센서네트워크의 위치추정에 있어 플립오류에 강건한 스티칭 기법

(Flip Error Resistant Stitching in Sensor Network Localization)

권 오 흠 † 박 상 준 †† 송 하 주 †
(Oh-Heum Kwon) (Sangjoon Park) (Ha-Joo Song)

요 약 패치-스티치(patch-and-stitch) 기법을 사용하는 위치추정 알고리즘에서 발생하는 플립오류는 두 패치를 하나의 좌표계로 통합하는 과정에서 패치가 잘못 뒤집혀 병합되는 경우에 발생한다. 본 논문은 플립오류의 발생을 억제하는 앵커프리(anchor free) 패치-스티치 위치추정 알고리즘을 제안한다. 제안하는 알고리즘은 두 단계를 거쳐서 플립 오류의 가능성을 제거한다. 첫째, 각각의 인접한 패치 쌍에 대해서 플립모호성(flip ambiguity) 검사를 통해 플립오류의 발생가능성이 높은 패치 쌍을 찾아낸다. 둘째, 전역적인 수준에서 플립충돌(flip conflict) 검사를 통해 플립 오류의 가능성이 높은 패치 쌍을 찾아낸다. 시뮬레이션을 통한 성능 평가는 제안하는 알고리즘이 기존 것에 비해 더 우수한 위치추정이 가능함을 보여준다.

키워드 : 무선센서네트워크, 위치추정 알고리즘, 플립오류 방지

Abstract In patch-and-stitch localization algorithms, a flip error refers to the kind of error in which a patch is stitched to the map as being wrongly reflected. In this paper, we present an anchor-free localization algorithm which tries to detect and prevent flip errors. The flip error prevention is achieved by two filtering mechanisms: the flip-ambiguity test and the flip-conflict detection. We evaluate the performances of proposed techniques though simulations and show that they achieve significant performance improvements.

Key words : wireless sensor network, localization algorithm, flip error prevention

1. 서 론

센서네트워크는 밀접하게 배포된 다수의 센서 노드들로 구성되며 각각의 노드들은 자체적인 감지(sensing) 기능과 인접하는 노드와의 정보교환을 위한 통신 기능을 가진다. 감지된 데이타는 그것이 발생한 위치와 함께 전달되는 것이 일반적이므로 센서네트워크에서는 각 센서들의 위치를 파악하는 것이 필수적이다. 노드의 위치를 파악하기 위한 가장 간단한 방법은 각 노드마다 GPS(Global Positioning System)를 부착하는 것이다. 그러나 이는 비용이 많이 들며, 물리적인 크기와 에너지 소모 측면에서 제약이 많은 센서 노드에 적용하기는 어렵다. 특히, GPS는 인공위성의 전파가 도달하기 어려운 건물 내부, 지하 또는 장애물이 존재하는 환경에서는 사용할 수가 없다.

이러한 GPS의 한계 때문에 노드들 간의 거리를 이용하여 위치를 추정하는 알고리즘에 대한 연구가 꾸준히 진행되어 왔다. 노드들 간의 거리를 측정하는 기술은 크게 ToA(Time of Arrival), TDoA(Time-Difference of Arrival), 그리고 RSSI(Received Signal Strength Indicator) 등으로 구분할 수 있다[1]. ToA와 TDoA는 전파의 전달 지연시간을 이용하는 반면, RSSI는 전파의 세기를 이용하여 거리를 측정한다.

1.1 기존의 연구

위치추정을 위해서 지금까지 다양한 알고리즘들이 제안되었다[1]. 일부 알고리즘은 수작업 또는 GPS를 통해

· 이 논문은 2008년 교육과학기술부로부터 지원받아 수행된 연구임(지역거점연구단육성사업/차세대물류IT기술연구사업단)

† 정 회 원 : 부경대학교 전자컴퓨터정보통신공학부 교수
ohkwn@pknu.ac.kr
hajusong@pknu.ac.kr
†† 정 회 원 한국전자통신연구원 RFID/USN연구본부 연구팀장
sangjoon@etri.re.kr

논문접수 : 2008년 8월 7일
심사완료 : 2008년 10월 28일

이미 자신의 위치를 알고 있는 앵커노드(anchor node)의 존재를 가정한다[2,3]. 앵커노드의 존재는 위치추정 작업을 단순화시켜 주지만 수작업 또는 GPS를 통한 초기화과정에서 추가적인 비용이 발생한다. 반면, [4-7]의 연구는 앵커노드를 사용하지 않는 앵커프리(anchor-free) 위치추정 알고리즘들이다. 이들은 미리 정의된 좌표계를 사용하지 않기 때문에 노드의 위치는 임의로 정의된 상대좌표계에서 표현된다. 따라서 노드들의 좌표는 이동 및 방향의 자유로운 변경이 가능하다.

점진적 알고리즘(incremental algorithm)은 앵커프리 위치추정 알고리즘의 가장 단순한 형태이다[7]. 이 알고리즘은 일반적으로 몇 개의 시작노드(seed node) 집합을 선택하여 이들에게 그들 간의 거리 측정값에 부합하는 좌표를 부여한다. 그런 다음 다른 노드들을 반복적으로 이 집합에 추가한다. 새로운 노드를 추가하는 것은 추가할 노드와 기존 노드 집합 내에 존재하는 3개 이상의 노드들과의 거리가 측정되면 가능하다. 이 알고리즘은 단순하고 분산적 구현이 용이하지만 오류가 심하게 누적되는 경향이 있어 위치추정의 성능이 좋지 않다고 알려져 있다.

Priyantha[6] 등은 AFL이라는 2단계의 앵커프리 위치추정 알고리즘을 제안하였다. 첫째 단계에서는 노드들 간의 연결성에 관한 정보만을 이용하여 초기 좌표를 부여 하고, 둘째 단계에서는 질량-용수철(mass-spring) 기법이라고 부르는 반복적 최적화 기법을 이용하여 추정된 위치를 교정해 나간다. AFL 알고리즘은 첫째 단계에서 모든 노드들이 장방형 모양의 영역에 흩어져 있는 것으로 가정한다. 따라서 실제 영역이 불규칙한 모양이거나 영역 내에 장애물이 존재하는 비등방적인(anisotropic) 토폴로지(topology)의 경우에는 초기 좌표 추정이 매우 나빠지며, 이 경우 둘째 단계에서 오류가 줄어드는 것을 보장할 수 없다.

1.2 패치-스티치 위치인식 알고리즘

본 논문은 패치-스티치 알고리즘으로 불리는 앵커프리 위치추정 알고리즘을 다룬다[4,5,8,9]. 우선 모든 노드들은 패치(patch)라 불리는 지역 맵(map)을 형성한다. 일반적으로 패치는 한 노드와 그것으로부터 한 홉(hop) 떨어진 (모든 혹은 일부의) 노드들로 구성된다. 각각의 패치는 그것이 포함하고 있는 노드들에 대해 임의의 상대 좌표계에서의 좌표를 부여한다. 패치들이 구성되고 나면 인접한 패치들을 서로 병합하여 최종적으로 단일한 전역 맵을 구성하게 된다. 두 패치를 병합하는 일을 스티칭(stitch)한다고 말한다.

