<학술논문> DOI:10.3795/KSME-A.2010.34.8.971 ISSN 1226-4873

D* 서치와 퍼지 알고리즘을 이용한 모바일 로봇의 충돌회피 주행제어 알고리즘 설계

정윤하* · 박효운** · 이상진*** · 원문철***†
* STX 조선해양 조선해양연구소, ** 윙쉽테크놀러지㈜ 선형항법팀, *** 충남대학교 메카트로닉스공학과

Development of a Navigation Control Algorithm for Mobile Robots Using D* Search and Fuzzy Algorithm

Yun-Ha Jung*, Hyo-Woon Park**, Sang-Jin Lee*** and Moon-Cheol Won***†
* Shipbuilding & Ocean Research Institute, STX Offshore & Shipbuilding,
** Hull & Navigation Team, Wing Ship Technology Co.,
*** Mechatronics Engineering, Chungnam Nat'l Univ.

(Received August 17, 2009 ; Revised June 17, 2010 ; Accepted June 18, 2010)

Key Word: D*Search(D* 서치), Mobile Robot(이동 로봇), Collision Avoidance(충돌 회피), Fuzzy Logic(퍼지 로직)

초록: 이 논문은 모바일 로봇이 고정 장애물 또는 움직이는 장애물이 존재하는 환경에서 장애물을 회피하며 운행될 수 있는 제어 알고리즘을 연구하였다. 이 제어 알고리즘은 D* 알고리즘과, 충돌 위험도 퍼지로직, 이동로봇의 행동결정 퍼지로직을 사용하여 전역경로계획과 지역경로계획을 수행한다. D* 알고리즘에는 로봇이 이동하는 2 차원 공간을 정방형 격자 분할하여 적용한다. 이 알고리즘은 파이썬 프로그래밍 언어와 이동로봇의 운동방정식을 사용한 시뮬레이션을 통해 검증하였다. 시뮬레이션 결과를 통해 알고리즘을 적용하여 로봇이 이동하는 장애물을 피하거나 모르는 고정 장애물을 피하면서 원하는 위치로 이동하는 것을 볼 수 있다.

Abstract: In this paper, we present a navigation control algorithm for mobile robots that move in environments having static and moving obstacles. The algorithm includes a global and a local path-planning algorithm that uses D* search algorithm, a fuzzy logic for determining the immediate level of danger due to collision, and a fuzzy logic for evaluating the required wheel velocities of the mobile robot. To apply the D* search algorithm, the two-dimensional space that the robot moves in is decomposed into small rectangular cells. The algorithm is verified by performing simulations using the Python programming language as well as by using the dynamic equations for a two-wheeled mobile robot. The simulation results show that the algorithm can be used to move the robot successfully to reach the goal position, while avoiding moving and unknown static obstacles.

1. 서 론

이동로봇(Mobile Robot)이 자율 주행을 함에 있어서 충돌회피 제어 알고리즘은 목적지까지의 최적의 경로를 설정하고 고정 장애물 또는 움직이는 장애물 등을 회피하는데 반드시 필요하다. 이런 충돌회피 제어 알고리즘은 크게 경로계획(Path Planning)과 장애물회피(Collision Avoidance) 등으로 나눌 수 있다. 경로계획 알고리즘은 주어진 지도나 기타 다른 정보들로부터 고정된 장애물과의 충돌을 피해서 원하는 목적지까지 최소의 비용으로 도달하는 최적의 경로를 생성하는 것이다. 이런 경로계획을 전역 경로계획(Global Path Planning)이라고도 하며 Potential Field 를 이용한 알고리즘,[1~4] 유전자 알고리즘(genetic Algorithm),[5] RRTs(Rapidly-Exploring Random Trees),[6] 구배법(Gradient Method, GM) 등[7]의 방법으로 연구되어지고 있다. 일반적으로 전역 경로계획은 시간이 많이 소요되기 때문에 움직이는 장애물이나 주어진 지도 정보 등이 정확하지 않을 경우 실시간으로 경로계획을 변경하기가 용이하지 않을 수 있다. 따라서 실시간으로 경로를 수정하며 장애물 회피가 가능한 알고리즘이 필요하며 이러한 알고리즘이 장애물회피 알고리즘이라고 생각할 수 있다. 장애물회피 알고리

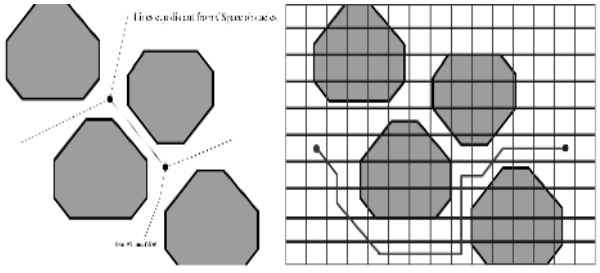
† Corresponding Author, mcwon@cnu.ac.kr

즘은 지역경로계획(Local Path Planning)이라고도 할 수 있는데 움직이는 장애물의 회피나 혹은 잘못된 정보에 기인한 경로의 수정이 필요할 경우, 또는 주어진 정보가 없는 탐험 등을 목적으로 하는 경우에 사용될 수 있으며 VFH(Vector Field Histogram),[8] CVM(Curvature Velocity Method),[9] MPC(Model Predictive Control),[10] DWA(Dynamic Window Approach),[11~13] 퍼지규칙(Fuzzy Rule)을 이용한 방법 등[14]으로 연구되어지고 있다. 지역경로계획은 일반적으로 국부최소(Local Minimum)에 빠질 위험성을 갖고 있기 때문에 단독으로 사용할 경우 목적지에 도달하지 못하는 문제가 생길 수 있게 된다. 본 논문에서는 D*서치 알고리즘[16]을 이용하여 전역 경로계획과 지역경로계획이 하나의 알고리즘으로 융합되는 충돌회피 알고리즘을 제안하고자 한다.

서치 알고리즘은 최단경로 미로찾기 문제와 같이 일반적으로 상태(state)간의 순서를 결정하기 위한 문제에서 비용을 최소화 하여 원하는 목표점을 찾는 알고리즘이다. 서치 알고리즘에서는 BFS(Breadth First Search), DFS(Depth First Search), UCS(Uniform Cost Search), A* 등의 다양한 알고리즘이 있으며,[15] 본 논문에서 D* 알고리즘[16]을 사용하여 충돌회피 제어 알고리즘을 제안하고자 한다.

