Journal of Korea Multimedia Society Vol. 9, No. 2, February 2006(pp. 172-188)

무선 센서 네트워크에서의 2단계 위치 추정 알고리즘

권오흠[†], 송하주[††], 김숙연[†††]

요 약

본 논문은 무선 센서 네트워크(wireless sensor network)에서 노드들의 위치 추정 문제를 다룬다. 위치 추정을 위한 기존의 기법들은 인접한 노드들 간의 거리를 이용하여 자신의 위치를 추정하는 LGB 기법과 멀리 떨어진 노드들 간의 거리를 추정하여 위치 추정에 이용하는 GGB 기법으로 분류할 수 있다. 그러나 LGB 기법의 경우 상대적으로 정확도가 낮은 거리 측정 기법 하에서는 적합하지 않은 면이 있고, 반면 GGB 기법의 경우 장애물이 있거나 혹은 노드들이 분포한 영역이 사각형이나 원형과 같은 정형이 아닌 경우에는 적용하기 어려운 면이 있다. 본 논문에서는 두 가지 기법을 절충하여 부정확한 거리 측정 기법 하에서 장애물이 있거나 비정형의 영역 내에 분포한 노드들에 대해서도 위치 추정을 가능하게 하는 새로운 위치 추정 알고리즘을 제안하고, 그 성능을 모의실험을 통해서 비교 분석한다.

Two-Phase Localization Algorithm in Wireless Sensor Networks

Oh-Heum Kwon[†], Ha-Ju Song[††], Sook-Yeon Kim[†††]

ABSTRACT

Sensor localization is one of the fundamental problems in wireless sensor networks. Previous localization algorithms can be classified into two categories, the GGB (Global Geometry-Based) approaches and the LGB (Local Geometry-Based). In the GGB approaches, there are a fixed set of reference nodes of which the coordinates are pre-determined. Other nodes determine their positions based on the distances from the fixed reference nodes. In the LGB approaches, meanwhile, the reference node set is not fixed, but grows up dynamically. Most GGB algorithms assume that the nodes are deployed in a convex shape area. They fail if either nodes are in a concave shape area or there are obstacles that block the communications between nodes. Meanwhile, the LGB approach is vulnerable to the errors in the distance estimations. In this paper, we propose new localization algorithms to cope with those two limits. The key technique employed in our algorithms is to determine, in a fully distributed fashion, if a node is in the line-of-sight from another. Based on the technique, we present two localization algorithms, one for anchor-based, another for anchor-free localization, and compare them with the previous algorithms.

Key words: Wireless sensor network (WSN)(무선 센서 네트워크), Localization(위치 결정), Location Identification(위치 추정)

1. 서 론

무선 센서 네트워크는 군사적, 상업적 분야를 비롯한 다양한 분야에서 그 응용 범위가 빠르게 확대되고 있는 중요한 연구 분야의 하나이다. 무선 센서 네트워크의 기본 개념은 수백 혹은 수천 개에 이르는

※ 교신저자(Corresponding Author) : 권오흠, 주소 : 부산광역시 남구 대연 3동 599-1(608-737), 전화 : 051)620-6886, FAX : 051)620-6450, E-mail : ohkwn@pknu.ac.kr
접수일 : 2005년 5월 11일, 완료일 : 2005년 10월 20일
[†] 정회원, 부경대학교 전자컴퓨터정보통신공학부
[††] 정회원, 부경대학교 전자컴퓨터정보통신공학부
(E-mail : hajusong@pknu.ac.kr)
[†††] 준회원, 한경대학교 컴퓨터공학과
(E-mail : entpentp@paran.com)

작고 값이 싼 센서 노드(node)들을 일정한 지역에 분산 배치한 후, 각 센서가 자신의 주변 환경을 감지하고, 감지된 데이터를 처리하며, 그 데이터를 중앙 시스템에 전송하는 것이다. 센서 노드들은 미리 정해진 위치에 계획적으로 배치되기 보다는 차량이나 비행기 등을 통하여 배포되어 그 위치가 무작위로 정해지는 경우가 많다. 반면 센서 노드에 의해서 감지된 데이터는 그 데이터가 감지된 위치가 어디인지를 알아야만 의미가 있을 것이다. 따라서 무작위로 살포된 센서 노드들 각각의 위치를 알아내는 것은 무선 센서 네트워크에서 해결되어야 하는 가장 기본적인 문제 중의 하나이다.

가장 간단한 해결책은 각 센서노드에게 GPS 수신기를 부착하는 것이다. 그러나 일반적으로 GPS 수신기는 부피가 크고 비싸며 또한 많은 전력을 소비하기 때문에 많은 수의 값이 싼 센서 노드들을 이용하는 무선 센서 네트워크에 적용하기에는 부적절한 경우가 많다. 또한 건물 내부라든가 동굴 속 등과 같이 GPS 수신이 불가능한 경우도 있다. 그래서 GPS와 같은 비싼 추가 장치 없이 노드들 간의 상대적인 거리 정보를 이용하여 노드들의 위치를 추정하는 방법에 대한 연구가 꾸준하게 이루어져 왔다.

노드들 간의 거리를 측정 혹은 추정하는 다양한 기법들이 존재한다. RSSI (Received Signal Strength Indicator) 기법은 일반적으로 신호의 강도는 거리의 제곱에 반비례하는 특성을 이용하여 수신된 신호의 강도를 측정하여 신호를 전송한 노드로부터의 거리를 추정하는 기법이다. 이 방법의 장점은 추가적인 하드웨어가 필요치 않으므로 비용이 저렴하다는 것이며, 단점은 비교적 거리 측정의 오차가 크다는 점이다.

ToA (Time of Arrival)와 TDoA (Time Difference of Arrival) 기법은 RSSI 에 비해서 훨씬 더 정확한 거리 측정이 가능하지만 훨씬 강력한 계산능력을 가진 CPU를 필요로 하므로 비용의 측면에서 단점이 있다.

RSSI나 ToA 등의 기법처럼 인접한 두 노드간의 거리를 직접 측정하는 대신 두 노드 간에 교신이 가능한지 아닌지 만을 기준으로 대략적인 거리를 추정하는 기법을 사용하는 경우가 있는데 이를 근접성(proximity)에 기반 한 기법이라고 부른다[1,3-6]. 근접성에 기반 한 기법의 경우 주로 멀리 떨어진 노드들 간에 두 노드가 몇 개의 경유 노드들 통하여 서로 연결되는지를 알아냄으로써 대략적인 거리를 추정하는데 사용된다.

또한 거리가 아닌 신호가 도착하는 방향을 측정하는 기법으로 AoA (Angle of Arrival) 기법이 있다. AOA 기법을 적용하기 위해서는 각 노드가 안테나 어레이(antenna array) 혹은 초음파 수신기(ultrasound receiver)를 장착해야만 한다.

2. 관련 연구

노드간의 측정 혹은 추정된 거리 정보를 이용하여 노드들의 위치를 추정하는 다양한 알고리즘들이 제시되어 있는데, 대부분의 알고리즘들은 대체로 다음과 같은 세 단계의 구조를 가지고 있다.

단계 1: 기준 좌표계의 설정
단계 2: 노드들의 위치 추정
단계 3: 추정된 위치의 수정

우선 노드들의 위치를 결정하기 위해서는 단일한 좌표계가 설정되어야 한다. 이 좌표계는 외부적으로 주어질 수도 있으나, 노드들 스스로 설정할 수도 있다. 좌표계가 설정되면 각 노드들은 서로 간의 거리 정보를 이용하여 설정된 좌표계 상에서 자신의 위치를 추정한다. 마지막으로 추정된 위치를 인접 노드들 간의 관계를 고려하여 보다 정밀하게 수정하는 단계를 거치기도 한다.

2.1 좌표계 설정 방법의 분류

노드들의 위치를 결정하기 위해서는 노드들의 위치를 정의하기 위한 하나의 통일된 좌표계가 존재해야 한다. 좌표계는 외부적으로 주어질 수도 있고, 노드들 스스로 설정할 수도 있다. 외부적으로 주어지는 좌표계를 절대(absolute) 좌표계라고 부르고, 노드들 스스로 외부의 개입 없이 설정하는 좌표계를 상대(relative) 좌표계라고 부른다.

가. 절대 좌표계 혹은 앵커 노드에 기반 한 기법

대 좌표계는 일반적으로 몇 개의 노드들의 좌표를 외부적으로 미리 지정함으로써 설정될 수 있다. 이때 좌표가 미리 지정된 노드들을 일반적으로 앵커

(anchor) 노드라고 부르며, 그런 의미에서 이런 방법을 앵커 노드에 기반 한(anchor-based) 기법이라고도 부른다. 앵커 노드들은 GPS 수신기를 부착하고 있을 수도 있고, 혹은 수작업으로 자신의 좌표를 입력 받을 수도 있다. 동일 직선상에 놓여있지 않는 세 개의 앵커 노드가 존재하면 다른 노드들의 좌표는 앵커 노드들에 대한 자신의 상대적 위치를 파악함으로써 결정될 수 있다.

앵커 노드들은 계획적으로 특정한 위치에 배치될 수도 있고[2], 무작위의 위치에 비계획적으로 배치될 수도 있다[5,6]. 앵커 노드의 수가 충분히 많거나 혹은 앵커 노드의 교신 가능 범위가 일반 노드에 비해서 충분히 멀어서 모든 노드들이 3개 이상의 앵커 노드와 직접 교신 가능하다고 가정하는 경우도 있고[4], 아주 작은 수, 가령 3~4개의 앵커노드만이 존재하기 때문에 일반적으로 앵커 노드와 교신하기 위해서는 다른 노드들을 경유해야한다고 가정하는 경우도 있다[5,6]. 고정되어 있지 않고 움직이면서 시시각각 자신의 위치를 다른 노드들에게 전송해주는 모바일(mobile) 앵커를 가정하는 경우도 있다.

