Journal of Institute of Control, Robotics and Systems (2011) 17(8):843-850
DOI:10.5302/J.ICROS.2011.17.8.843 ISSN:1976-5622 eISSN:2233-4335

무인자동차의 경로점 주행 시 장애물 회피를 위한 경로생성 알고리즘

A Path Generation Algorithm for Obstacle Avoidance in Waypoint Navigation of Unmanned Ground Vehicle

임 준 혁, 유 승 환, 지 규 인*, 이 달 호
(Jun-Hyuck Im[1], Seung-Hwan You[1], Gyu-In Jee[1], and Dal-Ho Lee[2])
[1]Konkuk University
[2]Kyungwon University

Abstract: In this paper, an effective path generation algorithm for obstacle avoidance producing small amount of steering action as possible is proposed. The proposed path generation algorithm can reduce unnecessary steering because of the small lateral changes in generated waypoints when UGV (Unmanned Ground Vehicle) encounters obstacles during its waypoint navigation. To verify this, the proposed algorithm and A* algorithm are analyzed through the simulation. The proposed algorithm shows good performance in terms of lateral changes in the generated waypoint, steering changes of the vehicle while driving and execution speed of the algorithm. Especially, due to the fast execution speed of the algorithm, the obstacles that encounter suddenly in front of the vehicle within short range can be avoided. This algorithm consider the waypoint navigation only. Therefore, in certain situations, the algorithm may generate the wrong path. In this case, a general path generation algorithm like A* is used instead. However, these special cases happen very rare during the vehicle waypoint navigation, so the proposed algorithm can be applied to most of the waypoint navigation for the unmanned ground vehicle.

Keywords: unmanned ground vehicle, path generation, waypoint navigation, obstacle avoidance

I. 서론

무인자율주행차량이 목적지까지 안전하게 주행하기 위해서는 다양한 센서들로부터 획득된 외부 환경 정보를 취합하고 하나의 디지털화된 맵을 생성하여 최적의 경로계획(optimal path planning)을 수행해야 한다. 특히 경로점 주행에 있어 무인자동차가 주행해야 할 전역경로 상에 장애물이 있을 경우 이를 회피하여 주행할 수 있도록 경로를 재 생성하여야 한다. 대표적인 경로생성 알고리즘에는 Dijkstra 알고리즘, A* 알고리즘, D*(Dynamic A*) 알고리즘, RRT (Rapidly-exploring Random Trees) 알고리즘 등이 있다[1]. Dijkstra 알고리즘은 어떠한 간선도 음수 값을 갖지 않는 방향성 그래프에서 주어진 시작지점과 목표지점 사이의 최단 경로 문제를 푸는 너비 우선 탐색 알고리즘이다[2]. 그러나 알고리즘의 특성상 많은 메모리를 사용하기 때문에 수행속도가 느린 단점이 있다. A* 알고리즘은 상태공간 안의 특정 노드에 인접한 노드들을 조사해 나가면서 시작 지점으로부터 목표 지점에 이르는 가장 적은 비용의 경로를 찾는 알고리즘이다. Dijkstra 알고리즘의 변형인 A* 알고리즘은 탐색해야 할 그래프의 크기를 줄이는 방법으로 더 빠르게 경로를 찾을 수 있다. D* 알고리즘은 맵의 어떠한 지점에서 수정이 있었는지 점검하고, 수정이 있었다면 수정된 노드부터 시작해서 오직 영향을 받은 영역에 해당하는 경로만을 수정한다[3]. 본질적으로 D* 알고리즘은 매우 작은 영역 안에서의 계획 재 수립에 초점을 둔다. 경로를 생성하는데 필요한 계산 비용은 그래프의 노드 개수에 비례하는데, D* 알고리즘은 검색영역을 최소화하기 때문에 A* 알고리즘에 비해 수행속도가 빠른 장점이 있다. RRT 알고리즘은 시작 지점과 목표 지점 사이를 효과적으로 탐색하는 single-query path planning을 해결하기 위한 방법이다[4]. 이 알고리즘은 네 알고리즘 중 가장 최근에 만들어졌으며 현재 무인자동차의 경로생성에 많이 이용되고 있는 추세이다.

일반적으로 차량은 미로와 같이 알 수 없는 길을 찾아가는 것이 아닌 이미 정해진 진행방향을 따라가야 하는 경로점 주행을 한다. 이러한 경로점 주행 시 고려해야 할 장애물은 대부분 정지상태의 간단한 장애물이나 돌발적인 근접 장애물이다. 간단한 장애물을 회피 주행할 때는 급격한 조향 변화가 적을수록 좀 더 안전하고 효율적인 주행이 가능하다. 위에서 언급한 경로생성 알고리즘들은 복잡한 장애물 환경에서 안정적인 경로생성이 가능하다는 장점이 있지만 생성된 경로점의 차량진행방향에 대한 횡방향 변화가 심해 급격한 조향 변화가 자주 발생하기 때문에 경로점 주행 시 효율성이 떨어진다. 물론 생성된 경로의 후 처리를 통해 경로를 효율적으로 수정할 수 있지만 이 경로생성 알고리즘들의 수행속도가 느리기 때문에 또 다른 후 처리 과정을 거치는 것은 비효율적이다. 비교적 수행속도가 빠른 D* 알고리즘의 경우에도 검색영역을 최소화하는데 있어 주변환경의 상황에 따른

* 책임저자(Corresponding Author)
논문접수: 2011. 1. 19., 수정: 2011. 4. 27., 채택확정: 2011. 6. 14.
임준혁, 유승환: 건국대학교 전자정보통신공학과
(junhyuck@konkuk.ac.kr/ysh12@konkuk.ac.kr)
지규인: 건국대학교 전자공학과(gijee@konkuk.ac.kr)
이달호: 경원대학교 전자공학과(dhlee@kyungwon.ac.kr)
※ 본 논문은 2008학년도 건국대학교의 지원에 의하여 연구되었음.

제약이 많아 빠른 수행속도를 항상 보장하지는 못한다. 이렇듯 수행속도가 느리기 때문에 돌발적인 근접 장애물에 대한 대처는 또 다른 장애물 회피 알고리즘을 이용할 수 밖에 없다. 이와 같이 위의 알고리즘들은 이미 주행 방향이 정해져 있는 경로점 주행에는 비효율적인 면이 존재한다. 그러므로 경로점 주행 시 안전하고 효율적인 주행을 위해서는 위의 경로생성 알고리즘들보다 좀더 경로점 주행에 적합한 경로생성 알고리즘이 필요하다.

