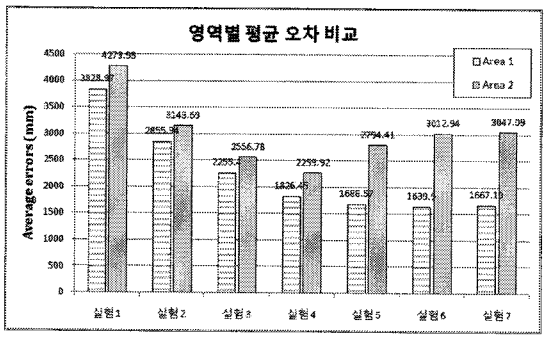


그림 17. Area 1과 Area 2에 대한 표 5의 실험 결과

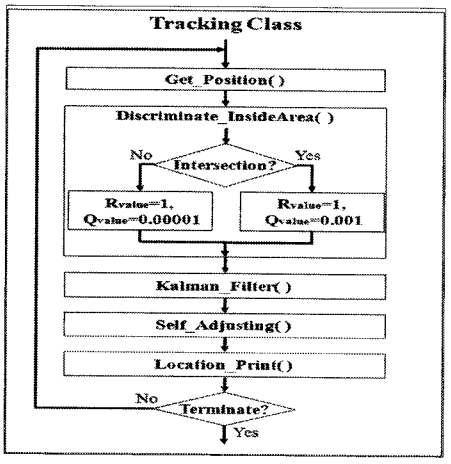


그림 18. 제안하는 방법의 이벤트 흐름도

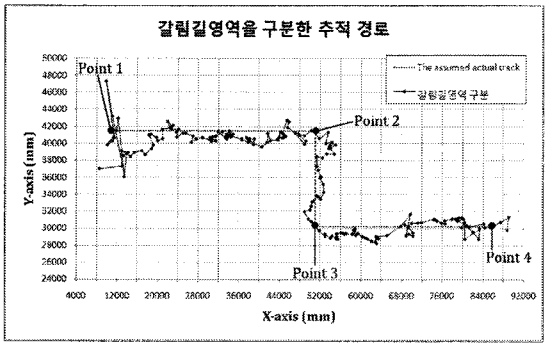


그림 19. 갈림길 영역을 구분할 경우의 KF 추적 경로

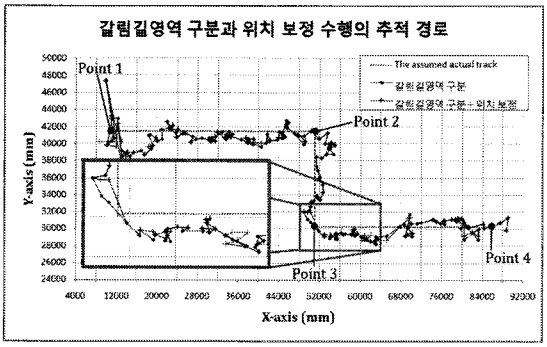


그림 20. 갈림길 영역을 구분한 후 위치 보정 작업을 수행한 KF 추적 경로

지도에 저장되어 있기 때문에 표 3의 알고리즘을 이용하여 사용자가 장애물을 통과하는지를 판단한다.

그림 19는 제안하는 방법 중 갈림길 영역을 구분했을 경우의 칼만 필터 궤적으로서, 평균 오차는 1806.88mm가 나왔다. 여기에 위치 보정 작업을 추가한 결과 그림 20과 같은 보정된 추적 경로를 얻을 수 있으며, 최종적으로는 평균 오차가 1791.48mm로 개선된 결과를 볼 수 있다. 그림 20의 왼쪽 아래에 확대된 그림은 보정 작업이 수행된 결과 중 일부를 크게 보인 것이다.

6. 결 론

본 논문은 옥내 측위 결과를 지도 정보를 이용하여 보정하는 방법을 제안하고, 제안하는 방법을 전형적인 실측 궤적에 적용하여 효율성을 검증하였다. 제안하는 방법은 첫째, 지도의 갈림길 영역 정보를 이용한 칼만 필터 방법이고, 둘째, 실시간 보정까지 가미하는 방법이다. 첫 번 방법은 오차를 49% 개선하

는 실험 결과를 보였으며, 두 번째 방법은 첫 번 방법의 오차를 1% 더 개선하는 실험 결과를 보였다.

참 고 문 헌

[1] K. Virrantaus, J. Veijalainen, and J. Markkula, "Developing GIS-Supported Location-Based Services," *Proc. of the Second International Conference on Web Information Systems Engineering*, Vol.2, Dec. 3-6 2001, pp. 66-75, 2001.