앵커 노드의 존재는 위치 추정 문제를 비교적 쉽게 만든다. 그러나 앵커 노드의 존재는 그 자체로 비용의 증가를 의미하여, 경우에 따라서 GPS 수신이 불가능하거나 혹은 수작업이 가능하지 않은 경우도 있으므로 적용에 있어서 한계가 있다.

나. 상대 좌표계 혹은 앵커-프리(anchor-free) 기법

이 기법은 앵커 노드의 존재를 가정하지 않는 기법이다. 앵커 노드가 존재하지 않는 경우에는 노드들의 절대적 좌표를 알아내는 것은 불가능하다. 그러나 노드들 간의 상대적인 위치관계를 표현하는 상대적 맵(map)을 구성하는 것은 가능하다. 경우에 따라서는 상대적 맵을 구성하는 것으로 충분한 경우도 있고, 또한 간단한 후 작업을 통해서 상대적 맵을 절대적 맵으로 변환하는 것도 가능하므로 앵커 없이 상대적 맵을 구하는 문제에 대한 연구가 있어 왔다.

앵커-프리 기법에서 좌표계를 설정하는 기법은 크게 두 가지로 분류할 수 있다. 첫째는 노드들 스스로 몇 개의 노드를 선정한 후 그들에게 기준점의 역할을 하는 좌표를 부여하는 방법이다. 이때 이 노드들을 코어 노드(core node)라고 부른다. 가령 가장 간단한 방법은 삼각형을 형성하는 세 노드를 선택하여 그 중의 하나는 원점의 역할을 하고, 다른 하나는 x -축 상에 있다고 가정하고, 나머지 한점의 위치를 1-사분면이나 2-사분면상에서 정의하면 하나의 좌표계가 정의된다[12]. 반면 [7]에서는 노드들이 분포하는 영역의 상하좌우 극단에 존재하는 네 노드를 찾아서 이들이 코어 노드의 역할을 하도록 한다.

두 번째 방법은 코어 노드를 정하지 않고 모든 혹은 여러 개의 노드들이 스스로를 중심으로 하는 독립적인 좌표계를 수립하고, 자기 자신 및 자신과 인접한 노드들로 구성되는 지역 맵(local map)을 구성하는 것이다. 각각의 지역 맵은 서로 다른 좌표계상에서 정의되는 셈이다. 두 지역 맵이 중첩될 경우 두 지역 맵에 공통으로 속한 노드들의 위치 정보를 이용하여 두 맵을 하나로 합병(merge)한다. 이런 작업은 네트워크 전체에서 동시에 일어나며, 최종적으로 모든 지역 맵이 하나로 합병됨으로써 위치 추정이 완료된다. 이런 기법을 병렬(concurrent) 위치 추정 기법이라고 부른다.

2.2 위치 추정 방법에 따른 분류

노드들 간의 거리 정보를 이용하여 노드의 위치를 추정하는 대부분의 알고리즘에 공통적인 기본 동작의 하나가 멀티레터레이션(multilateration)이다. 각 노드는 이미 자신의 위치를 알고 있거나 혹은 위치 추정을 완료한 몇 개의 노드들로부터 그들의 위치 정보를 전달받고 또한 그들과 자신간의 거리를 측정 혹은 추정한다. 이때 이미 자신의 위치를 알고 있는 이러한 노드들을 참조 노드(reference node)라고 부른다.

일반적으로 평면상에서 거리 정보를 이용하여 한 노드의 위치를 결정하기 위해서는 동일 직선상에 놓여 있지 않는 3개 이상의 참조 노드가 필요하다. 만약 참조노드들의 위치와 그들로부터의 거리가 오류 없이 정확하다면 노드의 위치는 평면상의 한 점으로 유일하게 정해진다. 그러나 일반적으로 거리 측정에는 오차가 있으므로 한 점으로 유일하게 정해지지는 않는다. 이 경우 추정된 위치와 각 참조 노드간의 거리를 계산하여, 그렇게 계산된 거리와 추정된 거리의 차이의 제곱의 합을 최소로 하는 지점을 선택한다. 가령 노드 v가 k개의 참조 노드의 위치 $(x_i, y_i), 1 \le i \le k$, 를 알고 있고, 각각의 참조 노드로부터의 거리 d_i,

$1 \le i \le k$,를 추정했다고 하자. 노드 v가 자신의 위치를 (x,y)라고 가정한다면 자신과 각 참조 노드와의 거리는 $calc_dist(v,v_i) = \sqrt{(x_i-x)^2+(y_i-y)^2}$가 된다. 이렇게 계산된 거리와 추정된 거리 $est_dist(v,v_i) = d_i$와의 차이의 제곱을 모든 참조노드에 대해서 합한 값, 즉

$$\sum_{i=1}^{k}(calc\_dist(v,v_i) - est\_dist(v,v_i))^2$$

을 최소로 만드는 위치 (x,y)를 찾음으로써 자신의 위치를 결정하는 방법을 멀티레터레이션이라고 부른다. 상세한 계산 과정은 문헌 [5]를 참조하라.

만약 참조 노드가 2개이거나 혹은 3개 이상이라 하더라도 모두 일직선상에 놓여 있다면 참조 노드까지의 거리에 관한 조건을 만족하는 지점은 2개가 된다. 두 지점은 참조 노드들을 연결하는 직선을 기준으로 서로 대칭인 위치에 있을 것이다. 위치 추정 알고리즘이 이런 상황에서 실제 위치의 반대쪽을 잘못 선택하는 것을 플립(flip) 오류라고 부른다.

일반적으로 참조 노드의 위치와 그들과의 거리가 정확한 것이 아니므로 실제로는 참조 노드가 3개 이상이고 그들이 일직선상에 있지 않은 경우에도 역시 플립 오류가 발생할 수 있다. 만약 세 개의 참조 노드 a,b,c가 그림 1과 같이 일직선에 가깝게 놓여 있다면 거리 추정의 오차와 참조 노드 위치의 오차에 의해서

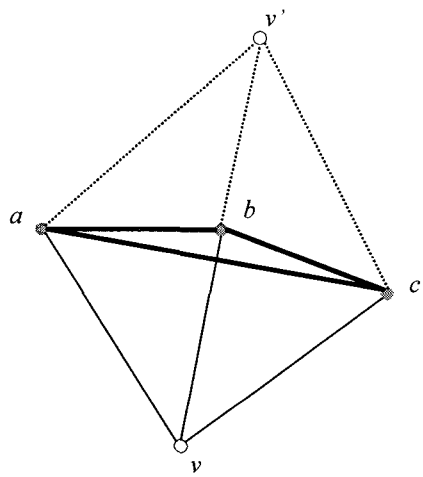


그림 1. 플립 오류의 발생

실제 위치 v와는 반대쪽의 위치 v'를 선택할 수 있다. 플립 오류는 일반적으로 거리 측정 오차의 점진적인 누적에 따라 발생하는 위치 추정의 오차에 비해서 한번에 매우 큰 위치 추정 오차를 발생시키기 때문에 플립 오류의 발생을 방지하는 것은 위치 추정 알고리즘에 있어서 가장 중요한 문제의 하나라고 할 수 있다.

멀티레터레이션 기법을 사용하는 위치 추정 알고리즘들은 각각의 노드가 자신의 위치를 추정할 때 어떤 노드들을 참조 노드로 이용하는가에 따라 GGB (Global Geometry based) 기법과 LGB (Local geometry based) 기법으로 분류할 수 있다.

가. GGB 기법

일반적으로 GGB 기법은 앵커 노드 혹은 코어 노드를 이용한 좌표계 설정 기법과 함께 사용되는 경우가 많다. 앵커 노드 혹은 코어 노드들이 다른 노드들의 위치 추정에 있어서 참조 노드의 역할을 수행하며, 모든 노드들은 이들로부터의 거리 정보를 이용하여 자신의 위치를 추정한다. 이때 위치를 추정할 노드와 참조 노드들은 일반적으로 인접하지 않고 여러 홉(hop) 떨어진 노드들이므로 그런 의미에서 "원 거리(multi-hop distance) 추정"에 기반 한 기법이라고 불리기도 하며, 또한 모든 노드들이 고정된 참조 노드를 이용한다는 의미에서 "고정된(fixed) 참조 노드"를 이용하는 기법이라고 부를 수도 있다.

이 기법의 장점은 각각의 노드가 고정된 참조 노드와의 관계에 의해서 독립적으로 자신의 위치를 추정하므로 설사 플립 오류가 발생하더라도 이것이 다른 노드의 위치 추정에 연쇄적인 영향을 주지 않는다는 것이다. 또한 각 부분 부분에서의 위치 추정의 정밀도는 떨어지지만 전반적인 노드의 분포 형태와 각 노드의 대략적인 위치를 비교적 쉽게 찾을 수 있다는 장점이 있다.