패치를 생성하는 방법에 대해서는 몇 가지 서로 다른 방법이 제안되었다. Meertens와 Fitzpatrick[8]은 한 홉 떨어져 있는 이웃한 노드들로 패치를 구성한다. 각 패치는 3개의 시작 노드로부터 출발하며 멀티레터레이션(multilateration) 기법으로 새로운 노드들을 추가함으로써 구성한다. Moore[5]는 거리에 관한 특정한 조건을 만족하는 완전 연결된(fully connected) 네 개의 노드를 하나의 패치로 생각하는 Robust Quadrilateral 알고리즘을 제안했다. Ji[4]와 Shang[9]은 패치를 구성하기 위해 다차원 스케일링(MDS, multi-dimensional scaling) 기법을 사용하였다.

2차원 평면에서 임의의 두 패치가 일직선상에 있지 않은 3개 이상의 공통 노드를 가진다면 그림 1에 나타낸 것처럼 두 패치를 모호함 없이 병합할 수 있다. 병합은 하나의 패치를 다른 패치에 평행이동(translation), 회전이동(rotation) 그리고 대칭이동(reflection) 시켜 공통된 노드들이 가장 잘 겹쳐지도록 위치시키는 것이다. 이 문제는 절대방향 문제(absolute orientation problem)[10] 또는 프로크루스테스 분석(Procrustes analysis)으로 불리어 왔으며, 이 변환을 절대방향변환(absolute orientation transformation)이라고 부른다. Horm[10]등은 절대방향변환에 대한 선형계산시간(linear time)을 갖는 닫힌 해(closed-form solution)를 제시하였다.

그림 2는 패치-스티치에 있어서 발생할 수 있는 최악의 오류를 나타낸 것이다. 그림 2(b)를 그림 1(b)와 비교해 보면, 노드 v_1이 실제 위치보다 약간 오른쪽에 치우쳐 있는데, 이런 일은 노드들 간의 거리측정 오차가 비교적 작은 경우에도 충분히 발생할 수 있다. 이 경우에 절대방향기법에 따라 두 패치를 스티치하면 그림 2(c)와 같은 결과를 낳게 된다. 우리는 이와 같은 스티치 오류를 플립오류(flip error)라고 부른다. 즉, 플립오류는 절대방향기법에서 잘못된 대칭이동을 수행하는 것을 의미한다. 단지 몇 개의 플립오류만으로도 위치추정의 결과는 매우 나쁘게 되므로 플립오류를 방지하는 것은 위치추정 알고리즘에 있어 매우 중요한 문제이다.

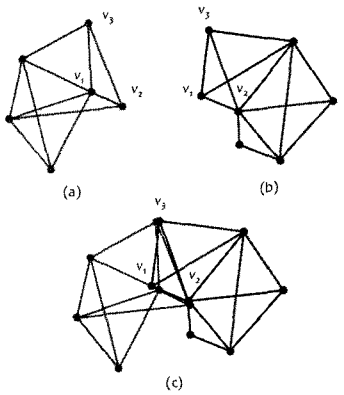


그림 1 패치-스티치의 예

1.3 본 논문의 결과

본 논문에서는 플립오류의 발생을 억제하는 앵커프리의 패치-스티치 위치추정 알고리즘을 제안한다. n개의 패치가 존재할 때, 3개 이상의 공통 노드가 존재한 모든 패치 쌍에 대해서 절대방향변환을 구한다면 이런 상황을 하나의 그래프로 표현할 수 있다. 즉, 그래프의 정점은 각각의 패치를 표현하고, 두 패치 간에 절대방향변환이 존재하면 에지로 연결한다. 우리는 이 그래프를 플립그래프(flip graph)라고 부른다. 본 논문에서는 각각의 절대방향변환이 대칭이동을 포함하는지 아닌지에 따라 "대칭적(reflectional) 절대방향변환"과 "비대칭적(reflection-less) 절대방향변환"으로 구분한다.

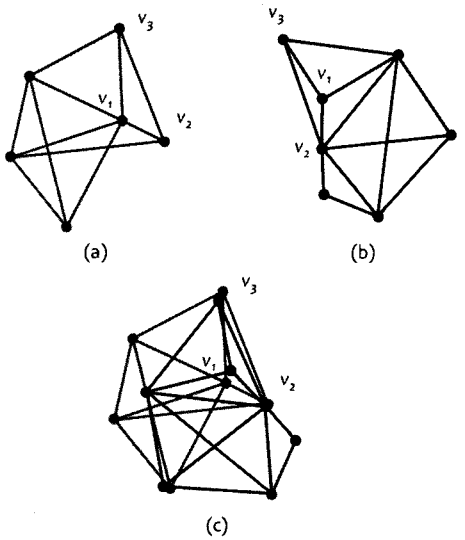


그림 2 플립오류의 예

제안하는 기법은 플립오류를 검출하고 방지하기 위해 두 가지 방법을 사용한다. 첫 번째는 플립그래프의 각 에지를 검사하여 플립모호성(flip-ambiguity)라 불리는 값을 에지별로 부여하고 그 값이 일정수준(threshold) 이상이면 플립그래프에서 해당 에지를 제거하는 것이다. 여기서 어떤 에지의 플립모호성(flip-ambiguity)이란 해당하는 두 패치를 대칭 이동을 포함하여 스티치한 경우와 대칭이동 없이 병합한 경우에 대해 발생하는 스티치 오차의 비율을 의미한다. 본 논문에서는 Horn이 제시한 해에서 별도의 추가적인 계산 없이 플립모호성을 계산할 수 있음을 보였다.

두 번째 기법은 플립그래프에서 에지의 일관성을 검사하는 것이다. 예를 들어 그림 3은 플립 그래프의 일부를 나타낸 것으로 0으로 표시된 에지는 대칭 없는 절대방향변환을 나타낸 것이고 1로 표시된 것은 대칭적인

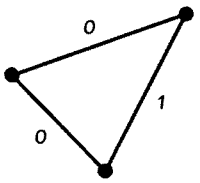


그림 3 플립충돌이 존재하는 플립 그래프의 예

절대방향변환을 나타낸다. 이 그래프는 대칭변환과 관련하여 하나의 모순이 존재함을 보여주며, 이와 같은 경우를 플립충돌(flip-conflict)이 일어난 것으로 정의한다.

제안하는 기법은 주어진 플립그래프에서 최소의 에지만을 제거함으로써 플립충돌을 제거하고자 한다. 이것은 매우 흥미로운 그래프 이론적 문제로 표현될 수 있으며, 우리는 이 문제를 코인플립(coin-flip) 문제라고 부른다. 이 문제는 NP-hard임을 쉽게 보일 수 있으며, 본 논문에서는 단순 반복형 휴리스틱(heuristic)을 이용하여 이 문제를 해결하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문에서 사용하는 표기들에 대한 설명과 더불어 절대방향문제에 대한 Horn의 기법을 간략하게 소개한다. 3장에서는 절대방향문제에 있어 플립모호성을 어떻게 계산할 수 있는지에 대해 설명하고 4장에서는 제안하는 위치추정 알고리즘을 제시한다. 5장에서는 시뮬레이션을 통해 제안하는 기법의 성능을 확인한다. 6장에서 결론을 맺는다.