[2] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, "Tracking Moving Devices with the Cricket Location System," *Proc. of MobisSYS'04*, June 2004.

[3] R. Want, A. Hopper, V. Falco and J. Gibbons, "The Active Badge Location System," *ACM Transactions on Infomation Systems* 10, pp. 91-102, 1992.

[4] C. Wann, & M. Lin, "Data Fusion Methods for Accuracy Improvement in Wireless Location Systems," *Proc. of the IEEE Wireless Communications and Networking Conference,* (WCNC 2004) Vol.1, 21-25 Mar. 2004, pp. 471-476, 2004.

[5] M. Youssef, A. Agrawala, & A.U. Shankar, "WLAN Location Determination via Clustering and Probability Distributions," *Proc. of IEEE International Conference on Pervasive Computing and Communications (PerCom),* 23-26 Mar. 2003, pp. 143-150, 2003.

[6] M. Youssef, & A. Agrawala, "Continuous Space Estimation for WLAN Location Determination Systems," *Proc. of 13th International Conference on Computer Communications and Networks,* (ICCCN 2004) pp. 161-166, 2004.

[7] J. Yim, "Introducing a Decision Tree-based Indoor Positioning Technique," *Expert Systems with Applications,* Vol.34, Issue 2, pp. 1296-1302. 2008.

[8] M. Ciurana, F. Barcelo-Arroyo, and Izquierdo, F., "A ranging system with IEEE 802.11 data frames," *IEEE Radio and Wireless Symposium,* 9-11 Jan. 2007, pp. 133-136, 2007.

[9] M. Ciurana, F. Barcelo-Arroyo, S. Cugno, "A novel TOA-based indoor tracking system over IEEE 802.11 networks," *16th Mobile and Wireless Communications Summit,* IST, 1-5 July 2007, pp. 1-5, 2007.

[10] P. Hwang, and R. Brown, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley and Sons.

[11] G. Welch, and G. Bishop, "An Introduction to the Kalman Filter," Updated: July 24, 2006, *http://www.cs.unc.edu/~welch/Kalman/kal*manIntro.html

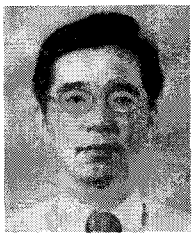
[12] H. Wang, H. Lenz, A. Szabo, J. Bamberger, and U. Hanebeck, "WLAN-Based Pedestrian Tracking Using Particle Filters and Low-Cost MEMS Sensors," *Proc. of the 4th Workshop on Positioning, Navigation and Communication*, Mar. 2007, pp. 1-7, 2007.

[13] H. Alt, and M. Godau. "Measuring the resemblance of polygonal curves." *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pp. 102-109, 1992.

[14] A. Maheshwari, and J. Yi, "On Computing Frechet Distance of Two Paths on a Convex Polyhedron," *EWCG 2005, Eindhoven,* Mar. 9-11, pp. 41-44, 2005.

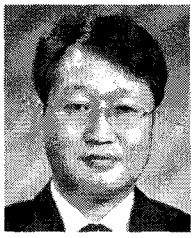
임 재 걸

1981년 동국대학교 전자계산학과 졸업
1987년 일리노이대학교 시카고 캠퍼스 컴퓨터과학 석사
1990년 일리노이대학교 시카고 캠퍼스 컴퓨터과학 박사
1992년~현재 동국대학교 과학기술대학 컴퓨터멀티미디어 학과 교수
관심분야 : 시스템 설계 및 분석, 인공지능, 페트리 넷 이론 및 응용.



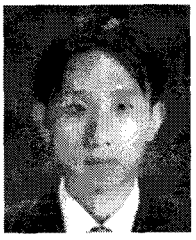
박 찬 식

1984, 1986, 1997년 서울대학교 제어계측공학과 학사, 석사, 박사
1984년~1997년 삼성전자 정보통신 책임연구원
1997년~현재 충북대학교 전기전자공학부 교수
관심분야 : GNSS, 자세결정, ITS, WSN.



심 규 박

1986년 동국대학교 대학원 통계학과 이학석사
1993년 동국대학교 대학원 통계학과 이학박사
1994년~현재 동국대학교 과학기술대학 정보통계학과 교수
관심분야 : 전산통계, 신뢰도검정, 통계 자료분석.



정 승 환

2007년 동국대학교 컴퓨터학과 졸업
2007년~현재 동국대학교 대학원 전자계산학과 석사과정 재학중
관심분야 : LBS, GIS, 802.11 표준