이 방법에서는 위치 추정을 위해서 멀리 떨어진 노드들 간의 거리를 이용한다. 따라서 RSSI나 TOA와 같은 물리적인 거리 측정 방법으로 참조 노드와 자신간의 거리를 직접 측정할 수 없고, 참조 노드와 자신을 연결하는 경로(path)의 홉 수(hop count), 혹은 최단 경로의 길이 등에 의해서 간접적으로 거리를 추정하여야 한다. 이는 추정되는 거리의 오차가 클 수 있다는 것을 의미한다. 또한 홉 수 혹은 최단 경로의 길이 등으로 거리를 측정하기 때문에 노드들이

분포하는 영역이 원형이나 사각형 등과 같이 정형(regular shape)이 아닌 경우, 노드들이 분포한 영역 내에 장애물이 있는 경우, 혹은 노드들의 분포가 심하게 불균등할 경우에는 적용하기 어려운 단점이 있다. 가령 그림 2와 같이 두 노드 a와 v 사이에 장애물이 있는 경우 두 노드를 연결하는 최단 경로는 장애물을 피해서 돌아가는 경로이므로 두 노드간의 실제 거리보다 훨씬 길다. 따라서 만약 a를 참조 노드로 하고 최단 경로의 길이를 이용하여 노드 v의 위치를 추정할 경우 실제 위치에서 완전히 벗어날 수 있다.

GGB 기법은 노드들의 대략적 위치를 쉽게 추정할 수 있으나 부분적인 정확도는 떨어진다. 위치 추정의 정밀도를 높이기 위해서 추정된 위치에 대한 반복적인 수정(iterative refinement) 단계를 거치기도 한다[7,11,13].

나. LGB 기법

LGB 기법은 각 노드가 자신과 인접한 노드들 중에서 이미 위치가 결정된 노드들을 참조 노드로 사용하는 기법이다. 이 방법에서는 자신과 직접 이웃한 노드와의 거리를 사용하므로 거리 측정에 있어서의 오차가 상대적으로 작고, 또한 이 방법은 국부적인 노드 배치에 의해서 자신의 위치를 추정하므로 영역의 형태라든가 장애물의 존재 유무가 별 문제가 되지 않는다는 장점이 있다.

반면 이 방법의 단점은 첫째, 거리 측정의 오차가 계속해서 누적되며, 둘째, 멀티레터레이션을 할 때

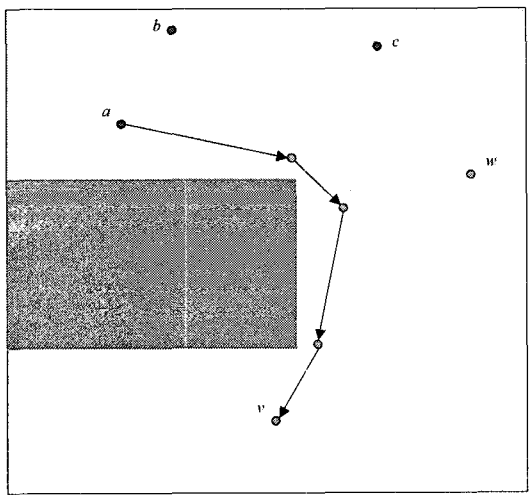


그림 2. 장애물이 있는 경우

참조 노드의 위치 자체가 정확한 위치가 아닌 추정된 위치이므로 그 만큼 플립 오류가 발생할 가능성이 높고, 셋째, 추정된 노드의 위치가 다른 노드의 위치 추정에 이용되므로 한번 플립 오류가 발생할 경우 다른 노드의 위치 추정에 연쇄적으로 치명적인 악영향을 끼칠 수 있다는 것이다. 따라서 이 기법에서는 어떻게 플립 오류가 발생할 가능성을 줄이는가 하는 것이 가장 핵심적인 문제가 된다.

현재까지 제안된 LGB 기법에는 몇 가지 형태가 있다. 우선 서로 근거리에 놓여있는 몇 개의 노드들을 코어 노드로 사용하는 좌표 설정 기법과 결합하면 전형적인 점진적(incremental) 알고리즘이 된다. 즉, 코어 노드들에서 출발하여 세 개 이상의 코어 노드와 인접한 노드는 그들을 참조 노드로 하여 자신의 위치를 추정하고, 그러고 나면 스스로 다른 노드들에 대해서 참조 노드의 역할을 수행한다[3].

이 방법은 매우 단순하다는 장점이 있으나 실제로는 플립 오류의 빈번한 발생에 의해서 위치 추정의 정확도가 떨어진다고 알려져 있다. 이 방법에서 플립 오류의 발생을 억제하기 위해서 참조 노드의 위치 및 그들과의 거리에 관한 특정한 조건을 검사하여 만족할 경우에만 멀티레터레이션을 하도록 알고리즘을 수정할 수 있다. 그러나 그 경우 조건을 엄격하게 할수록 위치 추정의 성공률이 낮아지는, 즉 조건을 만족하는 참조 노드들을 확보하지 못하여 아예 위치를 추정하지 못하는 노드의 비율이 높아지는 단점이 발생한다.

LGB 기법에서의 이런 문제점을 극복하기 위하여 제시되는 방법이 병렬 위치 추정 기법이다. [15]에서는 서로 인접한 네 노드가 특정한 조건을 만족할 경우에 이를 robust quadrilateral이라고 부르고, 모든 노드는 병렬적으로 자신이 소속하는 robust quadrilateral을 찾는다. 두 robust quadrilateral이 세 노드를 공유할 경우 하나로 합병된다. 이런 과정을 거쳐서 모든 robust quadrilateral이 합병되면 위치 추정이 완료된다. 반면 [14]에서는 각각의 노드는 자신의 인접 노드들과 자기 자신을 포함하는 지역 맵을 MDS (Multi-dimensional Scaling) 기법을 이용하여 구축하며, 각각의 지역 맵들은 인접한 지역 맵들과 합병되어 결국 모든 지역 맵이 하나로 합병되는 순간 위치 추정이 완료된다.

병렬 위치 추정 기법의 단점은 지역 맵을 통합하

기 위해서는 관련된 노드들 간에 비교적 복잡한 통신 과정을 거쳐야하며, 따라서 전체적으로 알고리즘의 복잡도가 증가한다는 것이다. 또한 지역 맵들 간의 합병이 결국 각 맵의 가장자리에 위치한 몇몇 노드의 좌표에 의해서 결정될 수 있으므로 플립 오류의 가능성이 여전히 남아 있다는 것이다.

2.3 연구의 목표 및 결과

본 논문의 연구 목표는 기본적으로 다음과 같은 두 가지 요구사항을 만족하는 위치 추정 알고리즘을 개발하는 것이다.

부정확한 거리 측정 기법 하에서의 견고성

일반적으로 거리 측정에 있어서 그 정밀도가 높을수록 고비용의 추가적인 하드웨어 장치가 필요하다고 할 수 있다. 본 논문의 연구 목표는 거리 측정의 정밀도를 높이는 것 보다는 저 비용의 상대적으로 부정확한 거리 측정기법 하에서도 적용될 수 있는 견고성(robustness)을 가진 위치 추정 기법을 개발하는 것이다.

노드 분포의 비 균일성 및 영역의 비정형성

현재까지 개발된 위치 추정 기법들은 노드들이 분포한 영역이 정사각형이나 원형 등의 정형적인 형태라고 가정하는 경우가 많다. 노드가 분포한 영역 자체가 비정형이거나 혹은 영역 내에 장애물이 있는 경우 등의 상황에서는 적용하기 어려운 경우가 많다. 또한 영역 자체가 정사각형이나 원형의 형태라고 하더라도 노드 밀도가 낮을 경우에는 부분적으로 노드들이 없거나 아주 희소한(sparse) 영역이 존재하게 되며, 그런 경우 장애물이 있는 것과 유사한 상황이 된다. 본 논문에서는 그러한 상황에서 적절히 적응할 수 있는 위치 추정 기법을 개발하는 것을 목표로 한다.

부정확한 거리 측정 기법은 LGB 기법의 적용을 어렵게 만드는 요소이다. 현재까지 알려진 LGB 기법의 경우 거리 측정의 정확성이 떨어지는 경우에는 매우 빈번한 플립 오류가 발생하여 위치 추정의 오차가 허용할 수 있는 범위를 넘어서거나, 혹은 위치 추정의 성공률이 매우 떨어진다. 반면 영역의 비정형성이나 장애물의 존재는 GGB 기법의 적용을 어렵게 만드는 요소이다. 대부분의 GGB 기법들은 멀리 떨어진 노드들 간의 경로의 길이를 이용하여 거리를 추정하는데, 장애물이 있는 경우 경로의 길이와 실제 거리의 연관성이 줄어들기 때문이다. 그런 의미에서 위의 두 가지 요구사항은 서로 모순되는 측면이 있다고 할 수 있다.

본 논문에서 제시하는 기법은 위의 두 가지 요구사항을 동시에 충족하기 위해서 기본적으로 GGB 기법에 기반 하면서 LGB 기법을 부분적으로 절충한 형태라고 할 수 있다. 가령 그림 3의 (a)와 같은 상황을 고려해 보자. 노드들이 분포하고 있는 영역의 오른쪽-가운데 부분에 건물 등의 장애물이 있어서 결국 노드들이 분포한 영역이 ㄷ-자 형상을 이루고 있다. 그림에서 검은색 큰 노드들이 앵커 노드들이다. 위쪽 부분에 2개의 앵커 노드 a, b가 있고, 아래쪽 부분에 2개의 앵커 노드 c, d가 있다.