2. 용어정리 및 절대방향변환 기법

센서네트워크는 하나의 그래프 $G=(V,E)$로 표현이 가능하다. 각각의 정점 $v\in V$는 하나의 센서노드를 의미하고 직접통신이 가능한 두 노드 v와 w를 하나의 에지 $(v,w)\in E$로 표현한다. 각 에지에는 두 센서노드간의 거리 측정값이 부여되어 있으며 $d(u,v)$로 나타낸다. $N(v)$는 센서노드 v에 이웃하는 센서노드들의 집합을 나타내고, $N^+(v)=N(v)\cup\{v\}$로 정의한다.

각각의 패치는 하나의 중심노드(center node)와 중심노드에 인접한 노드들로 구성된다고 가정하며, 노드 v를 중심 노드로 하는 패치를 P_v로 표기한다. 보다 정확히 기술하면 패치 P_v는 순서쌍 (M_v,g_v)로 정의된다. 여기서 M_v는 V의 부분집합이며 g_v는 M_v의 각 노드에 2차원 좌표를 대응시키는 함수 $g_v : M_v \rightarrow R^2$이다. 노드 u에 할당된 좌표 $g_v(u)$를 '노드 u의 P_v에서의 지역좌표(local coordinate)'라고 부른다. 반면, 각각의 노드가 패치-스티치의 최종 단계에서 할당받는 전역 맵에서의 최종 좌표를 '전역좌표(global coordinate)'라고 부를 것이다. 본 논문에서 각 패치는 중심노드와 그것으로부터 1-

홉 떨어진 노드들의 부분집합으로 구성된다고 가정한다. 즉, $M_v \subseteq N^+(v)$이다.

스티치될 두 패치를 각각 P_u와 P_v라하고 편의상 P_u를 왼편, P_v를 오른편 패치라고 부르자. 두 패치 사이에 공통 노드가 $m \geq 3$개 존재한다고 가정하고, 공통 노드의 양쪽 패치에서의 좌표를 (l_i, r_i), $i=1,\cdots,m$로 나타내자. 즉, l_i와 r_i는 각각 공통 노드 i의 왼편 패치(P_u)와 오른편 패치(P_v)에서의 좌표를 의미한다. 본 논문에서 좌표는 2차원 열벡터(column vector)로 나타낸다. 절대방향변환은 오른편 패치 즉, P_v의 좌표들을 평행이동(t)과 회전 및 대칭이동(R) 시켜 공통 노드들의 양쪽 좌표 간의 오차의 제곱이 최소가 되도록 하는 것이다. 이는 다음과 같은 식으로 나타낼 수 있다.

$$(t,R) = \underset{(t,R)}{\operatorname{argmin}} \sum_{i=1}^{m} \|e_i\|^2$$
$$e_i = l_i - (Rr_i + t)$$

Horn[10]은 이 문제를 선형시간 내에 풀 수 있는 닫힌(closed form) 해를 제시하였고, [1]에서는 Horn의 해에서의 계산의 각 단계들을 보다 쉽고 명료하게 요약하여 기술하였다. 여기서는 본 논문의 자기완결성을 위하여 [1]에서 기술한 단계들을 다음과 같이 재요약한다.

$\bar{l}$과 $\bar{r}$을 각각 두 패치에서 공통인 노드들의 중심점(centroid)라 하면 이들은 다음과 같이 나타낼 수 있다.

$$\bar{l} = \frac{1}{m}\sum_{i=1}^{m} l_i,\ \bar{r} = \frac{1}{m}\sum_{i=1}^{m} r_i$$

각 패치 내의 모든 노드들에 대해 중심점을 기준으로 한 새로운 좌표($\bar{l_i}$와 $\bar{r_i}$)를 다음과 같이 부여할 수 있다.

$$\bar{l_i} = l_i - \bar{l},\ \bar{r_i} = r_i - \bar{r},\ i=1,\cdots,m$$

그러면 각 노드에서의 오차 e_i는 다음과 같고,

$$e_i = \bar{l_i} - (R\bar{r_i} + t'),\ t' = t - \bar{l} + R\bar{r}$$

따라서 두개의 패치를 스티치함으로써 발생하는 오차는 다음과 같이 표현된다.

$$\sum_{i=1}^{m}\|e_i\|^2 = \sum_{i=1}^{m}\|\bar{l_i} - R\bar{r_i} - t'\|^2$$
$$= \sum_{i=1}^{m}\|\bar{l_i} - R\bar{r_i}\|^2 - 2t' \cdot \sum_{i=1}^{m}(\bar{l_i} - R\bar{r_i}) + m\|t'\|^2$$

여기서 두 번째 항은 항상 0이 되고, 첫 번째 항은 t'에 무관하며, 마지막 항은 항상 0보다 크거나 같으므로, 전체 에러는 $t'=0$, 즉, $t = \bar{l} - R\bar{r}$ 일 때 최소가 된다. 즉, 최적의 변환은 왼편 패치에서 공통 노드들의 중심점과 회전 이동한 오른편 패치에서의 공통 노드들의 중심점이 일치하도록 평행 이동할 때이다. 즉, 최적의 평행이동은 회전이동과는 독립적으로 정해진다. 이제 최적의 회전이동은 다음과 같이 구할 수 있다.

$$R = \underset{R}{\operatorname{argmin}} \sum_{i=1}^{m}\|e_i\|^2$$
$$= \underset{R}{\operatorname{argmin}} \sum_{i=1}^{m}\|\bar{l_i}\|^2 + \sum_{i=1}^{m}\|\bar{r_i}\|^2 - 2\sum_{i=1}^{m}\bar{l_i}\cdot(R\bar{r_i})$$

이 식은 다음과 같은 경우에 최소값을 가진다.

$$R = \underset{R}{\operatorname{argmax}} \sum_{i=1}^{m}\bar{l_i}\cdot(R\bar{r_i})$$

이것을 다시 정리하면

$$R = \underset{R}{\operatorname{argmax}}\ Tr(R^T M)$$
$$M = \sum_{i=1}^{m}\bar{l_i}(\bar{r_i})^T$$

여기서 Tr은 궤적함수(trace function)를 의미한다. M^TM을 고유분해(eigen decomposition)하면 각각의 고유값(eigen value)을 λ_1과 λ_2, 고유벡터를 u_1과 u_2라 할 때 다음과 같이 나타낼 수 있다.

$$M^TM = \lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T$$

이제, $S=(M^TM)^{1/2}$과 $U=MS^{-1}$을 각각 계산한다.

$$S = \sqrt{\lambda_1}\,u_1u_1^T + \sqrt{\lambda_2}\,u_2u_2^T$$
$$U = MS^{-1} = M\left(\frac{1}{\sqrt{\lambda_1}}u_1u_1^T + \frac{1}{\sqrt{\lambda_2}}u_2u_2^T\right)$$