본 논문에서 제안하는 경로생성 알고리즘은 안전하고 효율적인 경로점 주행을 위해 주행 중 조향 변화와 수행속도의 최소화에 초점을 맞추었다. 본 논문의 알고리즘은 전역경로보다 차량의 진행방향에 중점을 두고 경로를 생성하기 때문에 무인차량의 경로점 주행 중 장애물과 조우 시에 최소한의 조향 변화만으로 장애물을 회피하여 주행할 수 있다. 또한 장애물의 복잡도와 상관없이 계산 시간이 항상 0.1초 이내이므로 또 다른 장애물 회피 알고리즘의 이용 없이 돌발적인 근접 장애물을 회피할 수 있다.

본 논문의 II 장에서는 조향 및 속도 명령 생성과정을 설명하고, III 장에서는 A* 알고리즘에 대하여 설명한다. IV 장에서는 본 논문에서 제안한 경로생성 알고리즘을 소개하고, V 장에서는 A* 알고리즘과 본 논문에서 제안하는 경로생성 알고리즘의 시뮬레이션 결과를 비교 분석한다. 마지막으로 VI 장에서는 결론을 서술한다.

II. 조향 및 속도 명령 생성

무인자율주행차량의 위치제어는 속력제어와 방위각제어를 통해 이루어진다[5-7].

무인차량이 경로추적을 원활히 수행하기 위해서는 차량의 방위각 설정이 매우 중요하다. 그림 1은 방위각 설정을 위한 무인차량 경로제어의 원리를 설명한다[5]. 방위각은 전 경로점과 다음 경로점을 이용하여 구할 수 있는데 횡 방향 거리 d가 최소화 되도록 해야 한다. 그림 1에서 $\vec{a}$ 는 전 경로점에서 차량까지의 벡터이고, $\vec{b}$ 는 전 경로점에서 다음 경로점까지의 벡터이다. 이 두 벡터의 사이 각이 ψ 이고 경로($\vec{b}$)에 대한 차량의 위치에 따라 양, 음의 부호를 갖는다. ψ 는 다음과 같이 구할 수 있다.

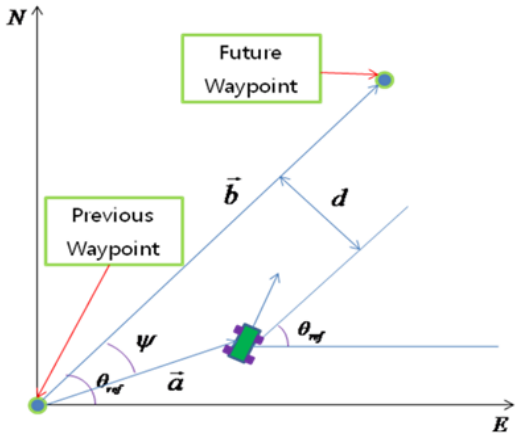


그림 1. 무인자동차의 경로 제어.

Fig. 1. Path control of UGV.

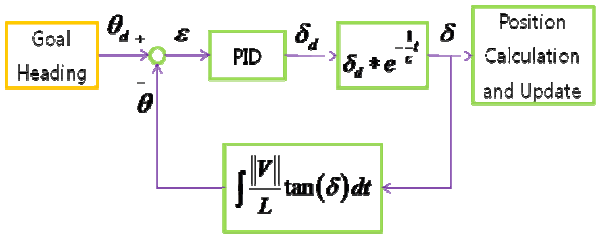


그림 2. 조향 명령의 생성과 Heading 각의 계산.

Fig. 2. The generation of steering commands and target heading angle calculation.

$$\psi = \frac{\det\left[\vec{a},\vec{b}\right]}{\left\|\det\left[\vec{a},\vec{b}\right]\right\|}\cdot\cos^{-1}\left(\frac{\vec{a}\bullet\vec{b}}{\|\vec{a}\|\|\vec{b}\|}\right) \tag{1}$$

현재 무인차량의 위치에서 다음 경로점으로 향하는 새로운 heading 각은 다음과 같이 구할 수 있다.

$$\theta_d = \theta_{ref} + \frac{\|\vec{a}\|\cdot\sin(\psi)}{\tau\cdot\|V_{ref}\|} \tag{2}$$

여기서 $\|\vec{a}\|\cdot\sin(\psi)$ 가 그림 1에서의 d이다. d는 차량이 $\vec{b}$ 의 우측에 있으면 양수이고 좌측에 있으면 음수 값을 갖는다. 이 식은 거리 d를 줄이기 위한 방위각을 제공하며 τ 는 약 1.1~2.0초 사이의 값을 가진다.

그림 2는 조향 명령 생성과정과 위치계산 과정을 간단히 나타낸 것이다. 목표 Heading 각(θ_d)과 현재 Heading 각(θ)의 차이(ε)로부터 조향 명령(δ_d)이 생성되고, 현재 조향 각(δ)으로부터 현재의 위치가 계산 되어지며, 그림에서와 같이 적분을 통해 바로 다음 시간의 Heading 각을 계산할 수 있다. 그림 2의 식들 중 $*$는 Convolution을 나타낸다.

속력제어기로의 입력 값 (V_{ref}) 은 ψ 의 함수로 나타나고, 다음과 같이 구할 수 있다.

$$V_{ref}(\psi) = \frac{V_{Lim}}{2}\left(1+\cos(\psi)\right) \tag{3}$$

여기서 V_{Lim}는 차량의 현재 위치에서의 제한 속도이다.

III. A* 알고리즘

본 논문에서 제안하는 경로생성 알고리즘의 성능은 A* 알고리즘과의 비교를 통해 검증하였다. 본 장에서는 A* 알고리즘을 구현한 방법에 대해 기술한다.

A* 알고리즘은 다음과 같은 비용함수를 기준으로 경로를 생성한다.

$$f(n) = g(n) + h(n) \tag{4}$$

여기서 $f(n)$은 n 노드가 갖는 비용 값, $g(n)$ 은 출발 지점부터 현재의 n 노드까지 거리를 비용으로 환산한 값, $h(n)$ 은 n 노드부터 목표점까지의 거리를 비용으로 환산한 값을 나타낸다. 이때 $f(n)$이 최소가 되는 노드들의 집합이 최종 경로로 선택된다[1]. 그림 3은 A* 알고리즘에서 식 (4)에서의 비용을 구하는 방법과 그에 따른 탐색 과정을 설명한다.