이런 상황에서 앵커 노드들을 참조 노드로 하고, 앵커와 자신간의 최단 경로의 길이를 거리로 추정하여 GGB 기법을 적용할 경우, 그림에서 회색으로 표시된 노드들은 네 개의 앵커 노드 중 적어도 세 개 이상의 앵커와 자신을 연결하는 직선이 장애물과 만나지 않는다. 가령 그림에서 앵커 a, b, c 각각을 노드 e와 연결하는 직선은 장애물을 만나지 않는다. 이 경우 앵커 a, b, c는 노드 e에 대해서 유효한 앵커라고 부른다. 세 개 이상의 유효한 앵커를 가진 회색 노드들은 유효한 앵커들과 자신간의 거리를 추정한 후 그들을 참조 노드로 하여 자신의 위치를 추정하면 된다. 여기서 각 노드들이 어떻게 각 앵커의 유효성을

표 1. 두 가지 기준에 따른 위치 추정 알고리즘의 분류

		좌표 설정 방법에 따른 분류		
		앵커 기반	코어 노드 기반	병렬 기법
위치추정 방법에 따른 분류	LGB	APIT [4]	Incremental [3]	Robust Quadrilateral [15] Distributed MDS [14]
	GGB	DV-hop [5] DV-distance [5]	Anchor-free [7]	

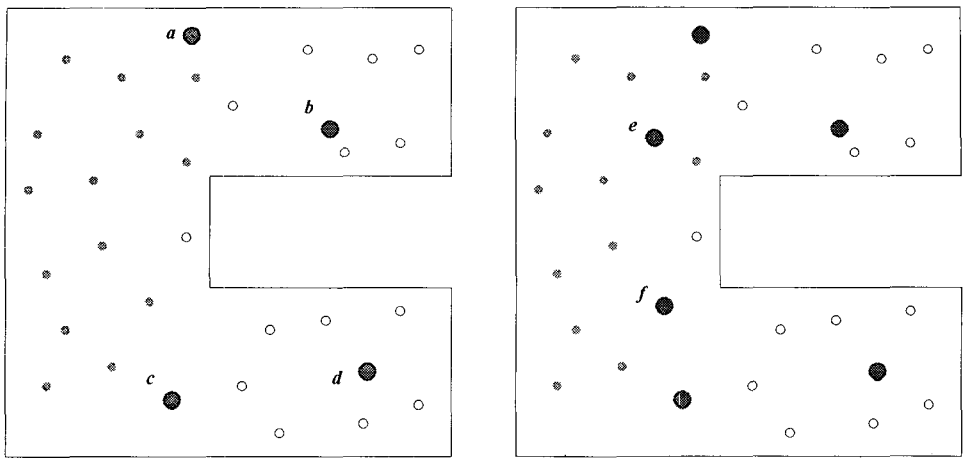


그림 3. 2단계 위치 추정 기법

판단하는가의 문제는 일단 나중으로 미루도록 하자.

더 근본적인 문제는 회색 노드들을 제외한 나머지 노드들은 그런 조건을 만족하는 세 앵커 노드가 존재하지 않는다는 점이다. 따라서 앵커 노드들만을 참조 노드로 하여 위치를 추정하게 되면 올바른 위치를 추정할 수 없다. 즉, 앵커 이외의 다른 참조 노드를 필요로 한다.

따라서 한번에 모든 노드의 위치를 추정하는 대신, 다단계를 거쳐서 위치를 추정하는 접근법이 요구된다. 즉, 그림의 예에서 3개 이상의 유효한 앵커를 확보한 회색 노드들이 일차적으로 자신의 위치를 추정한다. 그런 다음 위치를 추정한 노드들 중에서 일부의 노드를 선택하여 그들이 나머지 노드들의 위치 추정에 대한 참조 노드의 역할을 수행하는 것이다. 그림 (b)에서 노드 e와 f가 그런 역할을 하는 노드들의 예이다. 예를 들어서 노드 f는 앵커 노드 a, c, 그리고 d를 참조 노드로 하여 자신의 위치를 추정하였다. 그런 다음 f는 다른 노드, 가령 노드 g의 위치 추정에 있어서 참조 노드의 역할을 하게 된다. 이렇게 앵커 노드가 아니면서 다른 노드들의 위치 추정에 있어서 참조 노드의 역할을 하는 노드들을 가상 앵커(virtual anchor) 노드라고 부른다.

이러한 접근법은 고정된 수의 노드들이 모든 노드의 위치 추정에 있어서 참조 노드의 역할을 하는 GGB 기법과 위치 추정이 완료된 모든 노드들이 인접한 다른 노드의 위치 추정에 있어서 참조 노드의 역할을 하는 LGB 기법의 중간적인 형태라고 할 수 있다.

위의 예에서는 두 단계에 걸쳐서 노드들의 위치가 추정되었으나, 이를 보다 많은 단계로 일반화하는 것도 가능하다. 앵커 노드의 수가 상대적으로 희박하고 영역 형태의 복잡도가 높으면 2단계 이상의 여러 단계로 나누는 것이 필요해진다. 다만 단계가 늘어날수록 가상 앵커 노드의 위치의 정확성을 보장하기가 힘들어진다. 우리가 실시한 모의실험에 의하면 3단계 이상의 다단계로 위치 추정을 실시하는 것은 바람직하지 않은 것으로 나타났다.

이러한 접근이 성공적으로 동작하기 위해서는 다음과 같은 두 가지 문제가 해결되어야 한다. 첫째는 각각의 노드가 어떤 앵커가 장애물에 의해서 가려져 있으며 어떤 앵커가 가려져 있지 않은지를 판단할 수 있는 방법이 필요하다. 이 문제는 본 논문의 4장에서 다루어진다. 둘째로 가상 앵커 노드는 다른 노드의 위치 추정에 있어서 참조 노드의 역할을 수행하므로, 가상 앵커 노드의 위치 추정 오류는 다른 노드의 위치 추정에 심각한 악영향을 미친다. 따라서 가상 앵커 노드는 매우 신중하게 선택되어야 한다. 가상 앵커를 선택하는 구체적인 방법은 5장에서 다루어진다.

본 논문에서 제시하는 위치 추정 알고리즘은 다음과 같은 가정 하에서 설계되었다. 첫째, 모든 노드들은 자신만의 고유한 ID를 가진다. 둘째, 각 노드는 RSSI 혹은 ToA 등의 거리 추정 기법을 사용하여 인접한 노드와의 거리를 추정할 수 있다. 셋째, 각 노드의 최대 교신 가능 범위는 하나의 실수 R로 주

어진다. 거리가 R 이상 떨어진 노드 간에는 직접 교신이 가능하지 않다. 실제로는 거리가 R 이내인 노드 간에도 항상 교신이 가능한 것은 아니다. 노드들 간의 교신은 예측할 수 없는 다양한 이유로 실패할 수 있다. 이런 측면을 반영하기 위해서 본 논문에서는 거리가 R 이내에 놓여 있는 두 노드는 거리의 제곱에 반비례하는 확률, 즉 두 노드간의 거리를 d라고 했을 때 $\frac{1}{R^2}(R-d)(R+d)$의 확률로 교신에 성공한다고 가정하고 모의실험을 수행하였다. 이런 가정이 현실을 정확히 반영한다고는 할 수 없겠으나 어느 정도는 그런 효과가 있다고 생각된다. 넷째로 앵커 노드는 자신의 위치를 미리 알고 있다는 사실을 제외하고는 보통의 노드와 완전히 동일한 기능을 가지고 있다고 가정한다.

본 논문의 구성은 다음과 같다. 먼저 3장에서는 본 논문에서 제시하는 알고리즘이 기초하고 있는 DV 최단 경로 알고리즘에 대해서 소개하고, DV 알고리즘을 이용한 간단한 위치 추정 알고리즘인 NDV-distance 알고리즘을 설명한다. 4장에서는 본 논문이 제시하는 알고리즘의 핵심적인 부분인 임의의 두 노드가 서로 장애물에 가로막혀 있는지 아닌지를 판단하는 기법을 설명한다. 5장에서는 4장에서 설명한 기법을 이용하여 앵커 노드가 존재하는 경우에 적용되는 위치 추정 알고리즘을 설명하고, 6장에서는 앵커 노드가 없는 경우의 위치 추정 알고리즘을 기술한다. 마지막으로 7장에서는 결론을 맺는다.

3. 최단 경로를 찾는 DV (Distance Vector) 알고리즘

3.1 DV 알고리즘

DV 알고리즘은 네트워크상에서 노드들 간의 최단 경로를 구하는 잘 알려진 분산(distributed) 알고리즘이다. 네트워크상에서 임의의 두 노드 u와 v간의 최단 경로의 길이를 $d(u,v)$라고 표시하고, 두 노드간의 실제 거리를 $ed(u,v)$라고 표시하자. 이하에서는 DV 최단경로 알고리즘에서 특정한 출발 노드 v_0로부터 다른 모든 노드로의 최단 경로가 어떻게 찾아지는지 설명하겠다.

처음에 $v \neq v_0$인 모든 노드 v에 대해서 $d(v_0,v) = \infty$이다. 알고리즘은 출발 노드 v_0가 자신의 이웃 노드들에게 메시지를 방송(broadcast)하는 것으로 시작된다. 알고리즘이 진행되는 동안 전송되는 모든 메시지에는 그 메시지를 전송하는 노드 v와 출발 노드 v_0간의 최단 경로의 길이 $d(v_0,v)$가 포함된다. 물론 최초에 출발 노드 v_0가 이웃 노드들에게 방송한 메시지에는 길이 $d(v_0,v_0)=0$가 포함된다.