여기서 $M=US$이고 $U^TU=I$이므로 U가 정규직교 행렬임을 감안하면 $Tr(R^TUS)$는 다음과 같이 표현할 수 있다.

$$Tr(R^TUS) = \sqrt{\lambda_1}\,Tr(R^TUu_1u_1^T) + \sqrt{\lambda_2}\,Tr(R^TUu_2u_2^T)$$

여기서 $Tr(R^Tu_1u_1^T)$는 $(Ru_i \cdot Uu_i)$와 같으므로

$$Tr(R^TUS) = \sqrt{\lambda_1}(Ru_1 \cdot Uu_1) + \sqrt{\lambda_2}(Ru_2 \cdot Uu_2) \quad (1)$$

$Tr(R^TUS)$는 $R=U$일 때 최대가 되므로 에러를 최소화하기 위한 R값은 다음과 같다.

$$R = U = M\left(\frac{1}{\sqrt{\lambda_1}}u_1u_1^T + \frac{1}{\sqrt{\lambda_2}}u_2u_2^T\right).$$

3. 플립 모호성(flip ambiguity)

일반적으로 잘못된 결정은 차라리 결정을 하지 않는 것보다 못한 결과를 낳는 경우가 많다. 그런 의미에서 패치-스티치 기법과 같이 연속적으로 어떤 결정을 내리고, 그 결과가 다음 결정에 영향을 주는 형태의 프로세스에서는 어떤 기준에 따라 내려진 결정을 각하하는(reject) 메커니즘이 필수적이라고 할 수 있다. 가장 단순한 방식의 방법은 오차의 평균값($\sum_{i=1}^{m}\|e_i\|^2/m$)을 이용하는 것이다. 즉, 각 절대회전변환에서 변환 오차의 평균이 일정 수준(threshold) 이상이 되면 해당 변환을 각하하는 것이다.

본 논문에서는 플립오류의 발생을 방지하는 것을 주목적으로 하는 새로운 각하 기법을 제안한다. 이 기법은

대칭이동의 적용 여부에 따른 두 가지 경우에 대해 각각 독립적으로 변환 오차를 계산하고 그것들을 비교하는 방식을 취한다. E_1과 E_2를 각각 대칭이동을 포함한 것과 그렇지 않은 경우에 발생하는 변환오차의 최소값이라 할 때, $\min(E_1,E_2)/\max(E_1,E_2)$를 '플립모호성값'으로 정의한다. 그리고 플립모호성 값이 일정수준(δ) 이상인 경우에는 해당 변환을 각하하는 것이다.

이제 남은 문제는 플립모호성 값의 계산방법이다. 행렬R은 이차원의 정규직교 행렬이므로 $R=[v_1 v_2]$로 나타낼 수 있다. 여기서 $v_i \in R^{2\times 1}$의 열벡터이다. 그러면 식 (1)은 다음과 같이 나타낼 수 있다.

$$\begin{aligned}
&Trace(R^T US) = \sqrt{\lambda_1}(Ru_1 \cdot Uu_1) + \sqrt{\lambda_2}(Ru_2 \cdot Uu_2)\\
&= \sqrt{\lambda_1}(u_1^T R^T U u_1) + \sqrt{\lambda_2}(u_2^T R^T U u_2)\\
&= \sqrt{\lambda_1}(u_1^T(v_1 w_1 + v_2 w_2)u_1) + \sqrt{\lambda_2}(u_2^T(v_1 w_1 + v_2 w_2)u_2)\\
&= \sqrt{\lambda_1}u_1^T v_1 w_1 + \sqrt{\lambda_1}u_1^T v_2 w_2 u_1 + \sqrt{\lambda_2}u_2^T v_1 w_1 + \sqrt{\lambda_2}u_2^T v_2 w_2 u_2\\
&= \sqrt{\lambda_1}u_1^T v_1 w_1 + \sqrt{\lambda_2}u_2^T v_1 w_1 + \sqrt{\lambda_1}u_1^T v_2 w_2
\end{aligned}$$

이것은 다시 다음과 같은 형태로 나타낼 수 있다.

$$Trace(R^T US) = av_1 + bv_2 \qquad (2)$$
$$a = \sqrt{\lambda_1}w_1 u_1 u_1^T + \sqrt{\lambda_2}w_1 u_2 u_2^T$$
$$b = \sqrt{\lambda_1}w_2 u_1 u_1^T + \sqrt{\lambda_2}w_2 u_2 u_2^T$$

R은 2×2의 정규직교 행렬이므로 v_2는 v_1과 직교하는 단위벡터이다. 이차원에서 v_1에 직교하는 단위벡터는 오직 두개만 존재한다. 그 두개의 단위벡터는 v_1을 시계방향으로 90° 그리고 반시계방향으로 90°를 각각 회전함으로써 구할 수 있다. 따라서 v_2는 v_1에 대해 다음과 같이 나타낼 수 있다.

$$v_2 = Av_1 \text{ 또는 } v_2 = A^T v_1, \qquad (3)$$
$$\text{여기서 } A = \begin{bmatrix} \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} \\ \sin\frac{\pi}{2} & \cos\frac{\pi}{2} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

(3)을 (2)에 대입하면 다음과 같은 식을 얻을 수 있다.

$$Trace(R^T US) = (a+bA)v_1 \text{ or } (a+bA^T)v_1$$

위 식에서 두 값은 각각 대칭이동을 포함한 경우와 그렇지 않은 경우를 나타낸다. 따라서 절대방향문제는 두 개의 선형최적화 문제에 대한 해를 구하되 그것들 중에서 목적함수(objective function) 값이 더 큰 것을 선택하는 것으로 볼 수 있다. 각각의 최적화 문제는 다음과 같은 형식을 취한다:

$\|v\|=1$ 일 때 cv의 값이 최대가 되도록 하라 (4)

여기서 $c \in R^{1\times 2}$이고 $v \in R^{2\times 1}$이다. $cv = \|d\| \cdot \|v\| \cos\theta = \|d\| \cos\theta$이므로 $\cos\theta = 1$일 때 cv는 최대값 $\|d\|$를 갖게 된다. 식 (2)를 전개해보면 $c = a + bA$인 경우에는 $\|d\| = \sqrt{\lambda_1} + \sqrt{\lambda_2}$이고 $c = a + bA^T$인 경우에는 $\|d\| = |\sqrt{\lambda_1} - \sqrt{\lambda_2}|$가 되는 것을 어렵지 않게 알 수 있다. 이것은 부가되는 계산 오버헤드 없이 플립모호성 값을 계산할 수 있음을 의미한다.