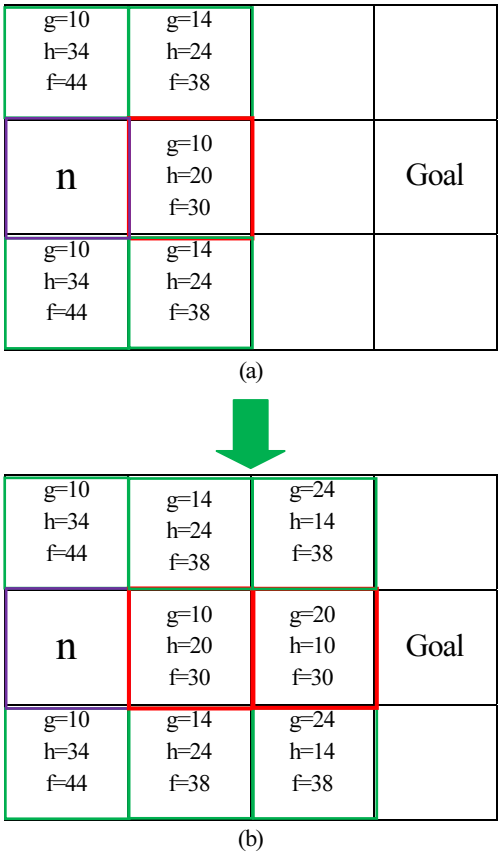


그림 3. A* 알고리즘의 탐색 과정.
Fig. 3. Search process of A* algorithm.

A* 알고리즘은 $f(n)$이 가장 작은 노드를 찾아서 그 노드로 이동하는 과정을 반복함으로써 가장 적은 비용의 최종 경로를 찾는다. 알고리즘의 시작에서 OPEN list에 시작 노드를 넣고, 인접한 노드들을 삽입하여 가장 유망한, 즉 $f(n)$ 비용이 가장 작은 노드로 확장하고 확장한 노드는 CLOSED list에 삽입하면서 인접한 노드들을 OPEN list에 추가한다. 이때 각 노드들 사이의 ARC의 비용은 맞붙어 있는 노드의 경우 10, 대각선에 위치한 노드의 경우 14로 정한다. 그림 3(a)에서는 시작 노드의 오른쪽 노드가 $f(n)$ 값이 가장 작으므로 이 노드가 CLOSED list에 들어가게 된다. 이 과정에서 이웃한 노드들이 미탐색 상태라면 OPEN list에 넣고, OPEN 상태의 노드이면서 이 전 노드들보다 비용이 적으면 그 노드에 대한 정보를 갱신한다. 반대로 CLOSED 상태의 노드들은 이미 조사를 마친 상태이므로 무시한다. 이와 같은 과정을 반복하다 그림 3(b)에서와 같이 인접한 노드들 중 목표 노드가 있다면, 프로세스는 종결된다.

이 일련의 반복과정 중, 부모 노드가 확장하면서 조사되는 인접 노드들을 자식 노드라고 정의하는데, 자식 노드의 백포인터는 부모 노드를 가리킨다. 목표 노드가 확장되면서 목표로부터 백포인터를 따라 시작 노드에 이르는 경로가 A* 알고리즘이 탐색한 최소 비용 경로이다.

본 논문에서는 위의 설명과 같이 A* 알고리즘의 가장 기본적이며 일반적인 방식으로 구현하여 시뮬레이션에 적용하였다.

IV. 제안한 경로생성 알고리즘

본 논문에서는 무인자동차의 경로점 주행에 적합한 경로생성 알고리즘을 제안한다. 차량은 주행 중 주차, 교차로 또는 급격한 코너 구간 주행, U턴 등 특별한 경우를 제외하면 급격한 조향 변화가 불필요하고 이는 무인자동차의 경로점 주행 시에도 마찬가지이다. 본 논문에서 제안하는 경로생성 알고리즘은 전역경로가 아닌 차량의 현재 진행방향에 우선하기 때문에 조향 변화를 최소화시킬 수 있고, 수행속도 또한 빠르기 때문에 갑작스런 장애물에 대한 대처도 가능하다. 다음은 본 논문에서 제안하는 경로생성 알고리즘에 대하여 기술한다.

무인자동차의 경로점 주행 중 장애물과 조우 시 그림 4와 같이 여러 센서들의 정보로부터 생성된 장애물 맵에서 차량의 폭을 고려하여 진행방향에 대해 일정영역만을 탐색한다. 전방에 대한 탐색을 마치면 몇 개의 경로 후보군(가장 긴 3개의 사각형이 나타내는 경로)이 선정되고 그 중에서 최적의 경로를 선택하여야 한다. 그림 5는 선정된 경로들의 경로점을 결정하는 과정을 설명하고, 그림 6은 최종적으로 선택된 최적의 경로를 나타낸다. 그림 4에서 탐색된 여러 경로들 중에 최적의 경로는 정해진 우선순위에 의해 결정된다. 경로선택의 우선순위는 다음과 같다.

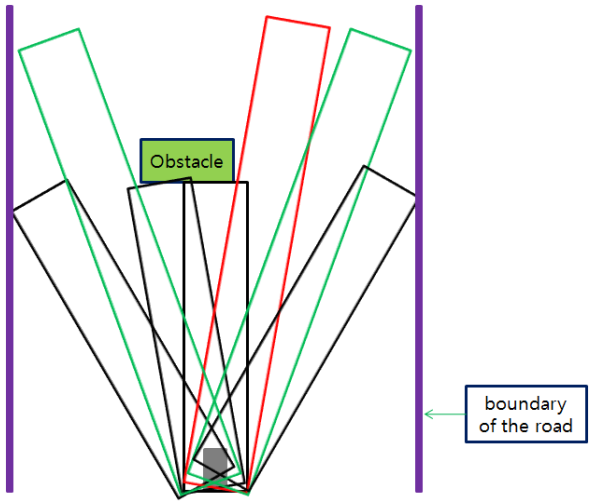


그림 4. 무인자율주행차량의 진행방향에 대한 경로탐색.
Fig. 4. Search of the candidate path for path generation.