임의의 노드 v가 인접한 노드 w로부터 메시지를 전송 받으면 노드 v는 자신이 현재 알고 있는 거리 $d(v_0,v)$와 메시지에 포함된 거리 $d(v_0,w)$를 비교하여 다음과 같이 자신의 거리 $d(v_0,v)$를 갱신한다.

if ($d(v_0,v) > d(v_0,w) + ed(w,v)$)
then
$$d(v_0,v) = d(v_0,w) + ed(w,v);$$

여기서 $ed(w,v)$는 인접한 노드인 w와 v간의 측정된 거리이다. $d(v_0,v)$의 값이 갱신되면 노드 v는 새로운 메시지를 이웃들에게 방송하며, 그 메시지에는 갱신된 $d(v_0,v)$값이 포함된다. 더 이상 새로운 메시지가 전송되지 않는 상태에 이르면 각 노드가 가진 거리 값이 바로 노드 v_0로부터 자신까지의 최단 경로의 길이이다.

본 논문에서 제시하는 위치 추정 기법은 기본적으로 DV 알고리즘에 기반하고 있다. 본 논문에서는 모든 노드들이 고유한 ID를 가지고 있고, 각 노드가 전송하는 모든 메시지에는 메시지를 전송하는 노드의 ID가 포함되어 있다고 가정한다. 어떤 노드 v에 대해서 노드 w가 전송한 메시지에 의해서 $d(v_0,v)$값이 최종적으로 갱신되었다는 것은 노드 v_0로부터 노드 v까지의 최단경로 상에서 노드 v에 도달하기 직전에 지나는 노드가 w임을 의미한다. 이 노드를 노드 v_0로부터 노드 v까지 이르는 최단 경로 상의 선행(predecessor) 노드라고 부르고, $pred(v_0,v)$라고 표시한다. 즉 $pred(v_0,v) = w$가 된다. $pred(v_0,v)$는 다음과 같이 구해진다.

if ($d(v_0,v) > d(v_0,w) + ed(w,v)$)
then
$$d(v_0,v) = d(v_0,w) + ed(w,v);$$
$$pred(v_0,v) = w;$$

3.2. Nearest DV-distance 위치 추정 알고리즘

DV 최단 경로 알고리즘에 기반 한 가장 간단한 형태의 위치 추정 알고리즘은 DV-distance 알고리즘이다[5]. DV-distance알고리즘에서는 각각의 노드는 각각의 앵커 노드로부터 자신까지의 거리를 DV 알고리즘을 이용하여 알아낸다. 세 개 이상의 앵커 노드로부터 자신까지의 거리를 추정한 노드들은 앵커 노드를 참조 노드로 하여 자신의 위치를 멀티레터레이션 기법으로 추정한다.

DV-distance 알고리즘을 장애물이 있거나 노드들이 분포한 영역이 비정형인 경우에 적용할 수 있는 가장 간단한 방법의 하나는 각 노드들이 앵커 노드들 중에서 자신에게 가장 가까운 세 개의 앵커 노드만을 참조 노드로 이용하여 자신의 위치를 추정하는 방법이다. 일반적으로 멀리 떨어진 앵커일수록 장애물 가려져 있을 가능성이 높다고 보면, 이 방법은 나름대로 일리가 있다고 할 수 있다. 본 논문에서는 이렇게 자신에게 가장 가까운 세 앵커노드를 참조 노드로 하여 자신의 위치를 추정하는 알고리즘을 NDV-distance 알고리즘이라고 부르며, 4장에서는 우리가 제안한 알고리즘의 성능을 NDV-distance 알고리즘과 비교하였다.

4. 최단 경로의 유효성 판정 기법

임의의 노드 v에 대해서 앵커 u와 노드 v를 연결하는 직선이 장애물을 통과하거나 혹은 노드들이 존재하는 영역을 벗어나게 될 때 앵커 u는 노드 v에 대해서 유효하지 않다(invalid)고 말하고, 그렇지 않을 경우에 앵커 u는 노드 v에게 유효하다(valid)고 말한다. 가령 그림 2에서 앵커 a와 b는 노드 v에 대해서 유효하지 않으며, 노드 w에 대해서는 유효하다.

문제는 장애물의 존재를 알지 못하는 상황에서 각각의 노드가 어떻게 앵커의 유효성을 판단할 수 있는가 하는 것이다. 우리의 기본적인 아이디어는 다음과 같다. 그림 4에서 노드 a와 b로부터 노드 v에 이르는 장애물을 관통하지 않는 기하학적 최단 경로를 고려해 보자. 여기서 기하학적 최단 경로란 다른 노드들을 경유하는 네트워크상의 경로가 아니라 평면상에서 장애물을 피해서 두 노드를 연결하는 가장 길이가 짧은 선을 의미한다. 만약 어떤 앵커가 어떤 노드에 대해서 유효하다면 기하학적 최단 경로는 물론 두 노드를 연결하는 직선이 될 것이다. 그림 4에서 굵은 선이 노드 a와 b로부터 노드 v까지의 기하학적 최단 경로이다. 노드 a와 b로부터 노드 v까지의 기하학적 최단 경로는 그림 5와 같이 장애물의 모서리에서 합쳐질 것이다. 일반적으로 하나의 장애물을 동일한 방향으로 우회하는 기하학적 최단 경로는 장애물의 모서리에서 만나게 된다.

네트워크상에서 노드들을 경유하여 두 노드들 연결하는 최단 경로는 물론 기하학적인 최단 경로와는 다르지만 기본적으로 기하학적 최단 경로를 근사하는 (approximate) 경로이므로, 만약 두 앵커 노드가 그림 4의 앵커 노드 a와 b처럼 비교적 근거리에 위치해 있다며, 두 앵커로부터 장애물 너머의 노드 v까지의 두 최단 경로는 모두 장애물의 꼭지점 부근을 지나게 되고, 그 부근에서 그림 4와 같이 합쳐질 가능성이 높다. 최단 경로는 일단 합쳐지면 다시 떨어지지 않기 때문에 결국 노드 v의 입장에서는 $pred(a,v) = pred(b,v)$가 된다. 즉, 앵커 a와 b로부터 노드 v에 이르는 최단 경로는 동일한 선행 노드를 가지게 된다.

본 논문에서는 이러한 관찰에 근거하여 다음과 같이 앵커 노드의 유효성을 판정한다. 먼저 각각의 앵커 a는 자신의 인접 노드 중에서 두 노드 b와 c를 선택한다. 노드 b와 c는 앵커 노드가 아니어도 무관

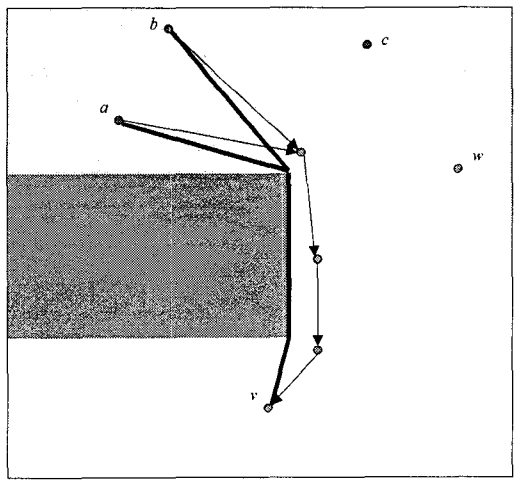


그림 4. 기하학적 최단 경로

하다. 이때 세 노드가 형성하는 삼각형이 다음과 같은 조건을 만족하도록 선택한다.

$$\min\{ed(a,b), ed(b,c), ed(c,a)\} \geq c_1 \times R$$
$$\min\{\angle abc, \angle bca, \angle cab\} \geq c_2 \times \pi$$

여기서 R은 노드들의 최대 교신 가능 범위이다. 즉 세 노드들이 최소 c_1R 만큼씩 서로 떨어져 있으면서, 또한 세 노드가 이루는 삼각형의 최소 내각이 $c_2\pi$ 이상이다. 이렇게 선택된 노드 b와 c를 각각 앵커 a의 친구(friend) 노드라고 부fms다.

앵커 a에 대해서 친구 노드 b와 c가 선택되면 세 노드는 각각 DV 알고리즘의 출발 노드가 되며 네트워크의 각 노드는 세 노드로부터의 최단 경로를 구한다. 임의의 노드 v에 대해서 만약 $pred(a,v) = pred(b,v) = pred(c,v)$가 되면 노드 v는 앵커 a가 자신에게 유효하지 않다고 판정하고, 그렇지 않으면 유효하다고 판정한다.

임의의 앵커에 대해서 유효한 노드들 중에서 실제 이 방법으로 유효하다고 판정되는 비율을 히트 율(hit ratio)라고 부르고, 반면 유효하다고 판정된 노드들 중에서 실제로는 유효하지 않은 노드들의 비율을 미스-히트 율(mis-hit ratio)라고 부른다. 그림 6은 $c_1 = \frac{2}{3}$, $c_2 = \frac{1}{6}$로 설정한 경우의 히트 율과 미스-히트 율을 모의실험을 통해 분석한 결과이다. 모의실험의 환경은 2장에서와 동일하며, 노드의 평균 분지수(degree)는 각 노드에 인접한 노드의 개수를 의미한다. 그림 6은 평균 분지수 10~15의 영역에서 미스-히트 율이 0~1% 수준으로 충분히 낮음을 보여준다. 반면 히트 율은 45~70% 수준으로 나타났다. 이 수치 자체는 특정한 실험 환경, 즉 노드의 수와 장애물의 형태 등에 의존적인 값이다. 유효한 앵커라고 하더라고 실제로는 거리가 멀어지면 유효하지 않게 판정될 가능성이 높아진다. 따라서 동일한 노드 밀도를 유지하면서 노드의 수가 늘어나면 히트 율은 감소할 수밖에 없고, 반대로 노드의 수가 감소되면 히트 율은 일정 정도 증가한다. 그럼에도 불구하고 이 실험은 우리가 제시한 방법이 유효한 앵커와 유효하지 않은 앵커를 구분하는 상당한 변별력을 가지고 있음을 입증하는 근거가 될 수 있다고 생각된다.