요약하면, 두개의 패치 P_u와 P_v에 대해 P_v의 좌표들은 그대로 둔 채로 P_u의 좌표들을 최적으로 이동시키는 회전이동 행렬과 평행이동 벡터를 각각 R_{uv}와 t_{uv}로 나타낸다. I_{uv}는 행렬 R_{uv}가 대칭이동을 포함하는 경우에는 $I_{uv}=1$이고 그렇지 않은 경우에는 $I_{uv}=0$이 되는 대칭이동 표시자(reflection indicator)로 정의한다. 세 순서쌍(R_{uv}, t_{uv}, I_{uv})를 'P_v에 대한 P_u의 AOT(Absolute Orientation Transformation)라고 부르고, 이 변환의 플립모호성이 기준값(threshold) δ 이하일 경우에 '유효한(valid) AOT'라고 정의한다. 유효하지 않은 모든 AOT들은 각하되며, 알고리즘에 의해서 더 이상 사용되지 않는다.

4. 위치추정 알고리즘

4.1 패치의 구축

기존 연구에서 사용된 대표적인 패치구축기법은 두 가지이다. 첫째는 MDS(Multi-Dimensional Scaling) 방식으로 [4,9]와 같은 기존 연구에서 사용하였다. MDS는 패치 내의 모든 노드들 간의 거리를 나타내는 하나의 행렬을 입력으로 받아 거리 행렬을 근접하게 만족시키는 점들의 좌표를 출력한다[11]. MDS는 몇 가지 단점이 있다. 첫째는 MDS는 모든 노드들 간의 거리를 요구하지만 패치 내의 일부 노드들은 서로 인접하지 않기 때문에 직접적으로 거리를 측정할 수는 없으며, 따라서 다른 방법을 통해 추정해야 한다. 둘째는 MDS 방식의 경우 분산 알고리즘이 제안된 바가 없다. 따라서 센터 노드가 패치 내의 모든 노드들 간의 거리를 알아야하고, 또한 각 노드들의 좌표를 결정한 다음 그 결과를 다른 노드들에게 알려야 한다.

패치 구축의 두 번째 방법은 반복적인 멀티레터레이션(multilateration) 방법이다. 이 방법은 중심 노드 v가 이웃한 노드 집합 $N(v)$ 중에서 거리 $d(v,u_1)$, $d(v,u_2)$, $d(u_1,u_2)$가 삼각형의 세변을 이룰 수 있는 두개의 노드 u_1과 u_2를 선택한다. 그다음엔 좌표계를 정의하고 위의 세 거리를 만족시키는 위치에 세 노드의 좌표를 부여한다. 그리고 나면 세 노드는 자신의 좌표를 이웃 노드들에게 전달한다. 이웃 노드들 중에서 좌표가 부여된 세 개 이상의 노드에 인접한 노드는 멀티레터레이션을 통해서 자신의 좌표를 결정하고, 그 좌표를 다시 이웃 노드들에게 전달한다. 이 과정을 반복하여 패치를 구성한다.

이 방법의 장점은 중심 노드에 작업부하가 집중되지 않는다는 점이다. 중심 노드 v는 자신과 이웃하는 노드와의 거리만 필요하고, 다른 노드들 간의 거리는 알아야 할 필요가 없다. 반복적 멀티레터레이션은 단계가 진행

될수록 추정 오차가 누적되는 경향이 있지만[8], 우리의 경우에는 중심 노드에서 1-홉 떨어진 노드들에 대해서만 멀티레터레이션을 적용하므로 오차누적은 큰 문제가 되지 않는다.

반복적 멀티레터레이션을 통한 패치 구축에서는 각 노드별로 자신의 패치를 형성하므로 임의의 노드 v는 최대 $|N(v)^+|$ 개의 패치에 소속할 수 있다. 이것은 곧 각 노드가 최대 $|N(v)^+|$ 개의 서로 다른 좌표계에서의 좌표를 가질 수 있음을 뜻한다. 노드 u를 중심으로 하는 패치에서 노드 v의 좌표를 $crd_u(v)$로 나타내자. 반복적인 멀티레터레이션에서는 각 노드가 자신의 좌표를 계산할 때마다 이웃 노드에 좌표를 전달하게 되므로 패치의 중심 노드는 부가적인 메시지 교환 없이 패치 내의 노드들에 대한 좌표를 수집할 수 있다. 따라서 각 노드는 다음과 같이 좌표들의 집합을 유지하는 것으로 가정한다.

$$\bigcup_{u\in N^+(v)} \{crd_u(w)|w\in N^+(v)\}$$

4.2 노드쌍별 AOT의 계산과 플립충돌의 해결

임의의 두 패치에 대해 그것들의 중심 노드가 이웃하고 적어도 세 개의 공통 노드를 가지는 경우에 '두 패치가 이웃한다'라고 부른다. 제안하는 알고리즘의 두 번째 단계는 이웃하는 패치들에 대해 AOT를 계산하는 것이다. 패치의 중심 노드 v는 각각의 이웃한 패치에 대해 한번씩 AOT를 계산해야 하므로 최대 $|N(v)|$ 개의 AOT를 계산해야 한다. 즉, 노드 v에 대해 AOT들의 집합 $\{(R_{vu},t_{vu},I_{vu})|u\in N(v)\}$를 계산한다. 노드 v가 AOT를 계산하기 위해 필요한 모든 좌표들을 가지고 있으므로 이 과정에서 추가적인 메시지 교환은 불필요하다.

다음 단계는 무효한 AOT의 비율이 일정 기준을 초과하는 패치들을 가려내는 것이다. 본 논문에서 제시한 시뮬레이션의 경우에는 임의의 패치 P_v에 대해 그것의 AOT집합 $\{(R_{vu},t_{vu},I_{vu})|u\in N(v)\}$ 중에서 반 이상이 무효한 경우에는 P_v를 가려내었다. 이렇게 가려내진 패치들을 '불량 패치'로 부른다. 불량 패치들은 패치 구성 자체가 많은 오류를 포함하고 있을 것으로 추정되는 것들이다. 우리의 알고리즘은 이들 불량 패치들을 스티칭 과정에서 소외시킴으로써, 이 패치들이 내재하고 있는 오류가 가능한 한 전체 위치인식에 영향을 주지 않도록 하는 것이다.

계산된 AOT들을 플립그래프(flip graph)로 표현할 수 있다. 불량 패치를 제외한 모든 패치가 플립 그래프의 노드가 된다. 두 노드 u와 v에 대해 $AOT(R_{uv},t_{uv},I_{uv})$가 유효한 경우에 (마찬가지로 (R_{vu},t_{vu},I_{vu})가 유효한 경우에), 두 노드들 연결하는 에지(u,v)가 존재한다. 이 경우에 에지 (u,v)에는 I_{uv}가 레이블로 표시되는 것으로 가정한다.

모든 패치는 최종 맵에 그대로 혹은 대칭이동 되어 나타날 것이다. 플립 그래프에서 에지 레이블이 0으로 표시된 것은 연결된 두 패치가 그대로 혹은 양측 모두 대칭이동 됨을 의미이며 레이블이 1인 경우는 둘 중 하나만 대칭이동 됨을 나타낸다. 이러한 상황을 그래프이론 문제로 보고 다음과 같은 동전뒤집기(coin-flip problem) 문제로 이름 짓는다. 각 에지들에 대해 0 또는 1로 레이블이 부여된 그래프를 가정한다. 각 정점에 동전을 놓되 에지 레이블이 0이면 연결된 정점에는 동전을 같은 면으로 놓는다. 만약 에지 레이블이 1이면 정점에는 동전이 서로 다른 면으로 놓인다.