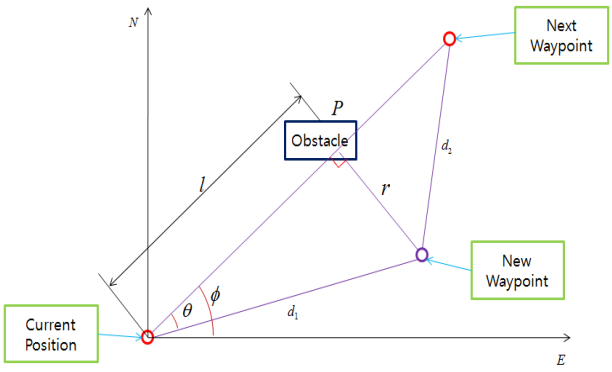


그림 5. 경로점 결정.
Fig. 5. Waypoint decision.

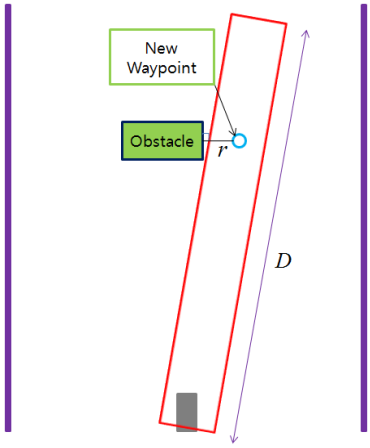


그림 6. 선택된 최종 경로 및 경로점.
Fig. 6. The selected final path and waypoint.

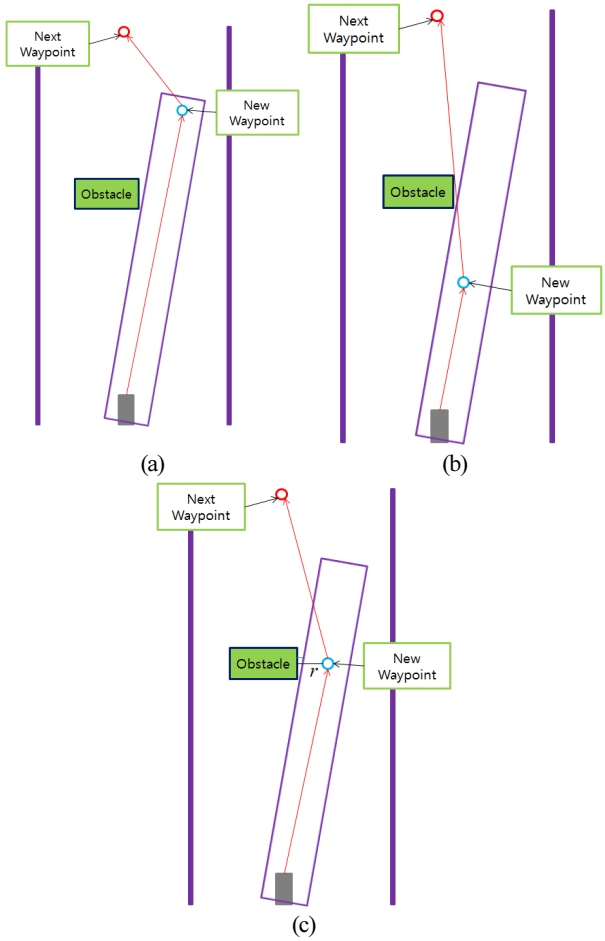


그림 7. 최적의 경로점 선택에 대한 설명. (a) 최대탐색거리에 위치한 경우, (b) 차량에 근접하여 위치한 경우, (c) 최적의 경로점.
Fig. 7. Selection of optimal waypoint. (a) located in the maximum search distance, (b) located in the proximity of the vehicle, (c) optimal waypoint.

1) 탐색영역에 장애물 또는 도로의 경계가 없는 경로
2) 장애물 또는 도로의 경계까지의 거리(D)가 최대인 경로
3) r이 최소인 경로

r이 최소인 경로를 선택해야 하는 이유는 차량의 예상주행거리(d_1+d_2)가 최소인 경로를 선택하기 위해서이다. 또한 새로운 경로점이 차량진행방향벡터와 수직을 이루고 장애물의 위치가 수선의 발인 수선의 연장선상에 위치하도록 하는 이유는 그림 7에서와 같이 설명할 수 있다.

그림 7(a)의 경우 무인자율주행차량이 장애물을 통과한 후에도 불필요하게 도로의 경계에 근접하게 된다. 또한 새로 생성된 경로점을 통과한 후 조향각의 급격한 변화가 이루어지게 되므로 차량의 주행에 악영향을 미친다. 그림 7(b)의 경우에는 예상주행거리(d_1+d_2)가 짧아지지만 주행 자체가 어려움을 알 수 있다. 새로 생성된 경로점을 통과 한 후 장애물에 충돌하게 될 것이고, 그렇지 않고 경로점 통과 후 새로운 경로탐색을 통해 재차 경로점을 구하여 장애물을 통과한다 하여도 이는 불필요한 과정이다. 그러므로 (a)와 (b)가 절충된 (c)가 최적의 선택이라고 할 수 있다.

그림 4의 경우에서는 1)번 우선순위에 의해 탐색된 경로들 중 가장 긴 사각형이 나타내는 3개의 경로가 선택된다. 선택된 경로가 3개이므로 2)번 우선순위에 의해 최적의 경로가 선택되어야 하지만, 2)번 우선순위의 조건은 선택된 3개의 경로들이 모두 만족하므로 그 다음 우선순위인 3)번 우선순위로 바로 넘어간다. 여기서 1)번과 2)번 우선순위를 따로 분리시킨 이유는 원래의 경로에 장애물이 없는 경우 불필요하게 경로탐색을 실행하지 않기 위해서이다. 이제 3)번 우선순위를 고려해보면 그림 5에서와 같이 r이 최소인 경로를 선택해야 한다. r을 구하기 위해 우선 현재 위치로부터 장애물까지의 거리(l)를 구한다. 현재 위치로부터 장애물까지의 거리는 장애물 맵에서 장애물 탐색과 동시에 알 수 있다. 거리 l을 알면 r은 다음과 같이 간단히 구할 수 있다.

$$r = l \cdot \tan(\theta) \tag{5}$$

이와 같이 각각의 경로에 대해 r을 구하면 그림 6의 최종 경로가 선택 되어진다.