히트율을 높이기 위해서는 상수 c_1의 값을 높이거나, 혹은 친구 노드의 수를 늘릴 수도 있다. 반면 상수 c_1의 값을 높이게 되면 그러한 조건을 만족하는 친구 노드가 존재하지 않을 가능성도 높아지게 된다. 다른 방법으로는 어떤 앵커 a를 유효하다고 판정한 노드 v가 존재할 때, 노드 v로부터 일정한 거리 이내에 있는 노드들은 모두 앵커 a를 유효하다고 판정하는 방법도 있다. 이 경우 히트 율은 상당한 정도로 높아지며, 미스 히트 율도 높아지게 되나, 미스 히트가 발생한 경우라도 실제로는 장애물에 의해서 살짝 가려지는 노드일 가능성이 높으므로 경로의 왜곡 정도가 심하지 않은 노드일 가능성이 높게 된다. 반면 이 경우에는 인접한 노드들 간에 자신에게 유효한 앵커에 관한 정보 교환이 필요해진다.

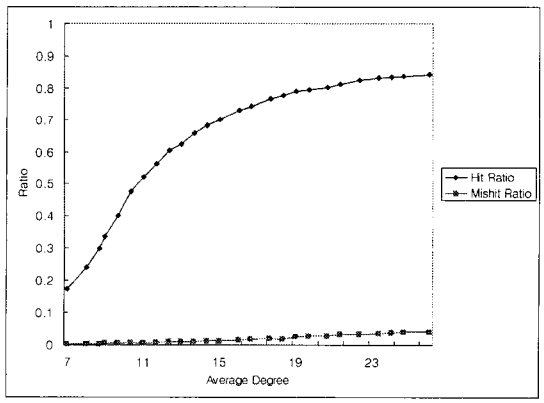


그림 5. 히트율과 미스 히트율

5. 앵커 노드가 있는 경우의 위치 추정

5.1 알고리즘

본 장에서는 3장에서 기술한 앵커 유효성 판정 기법을 사용하여 앵커 노드가 존재하는 경우에 노드들의 위치를 추정하는 알고리즘을 제시한다. 앵커 노드의 수가 충분히 많은 경우에는 각 노드가 자신에게 유효한 앵커들을 찾고, 유효한 앵커들을 참조 노드로 하여 자신의 위치를 추정하는 것만으로 비교적 안정적인 위치 추정이 가능하다. 그러나 앵커 노드의 수가 충분하지 않을 경우에는 세 개 이상의 유효한 앵커 노드를 확보하지 못하는 노드의 비율이 늘어나기 때문에 위치 추정의 성공률이 낮아진다. 따라서 2장에서 기술한 바와 같은 가상 앵커를 이용한 2단계

위치 추정 전략이 필요하게 된다. 가상 앵커를 이용하는 2단계 위치 추정 알고리즘은 다음과 같다.

(1) 각각의 앵커 노드는 인접한 노드들 중에서 두 개의 친구 노드를 선택한다. 각각의 앵커 노드와 친구 노드들은 자신을 출발점으로 하는 DV 알고리즘을 수행하며, 네트워크의 모든 노드들은 각각의 앵커 및 친구 노드로부터의 최단 경로의 길이와 선행 노드를 알아낸다.

(2) 각각의 노드는 자신에게 유효한 앵커 노드를 가려내며, 3개 이상의 유효한 앵커를 확보한 노드들은 그들을 참조 노드로 하여 자신의 위치를 추정한다. 이렇게 위치 추정에 성공한 노드들을 1차 노드라고 부른다.

(3) 1차 노드들 중에서 다음과 같은 조건을 만족하는 노드 v는 스스로를 가상 앵커로 선정하고, 그 사실을 인접 노드들에게 알린다.

① 자신의 위치 추정에 참조 노드의 역할을 수행한 유효한 앵커들이 적어도 하나의 각 변의 길이가 $\frac{R}{2}$이상이고, 세 내각의 크기가 각각 $\frac{\pi}{6}$ 이상인 삼각형을 형성한다.

② 멀티레터레이션의 평균 오류 $\frac{1}{|Ref|}\sum_{\forall u \in Ref}(cal_{dist}(u,v)-est_{dist(u,v)})^2$가 $\epsilon=\frac{1}{16}R^2$ 이하이다. 여기서 Ref는 위치 추정에 사용된 참조 노드의 집합이다.

③ 인접한 노드 중에 이미 가상 앵커로 선택되어서 그 사실을 통보해온 노드가 존재하지 않는다.

(4) 가상 앵커로 선택된 노드는 인접한 노드들 중에서 두 개의 친구 노드를 선택하며, 가상 앵커와 그 친구 노드는 단계 (1)에서의 앵커 노드와 마찬가지로 스스로를 출발점으로 하는 DV 알고리즘을 수행한다.

(5) 가상 앵커를 포함하여 세 개 이상의 유효한 앵커 노드를 확보한 모든 노드들은 자신의 위치를 추정한다. 이런 노드들을 2차 노드라고 부른다.

장애물의 형태나 영역의 복잡도에 따라서 이러한 단계들을 거치고 나서도 위치 추정에 실패하는 노드들이 존재할 수 있다. 이러한 노드들을 잔여 노드라고 부른다. 잔여 노드들의 위치를 추정하기 위해서 다음과 같은 단계를 추가한다.

(6) 자신의 위치 추정을 완료한 모든 노드는 자신의 추정된 위치를 이웃 노드들에게 전송한다.

(7) 위치 추정을 완료하지 못한 노드들은 이웃 노드의 위치를 참조하여 자신의 위치를 결정한다.

(8) 단계 (6)과 (7)은 모든 노드들의 위치 추정이 완료될 때까지 반복된다.

5.2 성능 평가

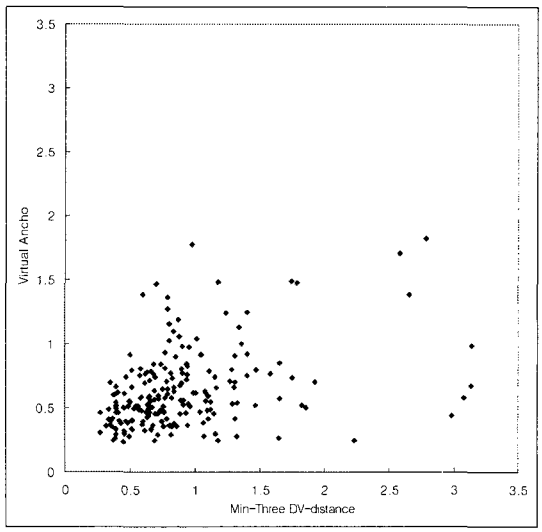
본 절에서는 우리가 제시한 알고리즘의 성능을 NDV-distance 알고리즘과 비교한 모의실험의 결과를 기술한다. 모의실험은 다음과 같은 환경에서 이루어졌다. 그림 3에서와 같은 ㄷ-자 형태의 영역에 300개의 센서 노드들이 균일하게 분포되어 있다. 앵커 노드의 개수는 전체 노드 수의 2%와 4%에 해당하는 6개인 경우와 12개인 경우를 각각 실험하였으며, 앵커 노드는 분포된 노드들 중에서 무작위로 선택되었다. 노드들 간의 측정된 거리는 실제거리에 대해서 ±10%의 범위 내에 균등하게 분포되어 있다고 가정하였다. 노드 밀도는 평균 분지수가 9~10 정도인 경우와 13~14 정도인 두 경우를 대상으로 하였다. 노드 밀도는 각 노드의 교신 가능 거리를 변경함으로써 조절되었다.

그림 6, 7, 8, 9는 각각의 경우에 대해서 두 알고리즘의 평균 위치 추정 오차를 측정한 결과를 보여준다. 그림에서 각각의 점은 300개의 센서 노드로 구성된 하나의 샘플 네트워크를 의미하며, 각각의 경우에 대해서 총 200개의 샘플 네트워크를 무작위로 생성하여 실험하였다. 각 샘플 네트워크에 대해서 가로축은 우리가 제안한 알고리즘의 오차율이며, 세로축은 NDV-distance 알고리즘을 적용한 경우의 오차율이다. 위치추정 오차는 노드들의 최대 교신 가능 거리를 1로 정규화 했을 때 각 노드들의 실제 위치와 추정 위치 사이의 거리의 평균이다.

그림에서 45도 방향의 대각선을 기준으로 아래쪽에 위치한 점들은 NDV-distance 알고리즘이 우수한 결과를 나타낸 샘플들이고, 반대로 대각선 위쪽에 위치한 점들은 우리가 제안한 알고리즘이 더 우수한 결과를 나타낸 샘플들이다.