위의 규칙을 만족하도록 모든 정점에 동전을 놓을 수 있으면 해당 그래프는 '일관성이 있는(consistent)' 것으로 정의하고 그렇지 않은 경우에는 '일관성이 없는(inconsistent)' 것으로 정의한다.

정리 4.1 임의의 그래프 G가 일관성을 가지기 위한 필요충분조건은 그래프 내에 에지 레이블들의 합이 홀수가 되는 사이클이 존재하지 않는 것이다.

증명: 먼저 합이 홀수가 되는 사이클이 존재할 경우 일관성이 없는 그래프임은 자명하다. 합이 홀수가 되는 사이클이 존재하지 않는다고 가정하자. 또한 연결된(connected) 그래프라고 가정하더라도 증명의 일반성이 훼손되지 않으므로 그렇게 가정하자. 다음과 같이 그래프의 정점들을 두 집합 X와 Y로 분할한다. 또한 두 집합은 항상 다음과 같은 조건을 만족하도록 유지한다.

1. 정점집합 $X\cup Y$에 의해 유도된(induced) 부그래프(subgraph)를 $G(X\cup Y)$라고 나타내자. 즉, $G(X\cup Y)$의 정점집합은 $X\cup Y$이고, 두 정점 $u,v\in X\cup Y$에 대해서 원래의 그래프 G에 에지 (u,v)가 존재하면 $G(X\cup Y)$에도 에지 (u,v)가 존재한다. 이때 $G(X\cup Y)$는 항상 연결된 그래프이다.
2. 부그래프 $G(X\cup Y)$에서 X에 속한 두 정점간에는 에지 라벨의 합이 짝수인 경로만이 존재하고,
3. Y에 속한 두 정점간에도 에지 라벨의 합이 짝수인 경로만이 존재하며,
4. 반면 X에 속한 한 정점과 Y에 속한 한 정점간에는 에지 라벨의 합이 홀수인 경로만이 존재한다.

먼저 임의의 한 정점을 집합 X에 포함시킨다. 이 상태에서 위의 네 가지 조건은 당연히 만족된다. 다음으로 다음의 과정을 모든 정점들이 두 집합 중 하나에 속할 때까지 반복한다: 아직 $X\cup Y$에 속하지 않은 정점들에 대해서 만약 집합 X에 속한 노드와 레이블 0인 에지로

연결되어 있거나, 집합 Y에 속한 정점과 레이블이 1인 에지로 연결되어 있으면 라벨 x를 부여하고, 반대로 X에 속한 노드와 레이블 1인 에지로 연결되어 있거나 집합 Y에 속한 정점과 레이블이 0인 에지로 연결되어 있으면 라벨 y를 부여한다.

이때 어떤 정점도 두 가지 라벨을 동시에 부여받지 않는다는 것을 보일 수 있다. 가령 임의의 노드 v가 두 라벨을 모두 부여받았다고 가정해 보자. 예를 들어서 집합 X에 속한 노드와 레이블 0인 에지로 연결되어 있으면서 동시에 집합 Y에 속한 노드와도 레이블 0인 에지로 연결되어 있는 경우를 가정해보자. 위의 조건 4에 의해서 에지 라벨의 합이 홀수인 사이클이 존재하게 된다. 이것은 가정에 위배된다. 나머지 경우들에 대해서도 유사하게 모순임을 보일 수 있다. 이제 라벨 x를 부여받은 모든 노드들은 집합 X에 포함시키고, 라벨 y를 부여받은 모든 노드들은 집합 Y에 포함시킨다. 이렇게 정점을 두 집합에 추가하더라도 위의 네 가지 조건은 그대로 유지된다는 것은 자명하다.

그래프 G는 연결된 그래프이므로 결국 모든 노드들은 두 집합 X와 Y로 분할된다. 이제 집합 X에 속한 정점들에는 동전을 바로 놓고, 집합 Y에 속한 정점들에는 동전을 뒤집어 놓으면 된다. □

일관성이 없는 그래프에서 위의 규칙을 만족시킬 수 없는 경우를 플립충돌(flip conflict)이라 하면, 우리의 관심사는 플립충돌이 최소화 되도록 동전을 놓는 것이다. 이것은 최소의 에지들만을 제거함으로써 그래프가 일관성을 가지도록 만드는 것과 같은데 이것은 NP-hard 문제이다. 모든 에지가 1로 주어지는 특별한 경우를 가정해 보면 이 문제는 최소의 에지를 제거함으로써 그래프를 이분 그래프(bipartite graph)로 변환하는 것과 동일한 문제이다. 이것은 '그래프 이분화 문제(the graph bipartization problem)'로 불리며 NP-hard로 알려져 있다.

본 논문에서 우리는 반복적인 최적화(iterative optimization)에 기반한 단순한 휴리스틱을 사용한다. 먼저 각 노드 v에 대칭표시자 $f_v \in \{0,1\}$를 할당한다. 만약 임의의 패치 P_v가 최종 맵에 대칭이동 되어 스티치 된다면 f_v는 1이 되고 그렇지 않으면 0이 된다. 최초 f_v의 값은 정의되지 않은 것으로 한다. 만약 이웃하는 두 패치 P_v와 P_u에 대해 $f_v \oplus f_u \oplus I_{uv} = 1$이 성립하면 이것은 플립충돌이 발생함을 의미한다.[1] 우리의 휴리스틱은 두 단계로 이루어진다. 첫 번째 단계에서는 초기 대칭표시자 값을 계산하는 단계이다. 두 번째 단계에서는 초기 할당값을 반복적으로 정제하는(refine) 단계이다.

첫 번째 단계는 다음과 같이 진행된다. 임의의 패치를 시작패치(seed patch)로 선택하고 그것의 대칭표시자를 0으로 고정한다. 그리고 나면 그것의 이웃 패치들에게 대칭표시자 값을 메시지로 전달한다. 수신자는 대칭표시자 값 f_v와 I_{uv}를 배타적-OR(exclusive-OR)한다. 즉, $f_u = f_v \oplus I_{uv}$를 계산한다. 일부 노드는 플립충돌을 수반하는 다수의 메시지를 수신할 수도 있다. 이런 경우에는 투표(vote)를 통해 값을 결정한다. 그림 4는 투표를 통해 대칭표시자 값을 결정하는 규칙을 설명한다. 회색 노드들은 검은색 노드의 대칭표시자가 0이 되어야 함을 요구하고, 반면 흰색 노드들은 1이 되어야 함을 요구한다. 투표에 의해서 검은 노드는 1로 결정을 내리게 된다.