서치 알고리즘을 적용하기 위해서는 Voronoi 다이어그램[17]을 이용하거나, 정방형의 격자(cell) 모양으로 공간을 분할하여 이동로봇이 움직여야 할 공간의 순서를 찾아야 한다.

Fig. 1 은 Voronoi 다이어그램과 격자분할공간(cell decomposition)을 이용한 이동 공간 분할 방법을 비교한 것이다. Voronoi 다이어그램의 경우 이동로봇이 주행하여야 할 경로는 장애물 사이에 그려진 선들이 되며 선들의 교점이 상태가 된다. 반면 격자분할공간에서는 격자 중심을 잇는 선들이 이동경로가 되며 격자 중심이 상태가 된다. 그렇기 때문에 격자분할공간을 사용할 경우가 Voronoi 다이어그램을 사용하는 경우보다 더 많은 상태를 가지게 되기 때문에 더 많은 데이터 저장 공간과 연산이 필요하게 된다. 하지만 경로를 수정해야 하는 경우에는 Voronoi 다이어그램은 장애물 사이의 선을 새로 만드는 등 어려움이 생기게 된다. 반면 격자분할공간의 경우는 격자단위의 상태가 존재하기 때문에 경로를 수정해야 하는 경우에도 용이하게 사용할 수 있다. 따라서 본 논문에서는 모르는 장애물이나 이동장애물 등을 발견하였을 경우 경로를 수정하는 알고리즘을 구현하고자 하므로 Voronoi 다이어그램보다는 격자분할공간을 사용하였다.



(a) Voronoi Diagram (b) Cell decomposition

Fig. 1 An example of space decomposition for path planning

본 논문에서는 로봇이 이동 장애물과 고정장애물을 구별할 수 있다고 가정한다. 실제로 Ess 등은 참고문헌 [19]에서 비젼시스템을 이용하여 이들을 구별하는 방법을 제시한다. 제안하는 충돌회피 제어 알고리즘은 아래의 Fig. 2와 같은 순서도로 표현될 수 있다.

Fig. 2의 순서도에서처럼 본 논문에서는 D^* 알고리즘과 퍼지 알고리즘을 사용하여 충돌회피 제어 알고리즘을 설계하였다. 최초에 주어진 정보에 의한 전역 경로계획과 주행 도중에 경로를 수정해야 하는 경우에는 D^* 알고리즘을 적용하였고 이동장애물이나 알려지지 않은 고정 장애물 발견 시 경로비용을

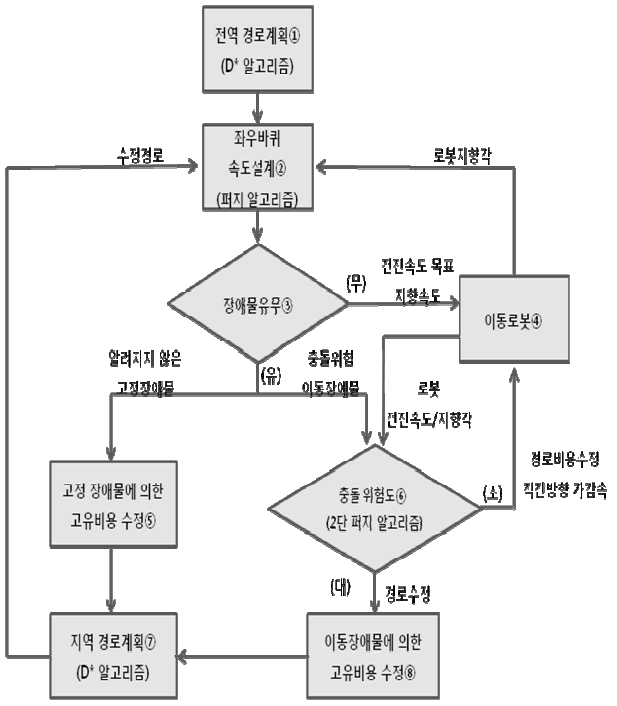


Fig. 2 Flow chart for suggested collision avoidance algorithm

변경하고 충돌 위험도를 평가하여 경로의 수정이나 혹은 이동로봇의 가/감속을 결정하는 것은 퍼지 알고리즘을 적용하였다. 또한 본 논문에서는 로봇이 초음파 센서등과 같이 제한된 근접범위만 탐지할 수 있는 장애물 감지 센서만을 가진 것으로 가정하였고, 자신의 현재 위치는 정확히 아는 것으로 가정하였다. 따라서 본 논문에서는 먼저 D* 알고리즘에 대해서 소개를 하고 이후 퍼지 알고리즘을 이용한 충돌회피 제어 알고리즘과 시뮬레이션을 통한 알고리즘의 검증 결과에 대해 논의 하고자 한다.

2. D* search algorithm 을 이용한 경로계획

본 논문에서는 서치 알고리즘 중의 하나인 D* 알고리즘을 사용하여 충돌회피 제어 알고리즘을 개발하고자 한다. D* 알고리즘은 다른 알고리즘에 비해서 많은 데이터 저장공간을 필요로 하는 단점이 있지만 전역 경로계획 뿐만 아니라 지역 경로계획도 동시에 가능하다.

따라서 예기치 못한 고정 장애물이나 움직이는 장애물을 회피하여 주어진 목적지까지 빠르게 도달할 수 있는 지역 경로계획을 신속히 알고리즘을 구현할 수 있는 장점을 가지고 있다. 이 경우 현재 지점부터 목표지점까지 전체 경로가 다시 계산되지 않고 기존 전역 경로 계획 아래 Fig. 3에서 화살표 방향 정보를 이용하여 현재 지점 주변 지역에서만 경로계획은 다시 하게 된다. 따라서 D* 알고리즘과 본문에서 제시하는 이동장애물 회피로직을 이용하여 알려지지 않은 고정장애물과 이동장애물이 있는 환경하에서 이동로봇의 충돌회피 주행제어 알고리즘을 제안하고자 한다.