본 논문에서는 경로점을 기반으로 한 경로추종주행을 하기 때문에 새로운 경로점을 구하는 과정이 필요하다. 이를 위해 먼저 장애물의 좌표(P)를 구한다. 현재 위치를 [x_0, y_0]라고 했을 때 장애물의 좌표(P)는 다음과 같다.

$$P = \left[x_0 + l \cdot \cos(\phi) \ , \ y_0 + l \cdot \sin(\phi) \right] \tag{6}$$

여기서 ϕ 는 경로의 방위각이다. 장애물의 좌표(P)를 알면 새로운 경로점(NW)은 다음과 같이 구할 수 있다.

$$NW = P + \left[r \cdot \sin(\phi) \ , \ -r \cdot \cos(\phi) \right] \tag{7}$$

이렇게 구해진 경로점은 차량의 조향 명령을 생성하는 기준이 된다. 이때 생성된 조향 명령을 바탕으로 차량이 이동하게 되면 그 순간 환경은 다시 변하게 되므로 경로점 역시 다시 생성되게 된다.

본 논문에서 제안한 경로생성 알고리즘은 경로를 생성함과 동시에 차량의 목표속도를 생성할 수 있다는 장점이 있다. 그림 6에서 차량이 현재 상태에서 주행 가능한 최대거리(D)와 비례하여 목표속도를 설정할 수 있다. 예를 들어 D가 40m라면 현재 차량의 위치부터 40m까지는 직선구간임과 동

시에 안전거리가 확보된 상태이므로 제동거리가 40m 이하가 되도록 하는 최대 속도를 낼 수 있다. 마찬가지로 D가 10m 라면 제동거리가 10m 이하가 되는 속도를 낼 수 있다. 본 논문에서는 속도와 제동거리와의 관계를 정비례한다 가정하여 차량의 제한속도를 다음과 같이 설정하였다.

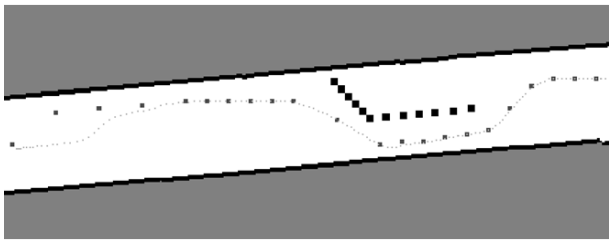
$$V_{Lim} = k \cdot D \tag{8}$$

여기서 k는 비례상수이고, 이렇게 구해진 V_{Lim}가 II 장에서 언급된 차량의 현재 위치에서의 제한 속도이다.

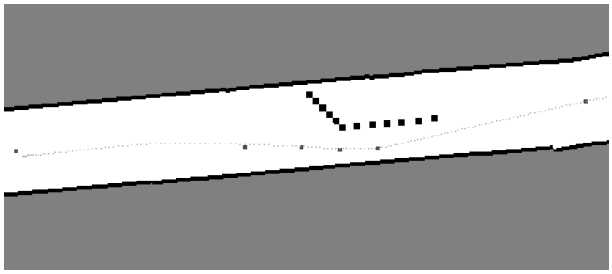
V. 시뮬레이션 결과 및 분석

무인자동차의 주행 시뮬레이션은 정지상태인 장애물 구간을 주행하는 세 가지 상황과 돌발적인 장애물 발생 상황에 대해서 수행하였다. 정지상태인 장애물 구간을 주행하는 각각의 상황을 편의상 협로 구간, S자 구간, 중앙 장애물 구간으로 명명하고, 각각의 상황에 대해 본 논문에서 제안한 알고리즘의 시뮬레이션 결과와 A* 알고리즘의 시뮬레이션 결과를 생성된 경로점, 차량의 이동궤적, 수행속도, 주행 중 조향 변화 등의 비교를 통해 성능을 분석한다. 본 논문의 경로생성 알고리즘은 경로점 주행에 특화된 알고리즘으로서 일반적인 경로점 주행 환경이 아닌 경우 잘못된 경로를 생성할 수 있다. 본 장의 마지막에는 잘못된 경로를 생성하는 경우와 이에 대한 시뮬레이션 결과를 보인다.

그림 8은 협로 구간에 대한 시뮬레이션 결과를 보여준다. 굵은 검정색 점으로 표시된 것이 장애물, 그보다 작은 점들이 생성된 경로점, 회색의 연속된 점들이 차량의 궤적을 나타낸다. 차량은 좌측에서 우측 방향으로 주행하고 도로의 폭은 12m이며 시뮬레이션 시 최대 조향각은 ±30°로 설정하였다. 그림 8에서 차량의 주행 궤적을 살펴보면 A* 알고리즘의 경우 차량의 진행방향에 대해 횡방향 변화가 심한 반면 본 논문에서 제안한 알고리즘은 차량의 진행 방향에 대한 횡방향 변화가 적기 때문에 효율적인 경로점 주행이 이뤄졌음을



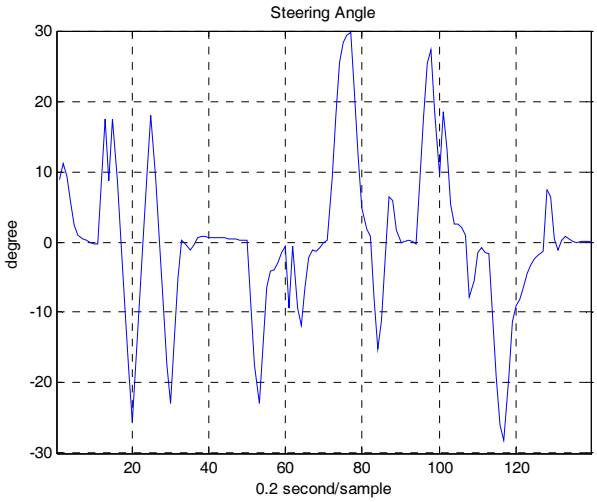
(a) A* algorithm.



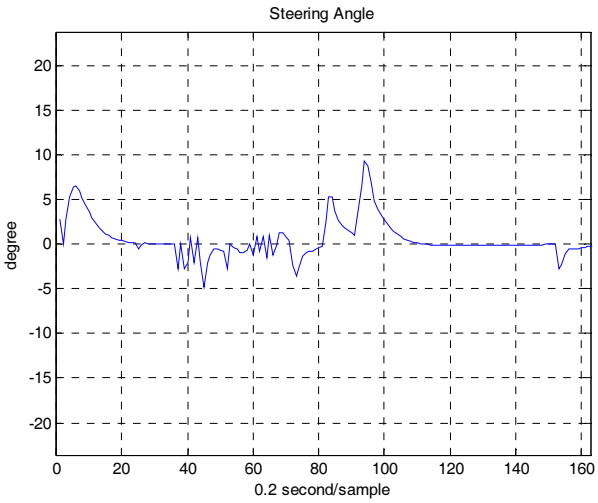
(b) The proposed algorithm.