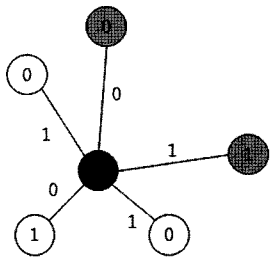


그림 4 대칭표시자의 결정

정제 단계는 사실 투표를 지속하는 과정이다. 각 노드는 이웃하는 노드와의 플립충돌이 최소화되도록 자신의 대칭표시자 값을 조정한다. 그리고 조정된 대칭표시자 값을 이웃에게 알린다. 이 과정은 전체 노드들에 걸쳐 병렬적으로(concurrently) 진행한다. 이 과정이 모든 경우에 반드시 종료될 것이라는 보장은 없다. 우리가 수행한 시뮬레이션에서는 대부분 3~4번의 반복 후 종료되었으며, 그렇지 않을 경우 반복의 최대 회수를 정해두고 이를 초과할 경우 강제종료하는 식으로 해결할 수 있다.

4.3 전역좌표의 계산

마지막 단계는 최종 전역맵에서의 각 노드의 좌표를 계산하는 것이다. 전역좌표를 계산하기 위해서 각각의 패치 P_v는 최종 변환 (R_v, t_v)를 계산해야 한다. 즉, 패치 P_v에서 지역좌표 x를 가지는 노드 v의 전역좌표는 $R_v x + t_v$가 된다.

최종 변환을 계산하는 기본 연산들은 다음과 같이 설명할 수 있다. 4.2절의 과정을 거쳐 모든 플립 충돌이 제거된 플립 그래프를 F'이라 하자. 패치 P_v에 대해 만약 k개의 이웃하는 패치 $P_1 \ldots P_k$가 F' 내에 존재하고 그 각각은 최종 변환이 이미 계산된 것으로 가정하자. R_i와 t_i가 각각 P_i의 최종 변환에서 회전이동과 평행이

1) $\oplus$는 Exclusive-OR를 나타낸다.

동을 나타내는 것으로 하고, P_i에 대한 P_v의 멤버 x의 최종 좌표를 x'라 하면 이것은 다음과 같이 나타낼 수 있다.

$$x' = R_i(R_{vi}x + t_{vi}) + t_i$$
$$= R_iR_{vi}x + (R_it_{vi} + t_i),$$

이러한 변환은 k개가 존재하므로 우리는 이들의 평균을 취한다. 따라서 최종적인 좌표는 다음과 같이 계산된다.

$$x' = \frac{1}{k}\sum_{i=1}^{k}(R_i(R_{vi}x + t_{vi}) + t_i)$$
$$= \frac{1}{k}\sum_{i=1}^{k}R_iR_{vi}x + \frac{1}{k}\sum_{i=1}^{k}(R_it_{vi} + t_i)$$

따라서 패치 P_v의 최종변환은 $(\frac{1}{k}\sum_{i+1}^{k}R_iR_{vi}, \frac{1}{k}\sum_{i+1}^{k}(R_it_{vi} + t_i))$가 된다.

최종 변환을 계산하기 위해서는 패치들의 부분순서(partial order)가 필요하다. 본 논문에서는 그래프 F' 내에서 시작패치로부터의 각 패치에 이르는 홉 카운트(hop count)를 사용하여 패치들에 대한 부분순서를 정의한다. 즉, 패치 P_v의 홉카운트가 인접한 패치P_u의 홉카운트보다 작은 경우에 P_v가 P_u보다 선행자(predecessor)가 된다. 잘 알려진 거리벡터(distance vector) 알고리즘을 사용하면 부분순서를 구할 수 있다.

이 과정은 시작노드(시작패치의 중심 노드)로부터 출발한다. 시작패치는 동치행렬(identity matrix)을 회전변환으로 사용하고 영벡터($\vec{0}$)를 평행이동으로 하는 변환을 최종변환으로 정의한다. 그리고 이 행렬과 벡터를 이웃하는 패치들에게 전달한다. 각각의 노드는 모든 선행 패치들이 최종 변환을 알려올 때까지 대기하다가 수신하는 즉시 자신의 최종 변환을 계산한 다음 그 결과를 이웃하는 노드들에게 전달한다. 불량 패치들은 자신의 최종적인 이동을 계산하지만 그 결과를 이웃 노드에 전달하지는 않음으로써 다른 패치들의 최종변환 계산에 영향을 끼치지 않는다.

4.4 시간복잡도

본 절에서는 이상에서 기술한 위치 인식 알고리즘의 수행 시간 복잡도를 고려한다. 우선 그래프의 최대 차수를 d라고 하자. 우선 패치를 구성하는 단계는 각각의 노드에 의해서 병렬적으로 수행된다. 각 노드는 최대 $O(d)$개의 자신의 지역 좌표를 계산해야 한다. 각각의 좌표는 멀티레터레이션을 통해서 계산되는데, 멀티레터레이션 연산의 경우 닫힌 해가 존재하지 않으므로 반복적 최적화 알고리즘을 적용한다. 가장 널리 사용되는 방법은 Newton 메서드를 적용하는 것이며, 이 경우 시간복잡도는 반복의 횟수에 의해서 정해진다. 만약 MDS기법으로 패치를 구축한다면 이 경우 $O(d^3)$의 시간이 필요하다.

다음으로 각 노드는 자신의 패치와 인접한 모든 패치와의 AOT를 계산해야 한다. 두 패치간의 AOT는 $O(d)$시간에 계산될 수 있다. 따라서 모든 AOT를 계산하는데 필요한 시간은 $O(d^2)$이다.

4.2절에서 기술한 플립충돌을 해결하는데 필요한 시간과 4.3절의 전역좌표 계산에 필요한 시간은 그래프의 지름(diameter)과 투표의 반복 횟수에 달려있다. 그래프의 지름을 m이라고 하고, 투표의 최대 반복횟수를 k라고 하면 $O(m+k)$ 시간이 소요된다.

5. 시뮬레이션을 통한 성능 평가

제안하는 알고리즘의 성능을 분석해 보기 위해 노드들이 영문자 C의 형태로 분포된 그림 5의 네트워 토폴로지를 사용하였다. 노드의 개수는 150개에서 400개까지 바꿔가며 실험을 수행하였다. 모든 노드는 동일한 통신반경(r)을 가지는 것으로 가정하며 거리가 $d(d \le r)$인 두개의 노드가 서로 통신이 가능할 확률은 $p = 1 - d^2/r^2$로 가정하였다. 또한 노드들 간의 거리 측정의 오차는 실제거리를 평균으로 하는 가우시안(Gaussian) 분포를 따르는 것으로 가정하였다.

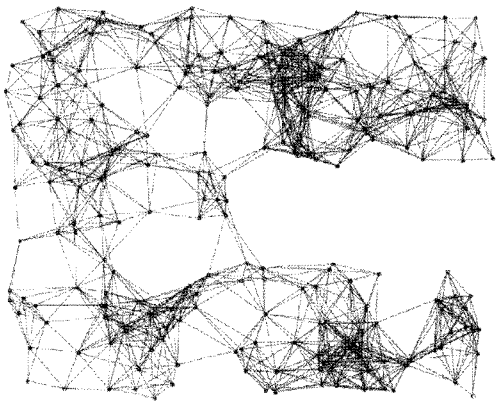


그림 5 성능 평가를 위한 네트워 구성의 예

앵커없는 위치추정에서는 추정된 좌표들은 임의의 상대좌표계상에서 정의되므로, 이를 직접 노드들의 실제 좌표와 비교하는 것은 무의미하다. 따라서 먼저 추정된 좌표들을 절대방향변환을 사용하여 노드들의 실제 좌표들과의 차이의 제곱의 합이 최소가 되도록 변환한 후, 위치 추정 오차를 계산하였다. 우리가 제안하는 알고리즘을 다음과 같은 세 종류의 버전으로 구분하여 테스트하였다.