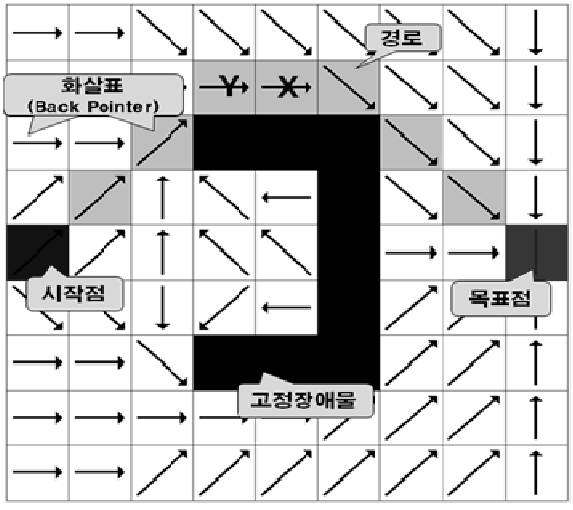


Fig. 3 An example of D* path planning

2.1 D* 알고리즘을 이용한 전역 및 지역 경로계획

D* 알고리즘은 아래 Fig. 3에서와 같이 시작점(Start state)에서 목표점(Goal state)까지의 경로비용(2.2절 참조)을 최소화하는 서치 알고리즘 중의 하나이다.

D* 알고리즘은의 전역경로계획은 지도데이터를 바탕으로 로봇이 출발하기 전에 이루어 질 수 있는데 Fig. 3의 각 셀의 화살표(Back Pointer)는 전역경로 계획된 결과를 나타낸다. 셀에서 화살표의 방향은 근처 셀 중에서 경로비용이 가장 작은 셀을 나타낸다. 즉 화살표는 각각의 셀에서 다음에 어떤 셀로 움직여야 하는지를 나타낸다. 전역 경로계획은 목표점에서 거꾸로 시작점을 찾아가는 Backward 서치 방법으로 이루어 진다. 지역경로계획은 로봇의 이동 중 새로운 고정 또는 이동장애물이 발견된 경우 기존 계획된 전역경로계획 결과를 바탕으로 새로운 장애물 근방의 고유비용(2.2절 참조)을 수정한 후 이를 바탕으로 주위 셀들의 경로비용과 화살표(Back Pointer)를 수정하게 된다.

Fig. 3에서 셀의 크기는 로봇의 크기 및 로봇의 위치를 알려주는 위치 센서 시스템의 정확도에 따라 달라 질 수 있을 것이다. 셀의 크기는 위치 센서가 정확할 경우 대략 로봇의 크기와 약간 큰 정도이면 충분히 만족할 만 할 것이다.

D* 알고리즘은 주어진 환경의 지도데이터가 틀렸을 경우와 움직이는 장애물이 있는 경우 새로운 환경 데이터를 기반으로 다시 맵을 구성해서 새로운 경로를 찾아야 하는 A* 알고리즘[15]과 달리 이미 계획된 전역경로계획을 기반으로 필요한 영역에서만 지역경로계획을 수행하므로 실시간으로 경로를 변경 계획하는 것이 용이하다.

본 논문에서 제안하는 충돌회피 주행알고리즘은 D* 알고리즘이 주어진 지도정보를 이용하여 전역경로 계획(①)을 수행하는 것으로부터 시작한다.

2.2 D* 알고리즘의 비용(cost) 함수

D* 알고리즘은 경로비용을 최소화하는 알고리즘이므로 D* 알고리즘을 효율적으로 사용하기 위해서 경로비용의 설계가 매우 중요한 역할을 한다. 본 논문에서도 이 경로비용을 창의적으로 설계하여 장애물 충돌회피 알고리즘에 사용하였다.

아래 식 (1)에서 $h(X)$ 는, 예를 들어 Fig. 3에서 상태 X까지 소요되는 총 경로비용이며 Y는 상태 X에 도달하기 바로 이전 상태를 의미하며, 식 (2)에서 $c(Y,X)$ 는 근접 경로비용으로 이전 상태 Y에서 현재 상태 X로의 경로비용을 의미한다. 근접 경로비용 비용 $c(Y,X)$ 는 식 (2)와 같이 설계되었다.

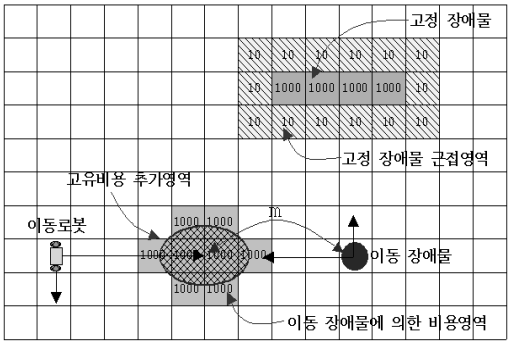


Fig. 4 Intrinsic costs for static and moving obstacles

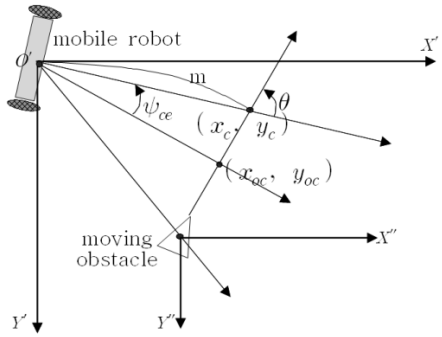


Fig. 5 Variable definition for development of moving obstacle collision avoidance

$$h(X)=h(Y)+c(Y,X) \tag{1}$$

$$c(Y,X)=A(Y,X)+I(X) \tag{2}$$

식 (2)에서 $A(Y,X)$는 상태 X와 상태 Y로 이동하는데 드는 비용이며 $I(X)$는 상태의 고유비용으로 인접한 고정 장애물과 이동장애물을 고려한 비용 함수이다. 아래에서는 Fig. 2의 ⑤와 ⑧블록에 해당하는 내용을 설명한다. 고정장애물에 의한 비용은 GM(Gradient Method)알고리즘[7]에서 사용한 것과 같이 고정 장애물과 거리의 함수로써 주어지게 되는데, 본 논문에서는 격자분할 공간을 사용하였기 때문에 고정 장애물이 존재하는 상태에는 매우 큰 값(1000)을 그리고 고정 장애물과 바로 인접 상태에는 작은 값(10)을 주었다.(Fig. 4 참조) 고정장애물과 인접한 상태에 고유비용을 부여한 이유는 가능하면 고정 장애물과 너무 근접하지 않고 이동로봇이 주행하도록 하기 위한 것이다.