그림 8. 협로 구간에 대한 시뮬레이션 결과.

Fig. 8. The simulation results for narrow road.



(a) A* algorithm.



(b) The proposed algorithm.

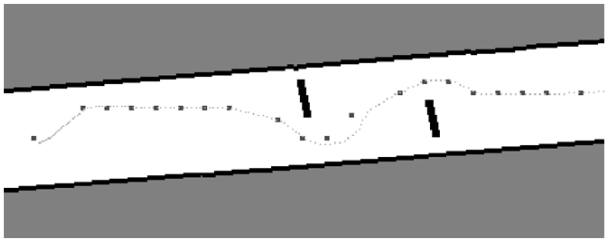
그림 9. 협로 구간 주행 중 조향각 변화.

Fig. 9. Steering angle changes for narrow road.

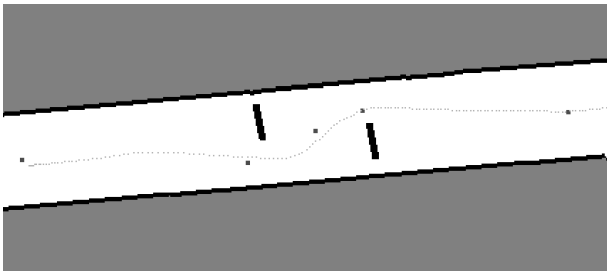
한눈에 알 수 있다. 그리고 제안한 알고리즘은 장애물 통과 후 최대한 현재 진행방향을 유지하도록 경로가 생성된다. 이는 본 논문에서 제안하는 경로생성 알고리즘이 전역경로가 아닌 차량의 현재 진행방향에 우선하기 때문이다. 그림 9는 협로 구간 주행 중 조향각의 변화를 나타낸다. 그림에서와 같이 제안한 알고리즘의 경우 최대 조향각이 ±10° 이내로 A* 알고리즘의 경우에 비해 급격한 조향 변화가 현저하게 적다.

그림 10, 11, 12, 13은 S자 구간과 중앙 장애물 구간에 대한 시뮬레이션 결과를 보여준다. 이는 협로 구간의 경우와 유사하며 차량의 궤적을 보아 본 논문에서 제안한 알고리즘이 A* 알고리즘에 비해 급격한 조향 변화가 현저히 적음을 파악할 수 있다. 그림 11에서 제안한 알고리즘의 경우에도 급격한 조향 변화가 발생하지만 이것은 장애물 회피 주행을 위해 발생하는 것으로 반드시 필요한 조향 변화이다. 이때 첫 번째 장애물에 근접할 수록 그림 6에서의 D가 작아지므로 차량의 속도가 낮아져 급격한 조향 변화가 가능해진다.

표 1은 경로생성 알고리즘의 수행속도를 파악하기 위한



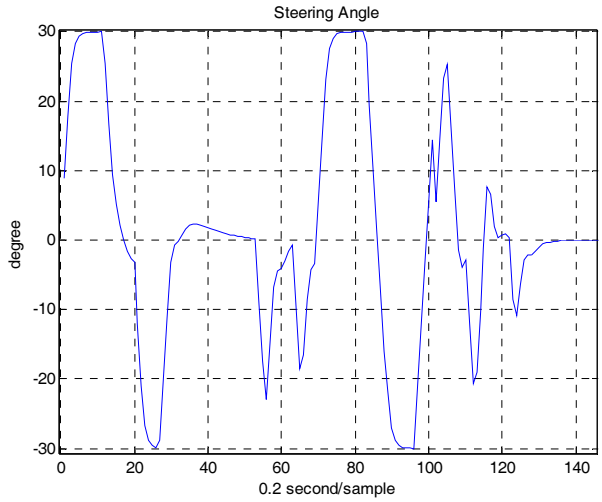
(a) A* algorithm.



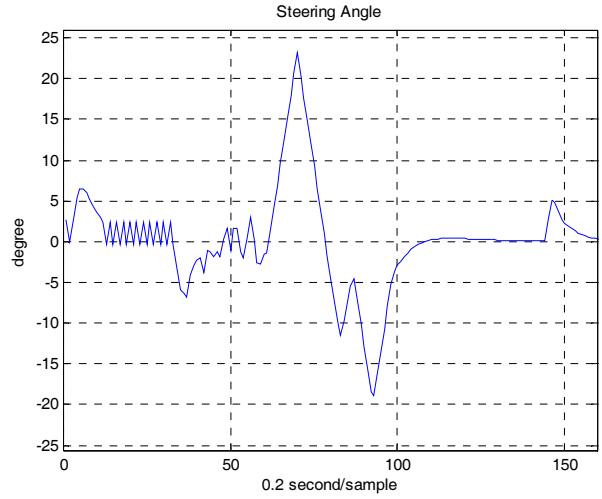
(b) The proposed algorithm.

그림 10. S자 구간에 대한 시뮬레이션 결과.

Fig. 10. The simulation results for S-shaped.



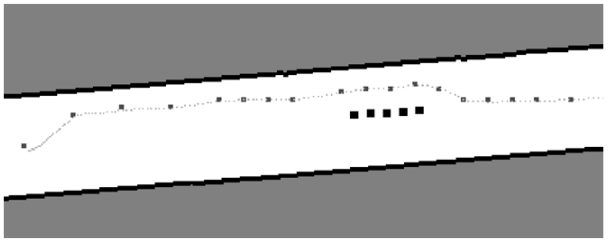
(a) A* algorithm.



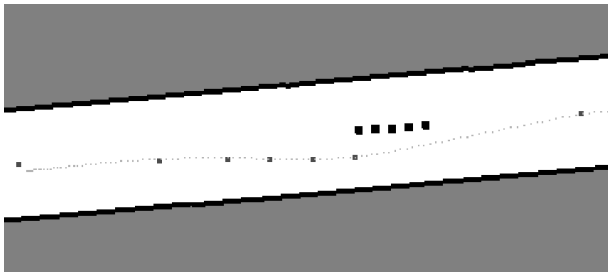
(b) The proposed algorithm.

그림 11. S자 구간 주행 중 조향각 변화.

Fig. 11. Steering angle changes for S-shaped.