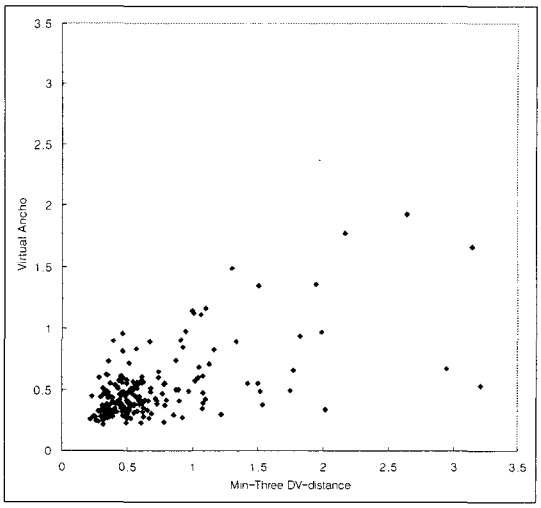
그림에서 보는 바와 같이 동일한 노드 밀도와 동일한 앵커 수에 대해서도 샘플들 간의 오차율의 편차가 비교적 큰 편이다. 이것은 앵커 노드의 위치가 오차율에 절대적인 영향을 미친다는 것을 의미한다.

반드시 그렇다고 할 수는 없지만 대체적으로 평균 오차율이 1.5 이내일 경우에는 전체적인 노드들의 분



	오차율	
	≥ 1.5	≤ 0.5
NDV	19	31
Ours	4	87

그림 6. ㄷ-자 형태, 앵커 수 6, 평균 분지수 9~10인 경우

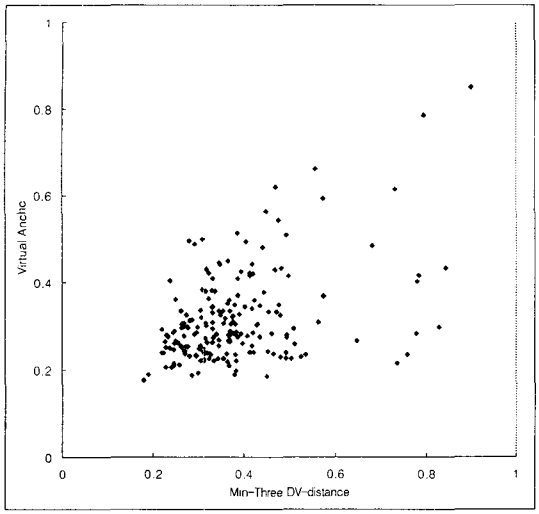


	오차율	
	≥ 1.5	≤ 0.5
NDV	14	88
Ours	3	123

그림 7. ㄷ-자 형태, 앵커 수 6, 평균 분지수 13~14인 경우

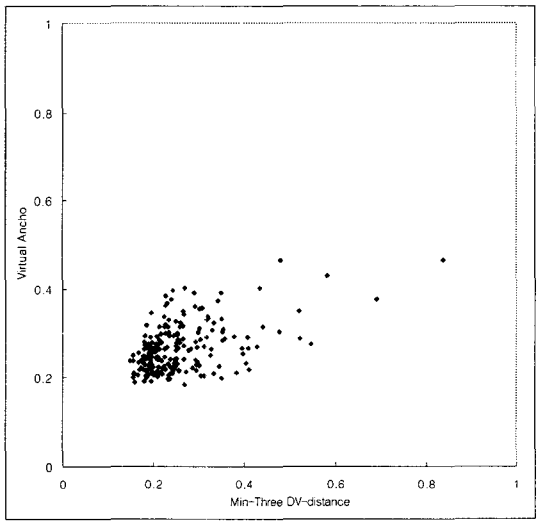
포 형태, 즉 ㄷ-자 형태는 유지되는 것으로 나타났다. 이런 경우에는 추가적으로 위치 수정 기법들을 적용하여 상당한 정도로 정확도를 향상하는 것이 가능하다. 두 알고리즘의 오차율이 모두 0.5 이내에 들어가는 경우는 앵커 노드들이 이상적인 위치에 배치된 경우에 해당한다. 그런 경우에 두 알고리즘의 오차율에는 특기할 만한 차이가 없는 것으로 나타났으며, 이는 알고리즘의 특성상 당연한 결과이다.

두 알고리즘의 가장 주목할 만한 차이는 오차율이 대략 1.5 이상인 경우, 즉 비정형의 영역 형태에 적응하는데 완전히 실패하는 빈도에 있어서의 차이이다. 그림 6, 7, 8, 9 각각에 서 옆에 기술된 표는 각각의 경우에 200개의 샘플 네트워크 중에서 위치 추정 오차율이 1.5이상인 경우와 0.5이하인 경우의 빈도수를 표시한 것이다. 대부분의 환경에서 우리가 제안한 알고리즘이 위치 추정에 완전히 실패하는 빈도가 훨씬 낮다는 것을 확인할 수 있다. 가령 그림 6에서 보는 바와 같이 앵커 수 6, 평균 분지수가 9~10인 경우에



	오차율	
	≥ 1.5	≤ 0.5
NDV	2	178
Ours	0	190

그림 8. ㄷ-자 형태, 앵커 수 12, 평균 분지수 9~10인 경우



	오차율	
	≥ 1.5	≤ 0.5
NDV	1	193
Ours	0	200

그림 9. ㄷ-자 형태, 앵커 수 12, 평균 분지수 13~14인 경우

오차율이 1.5 이상인 샘플들은 NDV-distance 알고리즘의 경우 전체 샘플의 10%에 이르지만 우리가 제안한 알고리즘의 경우 2% 이하이다.

우리가 제안하는 알고리즘은 각 앵커마다 두 친구 노드가 존재하고, 모든 앵커와 그 친구 노드들로부터 최단 경로 알고리즘을 수행해야 하므로 결과적으로 네트워크 트래픽이 NDV-distance 알고리즘에 비해 3배 정도 증가하게 된다.

6. 앵커가 없는 경우의 위치 추정

앵커가 없는 경우에 위치 추정을 위해서는 먼저 좌표계를 설정해야 한다. 본 논문에서는 좌표계를 설정하기 위해서 코어-노드 기법을 따른다. 일단 코어 노드가 선정되고 나면 코어 노드가 앵커 노드의 역할을 하게 되므로, 기본적으로 앵커 노드에 기반 한 위치추정과 동일한 문제가 된다. 그래서 여기서는 코어

노드를 선정하는 방법만을 기술한다.

6.1 코어 노드의 선정

코어 노드는 다른 노드의 위치를 추정하는데 있어서 기준의 역할을 하기 때문에 조심스럽게 선택되어야 한다. 코어 노드들이 서로 너무 가까운 거리에 모여 있거나 혹은 일직선에 가까운 위치에 있는 것은 바람직하지 않다. 그래서 다음과 같은 조건을 만족하는 세 노드를 찾아서 코어 노드로 선택한다.

조건 1: 세 노드 모두 서로에 대해서 유효하다. 즉, 장애물에 가려져 있지 않다.

조건 2: 세 노드가 이루는 삼각형의 내각이 모두 $\frac{\pi}{6}$이상이다.

조건 3: 세 노드가 이루는 삼각형의 가장 짧은 변의 길이가 가능한 한 길다.

위와 같은 세 조건을 만족하는 코어 노드 *a,b,c*가 선정되면, 세 노드 중에서 임의의 한 노드 a의 좌표는 원점 $(0,0)$으로 설정되고, 다른 한 노드 b의 좌표는 x-축 상의 점, 즉 $(d(a,b),0)$으로 설정된다. 노드 c의 좌표는 다른 두 노드와의 거리 조건을 만족하는 두 지점 중에서 양의 y-좌표를 가지는 지점으로 설정된다.

코어 노드를 선정하기 위해서는 노드들 간의 상호 유효성 및 거리를 알아야 한다. 그러나 모든 노드들 간의 거리를 다 구하는 것은 불필요하게 네트워크의 트래픽을 증가시키게 될 것이다. 따라서 코어-노드를 정하기 위해서 먼저 코어-노드가 될 후보(candidate)들을 선정하고, 그들 간의 최단 경로를 구하고, 그 정보를 이용하여 코어 노드를 선정한다.

1) 각 노드는 일정한 확률 p로 스스로를 코어 노드의 후보로 선정한다. 후보 수가 많을수록 네트워크의 트래픽이 증가하므로 후보 노드의 수는 적절하게 조절할 필요가 있다. 우리가 수행한 모의실험에 의하면 전체 노드의 5% 정도를 후보 노드로 선정하는 것이 적절한 것으로 나타났다. 즉 $p=\frac{5}{n}$ 정도가 적절하다.

2) 선택된 각각의 후보 노드는 자신의 이웃 노드들 중에서 두 개의 친구 노드를 선택하고, 각각의 후보 노드와 그 친구 노드들은 DV 알고리즘을 실행한다. DV 알고리즘이 실행되고 나면 네트워크의 모든 노드들은 유효한 후보 노드로부터의 최단 경로를 알게 된다.

3) 각 후보 노드들의 자신에게 유효한 후보 노드와 그들과의 거리에 관한 정보를 다른 모든 후보 노드들에게 전송한다. 각각의 후보 노드는 모든 후보 노드들 간의 유효성과 거리에 관한 정보를 바탕으로 코어 노드를 선정한다.

단계 3)에서 각 후보 노드가 자신에게 유효한 모든 후보 노드에 관한 정보를 다른 후보 노드들에게 전송하는 것은 코어 노드에 관한 조건 3을 만족하기 위해서이다. 실제로는 조건3을 반드시 만족해야하는 것은 아니고 비교적 큰 삼각형을 찾는 것으로 충분하다. 조건3을 비교적 큰 삼각형을 찾는 것으로 완화하는 대신에 네트워크의 트래픽을 줄이기 위해서 실제로는 다음과 같이 한다. 먼저 각 후보 노드 v는 자신에게 유효한 후보 노드 중에서 가장 멀리 떨어진 후보 노드 w를 찾아서, 두 노드의 ID와 거리 $(v,w,d(v,w))$를 다른 모든 후보 노드들에게 전송한다.