이동장애물에 의한 비용함수는 Fig. 4와 같이 이동 장애물의 진행방향에 일정한 비용 영역을 부여한다. 이동장애물의 진행방향에 비용 영역을 부여하여 경로를 수정하는 이유는 이동로봇이 장애물을 피할 수 있는 어느 정도의 시간이 필요하기 때문인데 움직이는 장애물의 전진방향으로 비용영역을 설정하여 움직이는 장애물의 경로를 미리 우회하고자 한다.(Fig. 4참조)

위의 Fig. 4에서 이동장애물을 고려한 비용영역은 우선 이동장애물의 진행방향에 타원모양의 비용영역을 설정하고 타원의 영역이 조금이라도 포함된 격자에 고유비용을 부여하는 방식을 취하였다. 타원모양은 이동로봇과 이동장애물의 속도(v_o)에 따라 식 (3)~(5)에 의해서 그 크기가 결정되어 진다. 식 (4)와 (5)에서 α, β는 비례상수로 이동장애물의 속도에 비례하여 타원의 크기가 결정되어 진다.

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \tag{3}$$

$$a=\alpha\times v_o \tag{4}$$

$$b=\beta\times v_o \tag{5}$$

Fig. 4에서 이동장애물과 비용영역 중심 사이의 거리 m은 다음절에 나오는 퍼지 회피로직에 의해서 결정된다.

3. 퍼지로직을 이용한 충돌회피 결정로직

이동로봇이 주행 중 이동장애물을 만날 경우 본 논문에서는 인간의 이동장애물 회피 방법을 모사하는 세 가지의 행동을 하여 이동장애물과의 충돌을 회피하고자 한다. 첫 번째는 속도를 증가시켜 앞질러가기, 두 번째는 속도를 감소시키거나 혹은 정지하여 이동장애물이 지나가는 것을 기다리기, 세 번째는 경로를 수정하여 주행하는 것이다.

본 논문에서는 먼저 충돌위험도를 구하는 1단계 퍼지로직과, 1단계에서 구해진 충돌위험도와 자신의 속도를 이용하여 로봇이 3가지 중 하나의 회피결정을 하는 2단계 퍼지로직을 구성하였다. 즉 Fig. 2의 ⑥번 로직의 기능을 구성하였다.

이동 장애물과 로봇의 상대위치는 아는 것으로 가정하였다. 이 경우 장애물의 진행 방향과 속도는 한 타임 스텝전의 위치를 이용하여 근사적으로 구할 수 있다.

3.1 충돌위험도 퍼지로직

Fig. 5는 충돌위험도를 구하는 퍼지로직에 사용되는 변수들을 보여준다.

이동로봇이 이동장애물과의 회피로직에 사용되는 기본원리는 이동로봇과 이동장애물과의 사잇각(θ)과 이동로봇의 지향각 방향으로의 직선과 이동장애물 지향각 방향으로의 직선의 교점 (x_c, y_c)을 이용하는 것이다. 이 직선의 교점과 이동로봇의 거리가 2장에서 소

개한 이동장애물과 비용영역 중심 사이의 거리 m이 된다. 이동로봇이 교점 (x_c, y_c)에 도달하는 시간을 t_{col}라 할 경우 t_{col} 후에 이동장애물의 위치는 (x_{oc}, y_{oc})이며 이때 (x_{oc}, y_{oc})와 (x_c, y_c)에 의해서 정의되는 각은 ψ_{ce}이다. 본 논문에서는 사잇각(θ)와 ψ_{ce}를 입력으로 하는 퍼지 규칙을 이용하여 이동 장애물과의 충돌의 위험도를 결정하였다. Fig. 6은 충돌 위험도를 구하기 위한 입력 멤버 함수(θ, ψ_{ce})를 나타내며 Fig. 7은 충돌위험도의 출력 멤버함수이다. Table 1은 충돌 위험도를 구하기 위한 퍼지규칙을 나타낸다. 충돌 위험도에서의 부호와 크기는 충돌회피에 필요한 가/감속과 크기를 의미한다.

3.2 로봇의 충돌회피 행동결정 퍼지로직

3.1절의 퍼지규칙에서 결정된 충돌 위험도와 이동로봇의 좌우바퀴 속도(v_L, v_R)를 가지고 경로를 수정하여 이동장애물과의 충돌을 회피할지 아니면 가/감속을 통해서 충돌을 회피할지를 결정하게 되는데 Table 2는 충돌회피 행동결정을 위한 퍼지 규칙이다. 여기서 위험도는 충돌회피에 필요한 가/감속과 크기를 의미하기 때문에 현재 이동로봇의 바퀴속도를 고려하여 위험도를 줄일 수 있게 가/감속을 하며 가/감속만으로 위험도를 줄일 수 없는 경우에 경로를 수정하여 위험도를 줄일 수 있게 하였다. 예를 들면 바퀴의 속도가 빠르고 위험도에서 감속이 많이 필요한 경우에는 충분이 감속을 할 수 있기 때문에 감속만으로 위험도를 줄일 수 있지만 바퀴의 속도가 느리고 위험도에서 감속이 많이 필요한 경우에는 로봇이 더 이상 감속을 할 수 없기 때문에 경로수정을 통해 위험도를 줄이도록 하였다. Table 2에서 회색 영역이 경로수정이 필요한 영역이며 2장에서 소개한 이동 장애물에 의한 고유비용을 추가하고 경로를 수정하여 이동 장애물과의 충돌을 회피하고 그 외 부분에서는 가감속을 통해 충돌회피를 하게 되는데 그 때 속도는 아래 충돌회피 퍼지 규칙에 의해서 결정된다. 경로수정의 경우는 Fig. 2의 ⑦번 블록에 해당하는 D*지역경로 수정 알고리즘으로 이루어진다. Table 2의 퍼지규칙의 입력인 좌우바퀴의 속도(v_L, v_R)와 출력인 좌우바퀴의 속도(v_L, v_R)는 4장의 Fig. 11의 멤버함수로 결정된다.

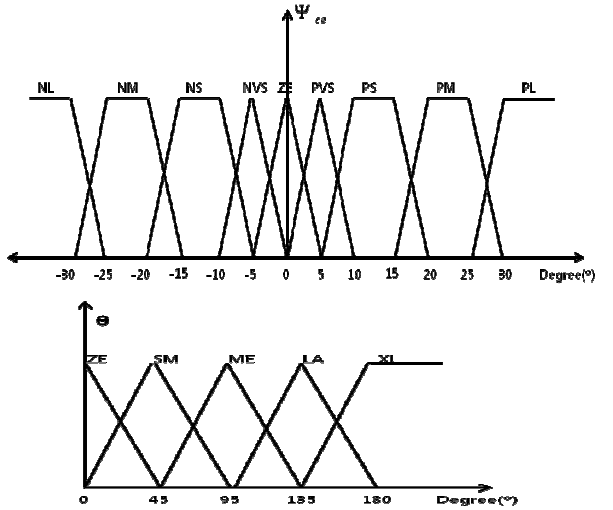


Fig. 6 Input membership function of collision danger(ψ_{ce}, θ)