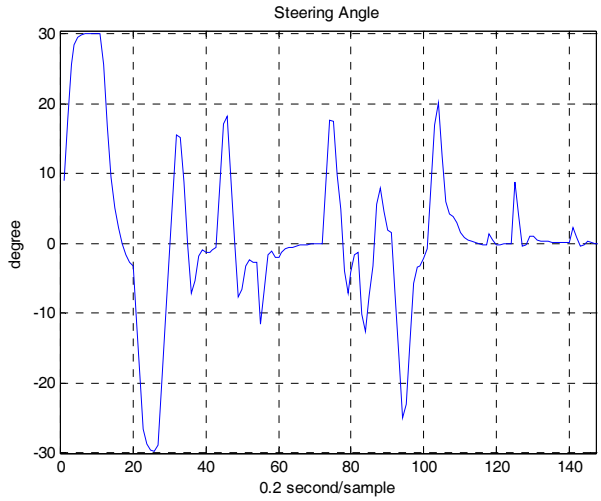
(a) A* algorithm.



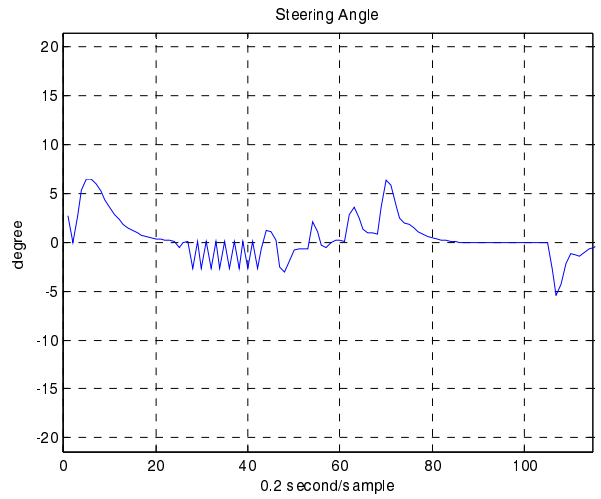
(b) The proposed algorithm.

그림 12. 중앙 장애물 구간에 대한 시뮬레이션 결과.

Fig. 12. The simulation results for central obstacle.



(a) A* algorithm.



(b) The proposed algorithm.

그림 13. 중앙 장애물 구간 주행 중 조향각 변화.

Fig. 13. Steering angle changes for central obstacle.

표 1. 두 알고리즘의 계산시간 비교.
Table 1. Comparison for the computation time of both algorithms.

	협로	S자	중앙장애물
A* 알고리즘	1.4657(s)	1.8908(s)	0.4717(s)
제안한 알고리즘	0.0498(s)	0.0427(s)	0.0576(s)

것으로 각 상황 별 주행 중 계산 시간들의 평균값을 나타낸다. Intel Core2 Quad Q8300 CPU, 4GB 메모리를 사용하였고, matlab을 이용하였다. 그리고 격자 맵의 한 격자 크기는 40cm로 설정하였다. 표 1에서와 같이 제안한 알고리즘의 경우 계산시간이 0.1초 이내로 A* 알고리즘에 비해 수행속도가 빠름을 알 수 있다. 그리고 A* 알고리즘은 장애물에 복잡도에 따라 수행속도의 차이가 많이 발생하는 반면 제안한 알고리즘은 각 상황 별 계산시간이 크게 차이 나지 않는다. 이것은 알고리즘 자체가 경로점 주행만을 고려하여 차량의 진행방향에 대해 항상 일정한 영역 내에서만 검색을 하기 때문이다. 이렇듯 알고리즘의 수행속도가 빠르기 때문에 갑작스러운 근접 장애물에도 효과적으로 대처할 수 있다. 그림 14는 차량이 도로주행 중 갑작스러운 근접 장애물을 만났을 때의 시뮬레이션 결과이다.

그림 14에서 작은 점들은 경로점들이고 검정색의 큰 정사각형이 장애물이다. 그리고 (a)에서처럼 회색 사각형 모양의 차량이 이미 주어진 전역경로를 따라 우측으로 주행 중이다. 차량이 경로점 주행 중 (b)와 같이 장애물이 차량 전방 5~6m 정도에 나타나면 그 순간 차량의 속도를 급격히 낮추게 되고 0.1초 이내에 경로생성이 완료된다. 그렇게 되면 (c)에서와 같이 새로운 경로점이 생성되고 (d)와 같이 갑작스러운 근접 장애물에 대한 회피 주행이 가능하다.

이와 같이 본 논문의 경로생성 알고리즘을 이용하면 차량의 경로점 주행 시 조향 변화가 적어지기 때문에 보다 안전하고 효율적인 주행이 가능하며, 수행속도가 빨라 돌발적인 근접 장애물에 대한 회피 주행이 가능하다. 그러나 본 장의 첫 부분에 언급했듯이 이 알고리즘은 경로점 주행만을 우선 고려하였기 때문에 일반적인 경로점 주행 환경이 아닌 특수한 환경에서는 잘못된 경로를 생성할 수 있다.

그림 15는 본 논문의 경로생성 알고리즘을 이용했을 때 잘못된 경로를 생성하게 되는 경우를 보여준다. (a)에서와 같이 도로의 중앙에 장애물이 있는 경우 장애물을 회피 주행할 수 있는 경로가 생성되고 차량은 생성된 경로를 따라 주행한다. 그러나 차량이 장애물 구간에 이미 진입했지만 (b)에서와 같이 길이 막혀 있는 경우 적절한 경로를 생성하지 못하게 되는데 이는 알고리즘의 특성상 차량의 진행방향에 대해서만 탐색을 하기 때문이다. 이럴 경우 막다른 길에서 빠져 나와 주행을 계속하기 위해 A* 알고리즘과 같은 일반적인 경로생성 알고리즘으로의 전환이 필요하다. 그러나 유효하지 않은 경로가 생성되었다는 것을 판단하기 위해서는 (b)에서와 같이 장애물 구간 진입 초기에는 이를 판단할 근거가 부족하기 때문에 차량이 막다른 길의 끝으로부터 미리 정해진 일정거리까지는 주행하여야 한다. 이때 현재 위치에서의 주행가능거리인 D가 일정치 이하이면 다른 일반적인 경로생성 알고리즘으로 전환한다.