- 알고리즘 A: 플립모호성 테스트와 플립충돌 테스트를 모두 사용하지 않음

- 알고리즘 B: 플립모호성 테스트를 사용하되 플립충돌 테스트는 사용하지 않음
- 알고리즘 C: 플립모호성 테스트와 플립충돌 테스트를 모두 사용함

또한 기존 연구와의 비교를 위하여 연구[4]와 [9]에서 제안한 그리디 스티칭(greedy stitching) 방법도 성능을 비교하였다. 그리디 스티칭 알고리즘은 각 단계에서 시작패치와 공통 노드를 가장 많이 갖는 패치부터 순서대로 시작 패치에 병합해 나가는 방법이다.

그림 6은 네트워크의 각 노드들에 대한 평균 차수가 증가함에 따라 위치 추정에 성공한 노드들의 비율을 나타낸 것이다. 위치추정에 성공한다는 것은 최종 전역 맵에 포함된다는 것을 의미한다. 네 가지의 알고리즘 중 어떤 것도 특별히 좋거나 나쁘지 않고 비슷한 성능을 나타내었다. 그림 7은 노드들의 평균 차수가 변화함에 따라 실제 좌표와 추정 좌표와의 오차율을 나타낸 것이다. 이 결과는 각각의 알고리즘 마다 상당히 큰 성능차이를 보였으며 제안하는 두 가지 필터링 기법 모두가 위치 추정 성능 향상에 도움을 줄 수 있음을 보여준다.

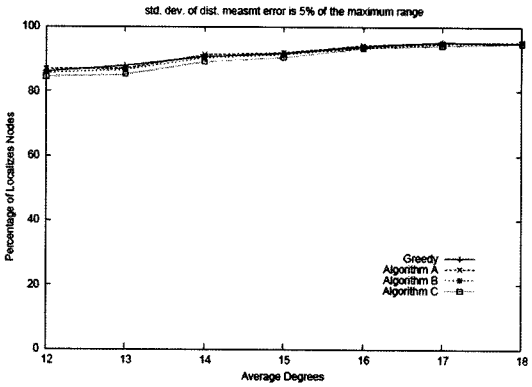


그림 6 노드 차수에 따른 위치추정 노드들의 비율 ($\delta=1/8$ 인 경우)

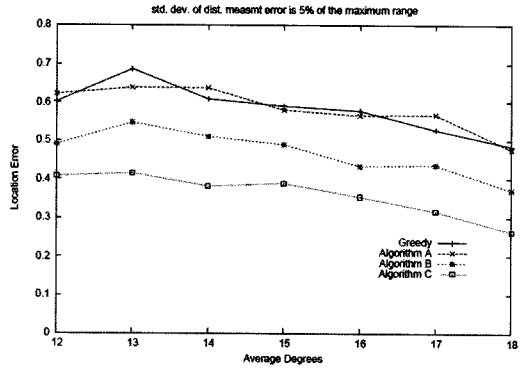


그림 7 노드의 차수에 따른 위치추정 오차율

6. 결 론

패치 스티치 전략에 기반 한 위치추정 알고리즘은 잘못된 대칭이동에 의해 추정 오차가 매우 크게 되는 일이 흔히 발생한다. 본 논문에서는 대칭이동의 오류를 최소화하기 위해 플립모호성 테스트와 플립충돌 테스트라는 두 가지 기법을 제안한다. 플립모호성 테스트는 패치를 스티칭하는 과정에서 대칭이동의 유무에 따라 발생하는 스티치 오차의 비율을 비교하여 일정 수준 이상의 값이 확보되는 경우에만 패치를 스티칭한다. 플립충돌 테스트는 스티칭 과정에서 대칭이동 유무를 전체적으로 파악하여 모순되는 대칭이동의 수가 가장 적게 일어나는 방향으로 패치-스티치를 수행하는 것이다. 시뮬레이션을 통해 언급된 테스트 기법을 적용하는 경우 기존의 위치추정 알고리즘 보다 더 작은 오차를 가지는 위치추정이 가능함을 보였다.

참 고 문 헌

[1] J. Bachrach and C. Taylor, "Localization in sensor networks," Handbook of Sensor Networks: Algorithms and Architectures, Wiley Pub., 2005.

[2] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," Telecommunication Systems, Vol. 22:1-4, pp. 267-280, 2003.

[3] Andreas Savvides, Chih-Chieh Han, and Mani B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," MobiCom 2001, Rome, Italy, July 2001.

[4] X. Ji and H. Zha, "Sensor positioning in wireless ad hoc networks using multidimensional scaling." Infocom, 2004.

[5] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," ACM Sensys-04, Nov 2004.

[6] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor free distributed localization in sensor networks," MIT Lab. for Computer Science, TR No. 892, April 15, 2003.

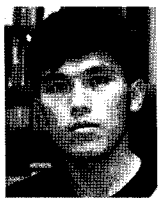
[7] C. Savarese, J.M. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," ICASSP, May 2001.

[8] L. Meertens and S. Fitzpatrick, "The distributed construction of a global coordinate system in a network of static computational nodes from inter-node distances," Kestrel Institute TR KES.U.04.04, 2004.

[9] Y. Shang and W. Ruml, "Improved MDS-based localization," IEEE INFORCOM, 2004.

[10] B. K. P. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," Journal of the Optical Society of America A, 5(7), 1988.

[11] T. Cox and M. Cox, Multidimensional scaling, 2nd Edition, Chapman & Hall, 2001.



권 오 흠

1988년 서울대학교 컴퓨터공학과 졸업(공학사). 1991년 KAIST 전산학과(공학석사). 1996년 KAIST 전산학과(공학박사). 현재 부경대학교 전자컴퓨터정보통신공학부 교수. 관심분야는 알고리즘 설계 및 분석, 분산 컴퓨팅, 유비쿼터스 센서 네트워크 등

박 상 준

1988년 경북대학교 전자공학과 졸업(학사). 1990년 경북대학교 전자공학과 졸업(공학석사). 2006년 North Carolina State University(공학박사). 현재 ETRI 감시정찰센서네트워크 연구팀 팀장. 관심분야는 무선 센서 네트워크

송 하 주

1993년 서울대학교 컴퓨터공학과 졸업(공학사). 1995년 서울대학교 컴퓨터공학과 졸업(공학석사). 2001년 서울대학교 컴퓨터공학과 졸업(공학박사). 2003년 ㈜아이티포웹 부장. 현재 부경대학교 전자컴퓨터정보통신공학부 조교수. 관심분야는 RFID 미들웨어, 데이타베이스, 웹서비스