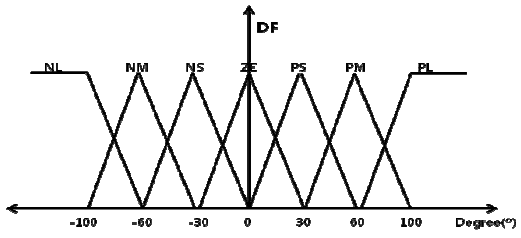


Fig. 7 Output membership function of collision danger

4. Fuzzy Rule 을 이용한 이동로봇 좌우바퀴 속도설계

앞 절들에서 이동로봇의 원하는 경로 및 전진속도가 설계되었다 해도 실제 로봇의 주행을 위해

Table 1 Fuzzy logic for collision danger

ψ_{ce} \ θ	ZE	SM	ME	LA	XL
NL	PL	NS	ZE	NS	PL
NM	PL	NM	NS	NM	PL
NS	PL	NL	NM	NL	PL
NS	PL	NL	NM	NL	PL
ZE	PL	NL	NL	NL	PL
PS	PL	NL	NL	NL	PL
PS	PL	NM	PM	NM	PL
PM	PL	PM	PM	PM	PL
PL	PL	PM	PS	PM	PL

Table 2 Fuzzy logic for collision avoidance

위험도 \ 바퀴속도	NL	NS	ZE	PVS	PS	PM	PL	PX	PXL
NL						ZE	PVS	PVS	PM
NM	경로수정 영역				ZE	PVS	PS	PS	PL
NS				ZE	PVS	PS	PM	PM	PX
ZE	NL	NS	ZE	PVS	PS	PM	PL	PX	PXL
PS	NS	ZE	PVS	PS	PM	PL	PX	PX	PXL
PM	ZE	PVS	PS	PM	PL	PX	PXL	PXL	PXL
PL				경로수정 영역					

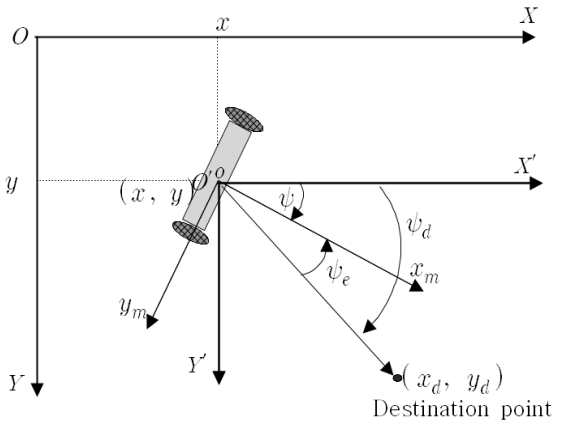


Fig. 8 Variable definition for mobile robot maneuvering

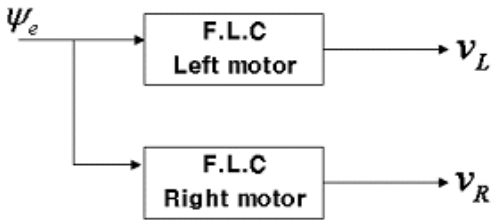


Fig. 9 Fuzzy logic concept for designing wheel velocities of mobile robot

서는 경로 및 전진속도를 추종하는데 필요한 좌우바퀴 속도명령 (v_L, v_R)을 결정하여야 한다. 즉 Fig. 2 의 ②번 로직 역시 퍼지로직을 이용하여 구현하였다. 서치 알고리즘 에서 구한 추종해야 할 경로는 격자의 중심들을 연결 하는 직선들의 형태로 만들어지므로 부드럽지 않다. 본 논문에서는 퍼지규칙[12]을 이용하여 이동로봇의 좌우 바퀴 속도를 설계하여 로봇이 급격한 방향수정이 필요한 경우 정상속도보다 느린 속도로 운행하면서 경로를 부드럽게 추종하게 하였다.

Fig. 8은 좌우바퀴 속도를 설계하기 위해서 필요한 좌표계와 변수들을 보여준다.

Fig. 8에서 $O'-X'Y'$ 는 지구고정 좌표계를 이동로봇의 중심으로 이동시킨 것이며 $o-x_m y_m$ 는 이동로봇의 몸체에 고정된 좌표계이다.

좌우바퀴 속도를 정하기 위해서 퍼지규칙의 입력으로 이동 로봇의 지향각(ψ)과 목표 지향각(ψ_d)의 차인 지향각 오차(ψ_e)를 사용하여 구현하게 되는데 목표 지향각과 지향각 오차는 각각 식 (6)와 식 (7)로 표시된다.

$$\psi_d = \tan^{-1}\left(\frac{y_d - y}{x_d - x}\right) \tag{6}$$

$$\psi_e = \psi_d - \psi \tag{7}$$

식 (6)에서 이동 목표지점좌표(x_d, y_d)는 다음 이동

Table 3 Fuzzy logic for forward velocity design

ψ_e	v_R	v_L
NX	PS	NL
NL	PM	NS
NM	PM	PS
NS	PL	PM
ZE	PX	PX
PS	PM	PL
PM	PS	PM
PL	NS	PM
PX	NL	PS

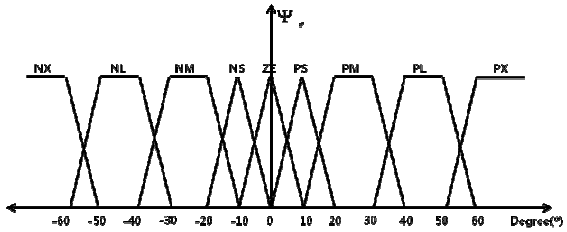


Fig. 10 Input fuzzy membership function of ψ_e

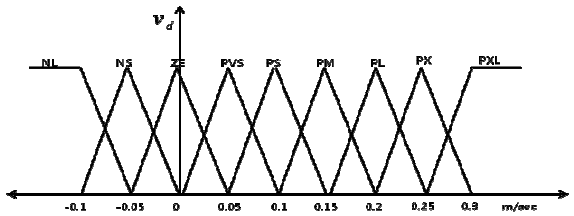


Fig. 11 Output Fuzzy Membership Function of v_R and v_L

격자의 중심점이다.

Fig. 9는 좌/우측 바퀴에 적용되는 두 개의 퍼지규칙의 개념을 나타낸다.