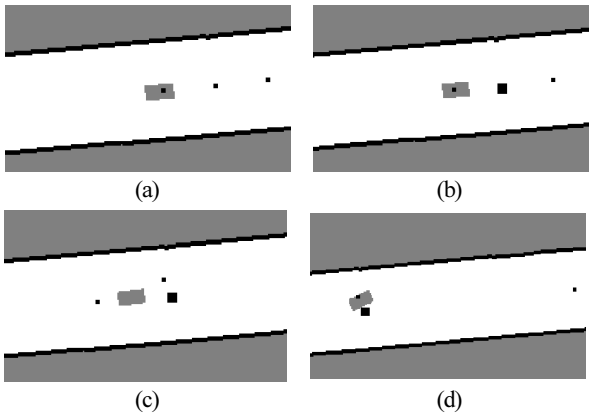


그림 14. 돌발적인 근접 장애물에 대한 회피.
Fig. 14. Avoidance for obstacles that encounter suddenly.

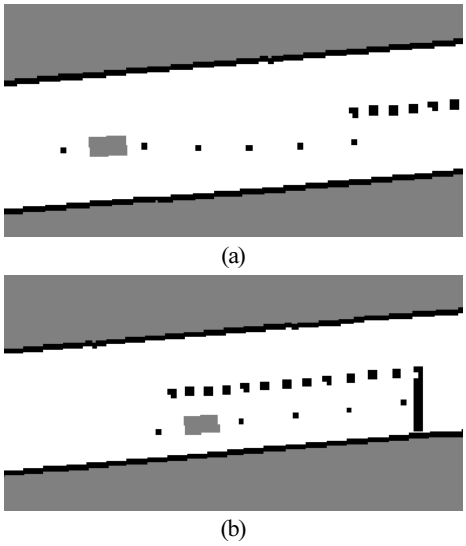


그림 15. 잘못된 경로생성의 예.
Fig. 15. Examples of invalid path generation.

본 논문의 경로생성 알고리즘은 그림 15의 예와 같이 차량이 목표지점까지 주행하는데 있어 차량진행방향으로 도로가 막혀 있지 않아야 한다. 그러나 그림 15와 같은 경우는 매우 특수한 경우로 이와 비슷한 환경은 드물기 때문에 본 논문의 알고리즘은 대부분의 경로점 주행에 적용될 수 있다.

VI. 결론

Dijkstra, A*, D*, RRT 알고리즘 등의 경로생성 알고리즘들은 복잡한 장애물 환경이나 길을 찾기 힘든 경우에 사용하는 대표적인 알고리즘들이다. 그러나 위의 알고리즘들은 이미 주행 방향이 정해져 있는 경로점 주행에는 비효율적인 면이 존재한다. 그러므로 경로점 주행 시 안전하고 효율적인 주행을 위해서는 위의 경로생성 알고리즘들보다 좀더 경로점 주행에 적합한 경로생성 알고리즘이 필요하다.

본 논문에서는 위의 일반적인 경로생성 알고리즘들이 갖는 단점을 보완하여 무인자동차의 경로점 주행에 적합한 경로생성 알고리즘을 제안하였다. 무인자동차의 경로점 주행 시에는 조향 변화를 최소화하여야 보다 안정적이고 효율적인 주행이 가능하며 또한 알고리즘의 수행속도 역시 최소화

시켜야 돌발적인 근접 장애물 회피가 효과적일 수 있다. 시뮬레이션 결과 본 논문에서 제안한 경로생성 알고리즘은 이 두 가지 조건에서 훌륭한 성능을 보여주는 것으로 판단된다.

본 논문의 알고리즘은 경로점 주행에 특화된 알고리즘으로 특수한 경우에 대해서는 잘못된 경로를 생성할 수 있는데, 이 때에는 A* 알고리즘과 같은 일반적인 경로생성 알고리즘으로의 전환을 통해 주행을 계속할 수 있다. 그러나 이러한 특수한 경우는 매우 드물기 때문에 본 논문의 알고리즘은 대부분의 경로점 주행에 적용될 수 있다.

참고문헌

[1] Y. J. Son, "Design of path planning & GPS estimation algorithm for unmanned autonomous ground vehicle," Graduate School of Automotive Engineering, Kookmin Univ., Doctoral Thesis, 2009.

[2] M. Sniedovich, "Dijkstra's algorithm revisited: the dynamic programming connexion," *Control and Cybernetics*, vol. 35, no. 3, 2006.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3310-3317, May 1994.

[4] J. J. Kuffner, Jr., S. M. LaValle, "RRT-connect an efficient approach to single-query path planning," *Proc. 2000 IEEE International Conference on Robitics and Automation*, vol. 2, pp. 995-1001, April 2000.

[5] C. K. Yang and D. S. Shim, "Controller transition management of hybrid position control system for unmanned expedition vehicles," *Journal of Institute of Control, Robotics and Systems*, vol. 14, no. 10, pp. 969-976, 2008.

[6] H. S. Yoon and T. H. Park, "Motion planning of autonomous mobile robot using dynamic programming," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 16, no. 1, pp. 53-60, 2010.

[7] D. H. Kim, C. J. Kim, and C. S. Han, "Geometric path tracking and obstacle avoidance methods for an autonomous navigation of nonholonomic mobile robot," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 16, no. 8, pp. 771-779, 2010.

임 준 혁

2008년 홍익대학교 전자전기공학부(공학사). 2010년 건국대학교 전자정보통신공학과(공학석사). 2011년~현재 건국대학교 대학원 전자정보통신공학과 박사과정 재학중. 관심분야는 무인자동차, Loran-C 신호처리, GPS, INS.

유 승 환

2009년 건국대학교 전자정보통신공학과(공학사). 2009년~현재 건국대학교 대학원 전자정보통신공학과 석사과정 재학중. 관심분야는 항법제어시스템, 자율주행로봇.

지 규 인

1982년 서울대학교 제어계측공학과(공학사). 1984년 서울대학교 제어계측공학과(공학석사). 1989년 Case Western Reserve Univ. System and Control Engineering(공학박사). 1992년~현재 건국대학교 전자공학부 교수. 관심분야는 GPS/INS 결합항법, GPS 수신기 신호처리, 무선측위, 소프트웨어 GPS, GPS anti-jamming.

이 달 호

1982년 서울대학교 공과대학 제어계측공학과 졸업. 1985년 동 대학원 졸업(공학석사). 1992년 동 대학원 졸업(공학박사). 1992년~현재 경원대학교 전자공학과 교수. 1998년 미국 USC 방문연구원. 관심분야는 시스템 식별, 필터링 기법, INS 응용, Data hiding.