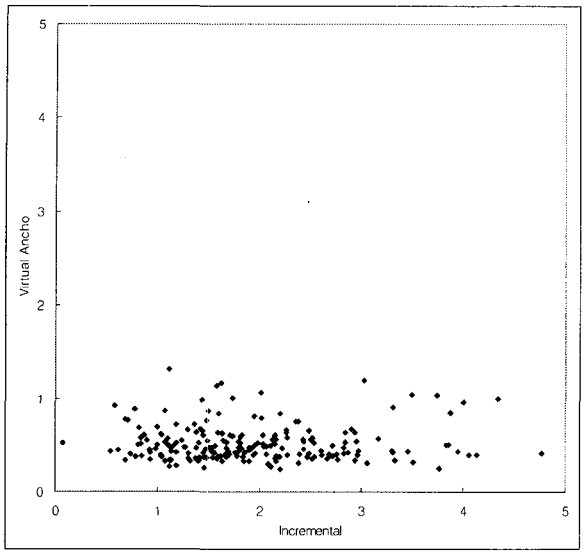
임의의 후보 노드 u가 메시지 $(v.w.d(v,w))$를 받으면, 두 노드 v와 w가 각각 자신으로부터 유효한지 검사한다. 만약 유효하다면 노드 u,v,w에 의해서 형성되는 삼각형이 코어 노드에 관한 조건2를 만족하는지 검사하고, 만족할 경우 세 노드 $u.v.w$는 코어 노드가 될 후보가 된다. 노드 u는 세 노드 u,v,w 및 그들 간의 거리를 포함하는 메시지 $(u.v.w.d(u.v).d(u.w).d(v.w))$를 만들고 이를 다른 모든 후보 노드들에게 전송한다.

코어 노드가 될 후보 세 쌍은 이런 식으로 여러 개가 찾아져서 다른 후보 노드들에게 전송될 수 있다. 2개 이상의 코어 노드 후보 세 쌍을 알게 된 노드들은 그 중에서 세 변의 최소 길이가 큰 후보를 선택하고 나머지는 무시한다. 이런 식으로 모든 노드들은 세 코어 노드를 선정할 수 있다.

이 경우 코어 노드가 찾아지지 않을 가능성은 존재하지만 그 가능성은 무시할 수 있을 정도이다. 또한 일정한 시간이 지나도 코어 노드가 선정되지 않을 경우 각 후보 노드는 다른 후보 쌍을 시도하도록 하면 된다.

6.2 성능 평가

본 절에서는 앵커가 없는 경우의 위치 추정 알고



	오차율	
	≥1.5	≤0.5
Incr	137	1
Ours	0	118

그림 10. T-자 형태의 영역, 평균 분지수 13~14인 경우

리즘의 성능을 모의실험을 통하여 분석한 결과를 기술한다. 기본적인 모의실험의 환경은 5.2절에서와 동일하며 영역의 형태는 T-자 형태이다. 그림 10은 우리의 알고리즘의 성능을 점진적(incremental) 위치 추정 알고리즘과 비교한 결과이다.

점진적 알고리즘에서는 코어-노드를 어떻게 선정하는지가 알고리즘의 성능에 중요한 영향을 미친다. 코어-노드 선택 방법에 따른 차이를 성능비교에서 배제하기 위해서 다음과 같이 실제로는 가능하지 않은 방법을 사용하여 위치 추정을 하였다. 우선 서로 인접하면서 서로 간의 거리가 적어도 $2R/3$ 이상이면서, 내각의 크기가 $\pi/6$인 삼각형을 형성하는 세 노드를 무작위로 선택한 후, 선택된 세 노드와 그들에게 인접한 모든 노드들이 실제 자신의 위치를 알고 있다고 가정하였다. 그런 다음 다른 노드들은 이 노드들을 참조 노드로 하여 LGB기법으로 위치를 추정한다.

위치 추정의 오차는 상대적 좌표계 상에서 정의된 세 코어 노드의 위치를 실제 위치에 가장 근접하게 평행이동, 회전, 그리고 뒤집기 변환을 수행한 후 얻어지는 좌표들에 대해서 5.2절에서와 동일한 방법으로 측정하였다.

실험의 결과는 점진적 알고리즘의 경우 이런 비현실적인 가정에도 불구하고 위치 추정의 오차가 매우 자주 허용할 수 있는 범위를 벗어난다는 사실을 보여주며, 반면 우리가 제안한 알고리즘의 경우에는 매우 안정적인 위치 추정 성능을 보여준다.

7. 결 론

본 논문에서는 멀티레터레이션 기법을 이용하는 위치 추정의 대표적인 두 가지 기법인 GGB 기법과 LGB 기법을 적절히 절충하고, 또한 임의의 두 노드가 장애물에 가로 막혀있는지 아닌지를 판단하는 분산적인 기법을 고안하여, 장애물이 있는 경우에 적용 가능한 위치 추정 기법을 제안하였다. 향후 연구과제는 다음과 같다. 먼저 앵커-프리 위치 추정에서 본 논문에서 실험한 환경보다 좀더 복잡한 형태의 영역에 적절히 적응하기 위해서는 3개 이상의 코어 노드를 사용할 필요가 있다고 보여 진다. 복잡한 형태의 영역에서 영역 전체에 걸쳐있는 많은 수의 코어 노드를 선정하는 기법에 대한 연구가 필요하다. 다음으로는 본 논문의 4장에서 기술한 앵커의 유효성 판단 기법에 대한 보다 정형화된 모델이나 분석이 필요하다고 판단된다.

참 고 문 헌

[1] R. Bischoff and R. Wattenhofer, "Analyzing Connectivity-Based MultiHop Ad-Hoc Positioning," *Proc. of the Second Annual IEEE*

International Conference on Pervasive Computing and Communications, 2004.

[2] N. Bulusu, J. Heidemann, and D. Estrin, "Density adaptive algorithms for beacon placement in wireless sensor networks," *IEEE ICDCS* 2001, Apr. 2001.

[3] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-free low cost outdoor localization for very small devices," *IEEE Personal Comm.*, Vol. 5, No. 5, pp. 28-34, 2000.

[4] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," *Proceedings of The Ninth International Conference on Mobile Computing and Networking (Mobicom)*, pp. 81-95, San Diego, CA, Sept. 2003.

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

[6] R. Nagpal, H. Shrobe, and J. Bachrach. "Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network," *2nd International Workshop on Information Processing in Sensor Networks (IPSN 03)*, Palo Alto, CA, pp. 22-23, 2003.

[7] N. B. Priynatha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," *MIT Laboratory for Computer Science, Technical Report* No. 892, Apr. 15, 2003.

[8] N. Patwari and A. O. Hero III, "Using proximity and quantized RSS for sensor localization in wireless networks," *WSNA'03*, San Diego, California, USA, Sept. 2003.

[9] N. Patwari, A. O. H. III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Trans. on Signal Processing*, Vol. 51, No. 8, pp. 2137-2148, 2003.

[10] A. Savvides, C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, Jul. 2001.

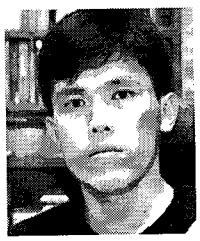
[11] C. Savarese, K. Langendoen, and J. Rabaey, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," *USENIX Technical Annual Conference*, Monterey, CA, pp. 317-328, 2002.

[12] C. Savarese, J. Rabaey, and J. Beutel, "Locating in distributed ad-hoc wireless sensor networks," *Proc. of ICASSP* 2001, pp. 2037-2040, 2001.

[13] A. Savvides, H. Park, and M.B. Srivastava, "The Bits and Flops of the N-hop Multilateration Primitive For Node Localization Problems," *WSNA'02*, Atlanta, Georgia, USA. Sept. 2002.

[14] Xiang Ji and Hongyuan Zha, "Positioning in wireless ad-hoc sensor networks using multidimensional scaling," *IEEE INFOCOM* 2004.

[15] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," *SenSys 2004*, Baltimore, Maryland, USA, Nov. 2004.



권 오 흠

1988년 서울대학교 컴퓨터공학과 졸업 (공학사)
1991년 KAIST 전산학과 (공학석사)
1996년 KAIST 전산학과 (공학박사)
1996년~1997년 전자통신연구소 박사후연수연구원
1997년~1999년 부경대학교 전자컴퓨터정보통신공학부 전임강사
1999년~2003년 부경대학교 전자컴퓨터정보통신공학부 조교수
2003년~현재 부경대학교 전자컴퓨터정보통신공학부 부교수
관심분야 : 알고리즘, 연결 네트워크, 센서 네트워크 등

송 하 주

1993년 서울대학교 컴퓨터공학과 졸업 (공학사)
1995년 서울대학교 컴퓨터공학과 졸업 (공학 석사)
2001년 서울대학교 컴퓨터공학과 졸업 (공학 박사)
2003년 ㈜아이티포웹 부장
2003년 9월~현재 부경대학교 전자컴퓨터정보통신공학부 전임강사
관심분야: 데이터베이스, 웹서비스

김 숙 연

1991년 연세대학교 전산과학과 (이학사)
1993년 KAIST 전산학과 (공학 석사)
1998년 KAIST 전산학과 (공학 박사)
1998년~2004년 한국전자통신연구원(ETRI) 선임연구원
2004년~현재 한경대학교 컴퓨터공학과 조교수
관심분야 : Interconnection Network, 알고리즘, 그래프 이론