퍼지규칙의 입력인 지향각 에러(ψ_e)와 출력인 좌우측 바퀴의 전진방향 목표속도(v)의 퍼지화를 위한 멤버 함수는 Fig. 10~11과 같다.

지향각 에러(ψ_e)에 대한 퍼지 규칙은 Table 3과 같다. 이동로봇 제어의 최종입력은 바퀴에 연결된 모터에 주어지는 토크명령이다. 이를 위해 원하는 바퀴속도 (v_R, v_L)를 구현하는 토크입력을 PID 제어기 형태로 구현하였다.

5. 시뮬레이션

5.1 시뮬레이션에 사용된 이동로봇의 운동방정식

설계된 알고리즘의 성능검증을 위해 세그웨이 형태의 이동로봇을 가정하여 주행 시뮬레이션을

수행하였다.

식 (8)~(10)은 이동로봇의 기구학식을 보여준다.

$$\dot{x} = u\cos\psi \tag{8}$$

$$\dot{y} = u\sin\psi \tag{9}$$

$$\dot{\psi} = w \tag{10}$$

식 (11)~(12)은 이동로봇의 동역학 식을 보여주는데 T_R, T_L 은 이동로봇의 양쪽 바퀴에 의해서 발생하는 토크를 나타낸다.

$$M\dot{u} = \frac{T_L}{r} + \frac{T_R}{r} \tag{11}$$

$$I\dot{\omega} = \frac{l}{2r}T_L - \frac{l}{2r}T_R \tag{12}$$

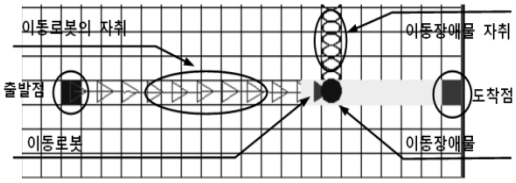
제시된 이동로봇의 기구학식 및 동역학식들은 주행제어 시뮬레이션에서 이동로봇의 궤적계산을 위해 사용되었다.

5.2 시뮬레이션 결과

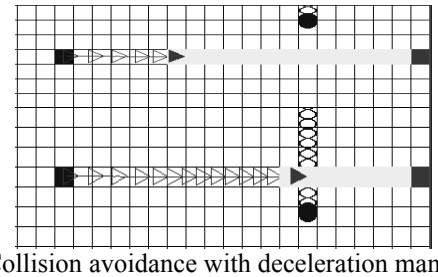
D* 알고리즘과 퍼지규칙을 이용하여 전역, 지역 경로계획이 동시에 가능한 이동로봇의 충돌회피 주행제어 알고리즘은 Python[16]으로 개발한 시뮬레이션 프로그램을 통해서 검증하였다. 또한 이동장애물의 속도 및 가속, 선회 성능이 이동로봇보다 빠른 경우에는 충돌을 회피하지 못하는 경우가 많기 때문에 이동장애물의 속도 및 가속, 선회 성능은 이동로봇을 능가 하지 못하는 경우로 가정하였다.

Fig. 12는 고정장애물이 없고 이동 장애물만 있는 경우 이동로봇이 감속을 통하여 이동 장애물이 지나가기를 기다린 후 주행하는 시뮬레이션 결과이다. Fig. 12(a)는 감속 알고리즘이 없을 경우 이동로봇과 이동 장애물이 충돌 하는 모습을 보여주는 것이며 Fig. 12(b)는 감속하여 이동로봇이 이동장애물을 피하는 과정을 보여준다. 삼각형 모양(▶)이 이동로봇이고 동그란 모양의 것(●)이 이동장애물이다. 이동로봇과 이동장애물의 자취는 일정한 시간간격의 위치를 표현한 것이다. 이 자취를 살펴보면 이동로봇이나 이동장애물의 속도변화를 볼 수 있는데 이 자취의 간격이 넓을수록 속도가 빠른 것이다. 따라서 Fig. 12(b)에서 이동로봇의 자취를 보면 이동장애물의 근처에서 자취의 간격이 줄어드는 것을 확인 할 수 있다. 이것은 이동로봇이 속도를 줄여 이동장애물이 지나가는 것을 기다렸다가 이동장애물이 지나간 후에 다시 원래의 속도로 목표점까지 간 것이다.

Fig. 13은 고정장애물이 없고 이동 장애물만 있는 경우 이동로봇이 가속을 통하여 이동 장애물을 앞질러

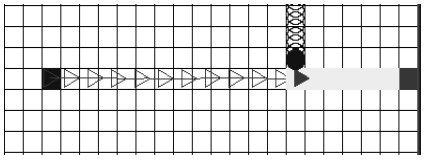


(a) Result without deceleration

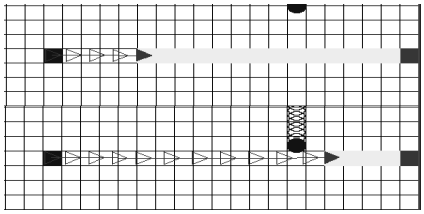


(b) Collision avoidance with deceleration maneuver

Fig. 12 Simulation of moving obstacle avoidance by deceleration



(a) Result without acceleration



(b) Collision avoidance with acceleration maneuver

Fig. 13 Simulation of collision avoidance with acceleration

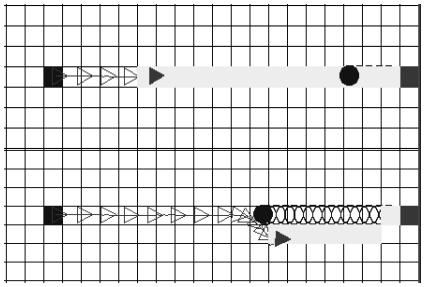
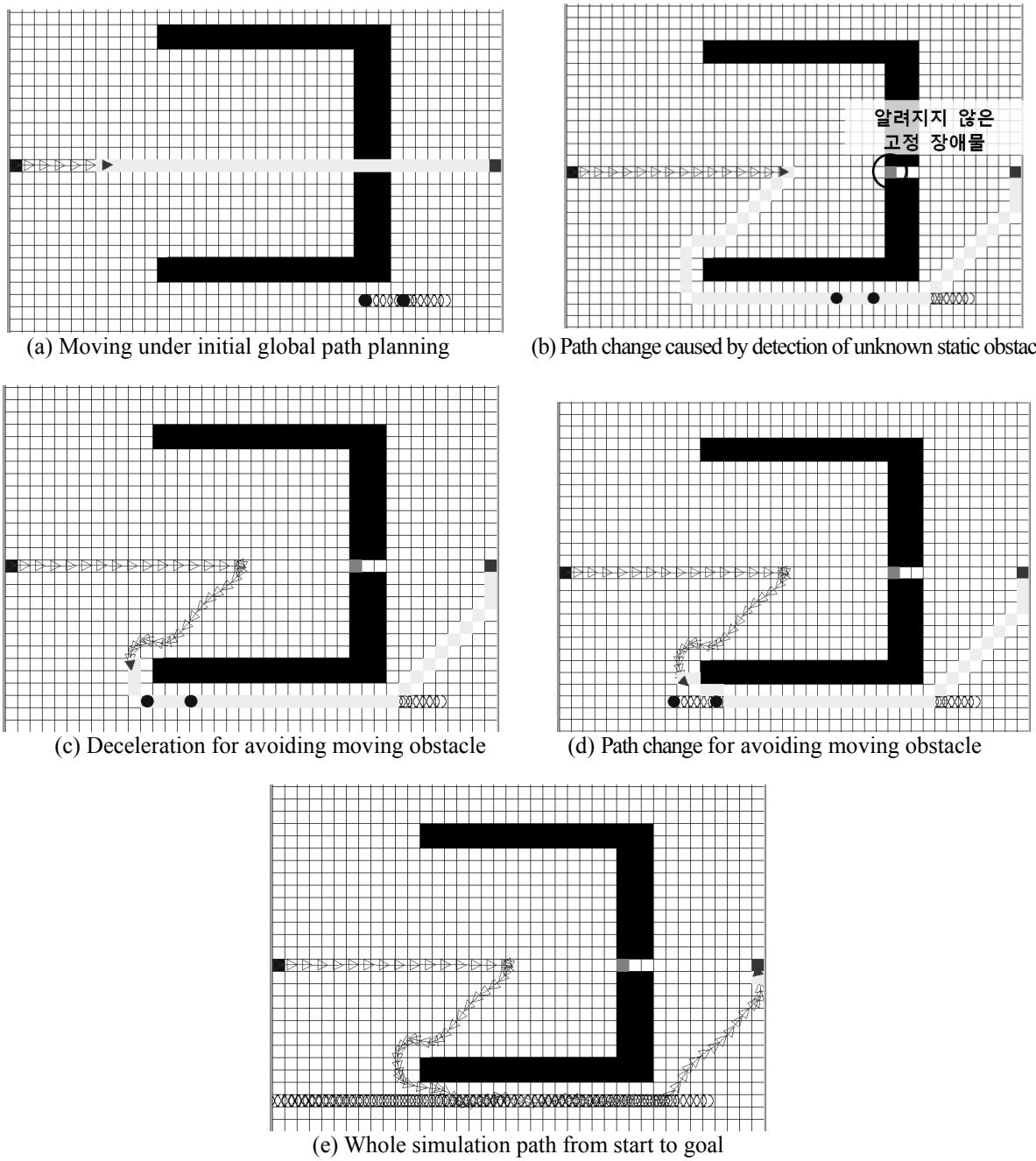


Fig. 14 Simulation of collision avoidance with path change

주행하는 시뮬레이션 결과이다. Fig. 13(a)는 가속 알고리즘이 없을 경우 이동로봇과 이동 장애물이 충돌하는 모습을 보여주는 것이며 Fig. 13(b)는 가속하여 이동로봇이 이동장애물을 앞질러 가는 과정을 보여준다. Fig. 13(b)에서 이동로봇의 자취를 보면 이동장애물의 근처에서 자취의 간격이 넓어지는 것을



(a) Moving under initial global path planning
(b) Path change caused by detection of unknown static obstacle
(c) Deceleration for avoiding moving obstacle
(d) Path change for avoiding moving obstacle
(e) Whole simulation path from start to goal

Fig. 15 Simulation of moving and static obstacle avoidance from start to foal position

확인할 수 있다. 이것은 이동로봇이 이동장애물을 앞질러 가기 위해서 속도를 증가시켰기 때문이다.

Fig. 14는 고정 장애물이 없고 이동 장애물만 있는 경우 이동로봇이 경로를 수정하여 장애물을 회피하는 시뮬레이션 결과이다. 가/감속만으로 이동 장애물을 회피할 수 없기 때문에 경로를 수정하므로 장애물을 회피하고 있다.

Fig. 15는 알려지지 않은 고정 장애물에 의한 경로계획 수정과 이동장애물에 의한 회피가 복합적으로 있는 시뮬레이션 결과이다. Fig. 15(a)는 최초에 주어진 지도 정보에 의해서 계획된 경로에 의해서 주행 하는 모습이고 Fig. 15(b)는 주행 중에 알려지지 않은 고정 장애물에 의해서 경로를 수정하여 주행하는 모습이다. 또한 Fig. 15(c)는 주행 중에 이동 장애물을 발견하여 감속하여 속도를 줄여 이동 장애물이 지나가기를 기다리고 있는 모습이다. Fig. 15(d)는 또다른 이동 장애물을 발견하여 경로를 수정하여 충돌을 회피하는

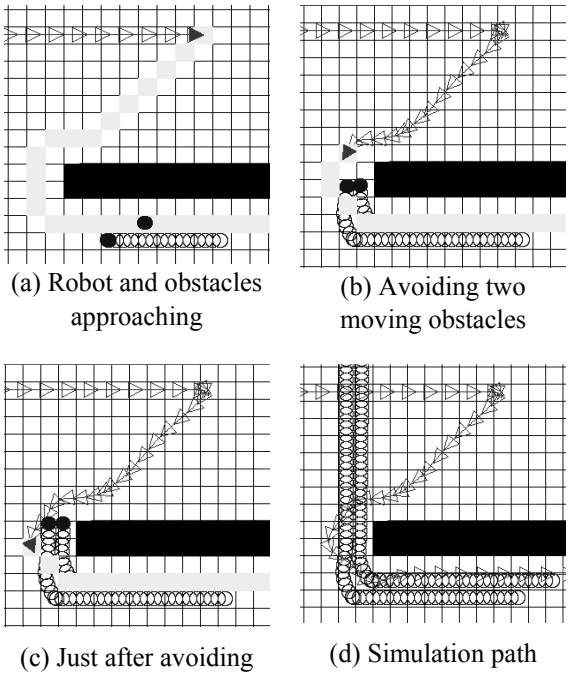


Fig. 16 Simulation of local collision avoidance for two obstacles moving together

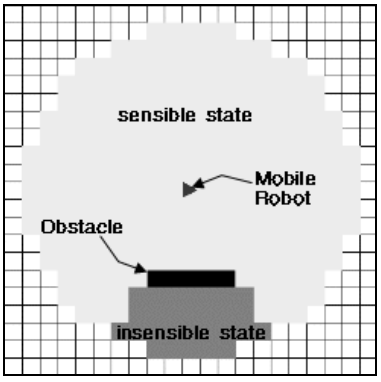


Fig. 17 Obstacle detection range of sensor used in simulation

모습이다. Fig. 15(e)는 목표점에 도착한 모습이다. Fig. 16은 역시 두 개의 이동 장애물이 로봇과 만나는 상황을 시뮬레이션한 결과로 (a), (b), (c), (d)가 순차적인 충돌회피 장면을 보여준다. 이때 장애물 감지 센서는 비젼이나 초음파센서처럼 장애물에 가려진 부분은 탐지하지 못한다고 가정한다. 아래 Fig. 17은 고정장애물이 있을 경우 이동로봇을 중심으로 계측이 가능한 영역과 계측이 불가능한 영역의 예를 보여주며 sensing range는 이동로봇의 최대 속도 및 가/감속 성능을 고려하여 결정되었으며 sensing range가 짧아 질 경우에는 위에서 제시한 위험도 및 충돌회피 행동결정 퍼지 로직의 수정이 필요하게 된다.

6. 결 론

본 논문에서는 D* 알고리즘과 퍼지규칙을 이용하여 전역, 지역 경로계획이 동시에 가능한 이동로봇의 충돌회피 주행 제어 알고리즘을 제안하였다. 일반적으로 전역경로계획과 지역 경로계획을 구분해서 사용하는 경우가 많은데 D* 알고리즘을 사용할 경우 예상치 않은 고정장애물 및 이동장애물에 의한 주변 환경이 변해도 영향을 받는 지역에서만 경로수정이 빠르게 가능하기 때문이다. 또한 인간의 행동을 모사하여 이동장애물과 충돌 위험이 있을 때 기다리기, 멈춰서기, 빨리가기, 그리고 이동장애물의 진행방향으로 비용 영역을 주어 경로 수정하기 등을 이용하여 충돌회피 주행 알고리즘을 제안하였다.

후 기

이 논문은 2007 년도 충남대학교의 학술연구비의 지원에 의해서 연구되었음.

참고문헌

(1) Khatib, O., Spring 1986, "Real-Time Obstacle Avoidance for manipulator and Mobile Robots," *The International Journal of Robotic Research*, MIT Press, Cambridge, pp. 90~98.
(2) Rimon, E. and Koditschek, E. E., October 1992, "Exact Robot Navigation Using Artificial Potential Functions," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 5.
(3) Yoo, H. I. and Kang, B. S., 2008, "Precise Motion Control of a Mobile Robot Based on Potential Field Method," *The KSME 2008 Dynamics and Control spring annual meeting*, pp.177~180
(4) Park, M. G. and Lee, M. C., 2003, "A New Technique to Escape Local Minimum in Artificial Potential Field Based Path Planning," *KSME International Journal*, Vol.17, No.12, pp. 1876~1885
(5) Park, M. G. and Lee, M. C., 2003, "A New Technique to Escape Local Minimum in Artificial Potential Field Based Path Planning," *KSME International Journal*, Vol.17, No.12, pp. 1876~1885
(6) Qu, Y.-H. Pan, Q. and Yan, J.-G., Nov., 2005, "Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms," *Industrial Electronics Society*, 2005. IECON 2005. 31st Annual Conference of IEEE, Volume , Issue , 6~10.
(7) Bruce, J. and Veloso, M., October 2002, " Real-Time Randomized Path Planning for Robot Navigation," *Proc. of the 2002 IEEE/RSJ Int. Conference on Intelligent Robots and Systems EPFL*, Lausanne, Switzerland, pp. 2383~2388.

(8) Konolige, K., 2000, "A Gradient Method for Real-time Robot Control," *Proc. of International Conf. on Intelligent Robots and Systems*, pp. 639~646.
(9) Borenstein, J. and Koren, Y., June 1991, "The Vector Field Histogram - Fast Obstacle Avoidance for mobile Robots," *IEEE Journal of Robotics and Automation* Vol. 7, No 3, pp. 278~288.
(10) Simmons, R., 1996, "The Curvature-Velocity Method for Local Obstacle Avoidance," *Proc. of International Conf. on Robotics and Automation*.
(11) Kim, H. J., Shim, D. H. and Sastry, S., 2002, "Nonlinear Model Predictive Tracking Control for Rotorcraft-Based Unmanned Aerial Vehicles," *Proceedings of the American Control Conference*, Anchorage, AK May 8-10.
(12) Ogren, P. and Leonard, N. E., 2003, "A Convergent Dynamic Window Approach to Obstacle Avoidance," *IEEE Transactions on Robotics and Automation*,.
(13) Stachniss, C. and Burgard, W., October 2002, "An Integrated Approach to Goal-Directed Obstacle Avoidance Under Dynamic Constraints for Dynamic Environments," *Proceedings of the 2002 IEEE/RSJ, intl, Conference on Intelligent Robots and Systems EPFL*, Lausanne, Switzerland.
(14) Fox, D., Burgard, W. and Thrun, S., 1997, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics and Automation*, Vol. 4, No. 1.
(15) Daios, E. P. and Maravillas O. A., Jr., 2002, "Cooperative Mobile Robots with Obstacle and Collision Avoidance Using Fuzzy Logic," *Proc. of the 2002 IEEE International Symposium on Intelligent Control*, Vancouver, Canada, October 27-30, pp. 75~80.
(16) Russell, S. and Norvig, P., 2003, "Artificial Intelligence A Modern Approach," 2nd Ed. Prentice Hall.
(17) Stentz, A., 1994, "Optimal and Efficient Path Planning for Partially-known Environments," *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA '94, Vol. 4, May, pp. 3310~3317.
(18) Aurenhammer, F. and Diagrams, V., 1991, "A Survey of a Fundamental Geometric Data Structure." *ACM Computing Surveys*, Vol. 23, No. 3, pp. 345~405.
(19) Lutz, M., 2006, "Programming Python," O'REILLY.
(20) Ess, A., Leibe, B., Schindler, K. and Van Gool, L., 2009, *Moving Obstacle Detection in Highly Dynamic Scene IEEE International Conference on Robotics and Automation (ICRA'09)* Kobe, Japan 2009