이동 로봇을 위한 동적 실내 환경에서의 효율적인 온라인 경로 계획 알고리즘

Efficient Online Path Planning Algorithm for Mobile Robots in Dynamic Indoor Environments

강 태 호*, 김 병 국
(Tae-ho Kang¹ and Byung-Kook Kim¹)
¹KAIST

Abstract: An efficient modified D* lite algorithm is suggested, which can perform online path planning for mobile robots in dynamic indoor environment. Online path planning should plan and execute alternately in a short time, and hence it enables the robot avoid unknown dynamic obstacles which suddenly appear on robot's path. Based on D* Lite algorithm, we improved representation of edge cost, heuristic function, and priority queue management, to build a modified D* Lite algorithm. Performance of the proposed algorithm is revealed via extensive simulation study.

Keywords: mobile robot, online path planning, D* Lite algorithm

I. 서론

과학 기술의 비약적인 발전에 힘 입어, 최근 로봇 연구 분야는 갈수록 다양해지고 있다. 특히 로봇의 작업 환경이나 로봇의 크기 등의 다양성이 크게 증가함에 따라 일반 목적의 시스템(general-purpose system) 뿐 아니라 특정 목적 시스템(specific-purpose system)이라고 할 수 있는 임베디드 시스템(embedded system)이 활발히 연구되고 있다. 임베디드 시스템은 문자 그대로 '특정 목적을 수행하기 위해' 설계된 시스템으로, 대표적인 예를 들면 극지 탐사를 위한 로봇이나 무인 주행 차량 등이 있다.

이와 같이 탐사를 목적으로 하는 로봇의 대부분의 경우, 탐사 지역에 대한 모든 정보를 가진 상태로 수행을 할 수는 없기 때문에 새롭게 알게 되는 지역 상황에 대처할 능력이 필요하게 된다. 이전에 알고 있지 못한 장애물이 나타났을 때 신속하게 경로를 다시 계획하여 이것을 회피하면서 목표를 달성해야 하는 것이다.

이러한 기본적인 장애물 회피와 경로 계획에 관한 연구는 다양하게 이루어져 왔다. 경로계획은 오프라인 경로 계획과 온라인 경로 계획으로 구분된다. 경로를 오프라인으로 계획한다는 것은, 로봇이 수행을 시작하기 전에 미리 경로를 계획해놓음으로써 보다 정확하고 최적의 경로를 계획할 수 있으나 시간이 오래 걸린다는 점, 동적인 장애물에 대처가 불가능하다는 등의 단점이 있다. 다만 여러 개의 이동 로봇이 주어진 지역을 비교적 빈틈없이 탐색한다거나[1], 소요 시간 보다는 경로의 최적화에 더욱 중점을 두는 경우[2], 혹은 해

당 지형과악이 목적인 경우[3]에는 오프라인 경로 계획이 상대적으로 강점을 지닐 수 있다.

온라인 경로 계획으로는, 장애물의 척력과 목표점의 인력을 이용한 퍼텐셜 장(potential field) 방법[4], 장애물까지의 거리를 히스토그램(histogram)으로 나타내는 벡터 장 히스토그램[5], 또는 그것의 개선된 방법[6]등이 연구되었다. 그러나 이러한 온라인 경로 계획 알고리즘은 극소값을 가지거나[7] 무한 루프에 빠지는 등[8]의 약점이 있었기 때문에 보다 개선된 알고리즘의 필요성이 대두되었다.

또한, 초기에 존재하지 않던 새로운 장애물이 나타났을 때, 그 지점에서 처음부터 다시 탐색을 시작하는 방법[9]도 연구되었으며, 이 방법은 미로처럼 복잡한 지형에서도 경로 계획이 가능하다는 강점이 있으나 처음부터 다시 시작한다는 점이 부담스럽다는 문제점이 있다. 실제 로봇으로 많은 실험을 수행하여 그 효율성을 입증된 D* 알고리즘[10]도 있다. 하지만 D* 알고리즘은 분석과 확장의 측면에서 어려움이 있다는 것이 개선되어야 할 사항으로 지적되기도 한다. 2005년에 발표된 D* Lite 알고리즘[11]은 D* 알고리즘의 단점을 보완했다는 장점을 가지고 있다.

본 논문의 목적은 D* Lite 알고리즘의 성능을 더욱 개선하여 효율적인 온라인 경로계획 알고리즘을 제안하는 것이다. 본 연구에서 살펴보려는 문제는 다음과 같다. 시작점과 목표점, 그리고 센서에 의한, 대상 지형에 대한 부분적인 정보만을 가지고 두 바퀴 이동 로봇의 최단 경로를 가능한 한 짧은 시간 안에 계획한다. 아울러 목표점으로 이동하는 도중에 새로운 장애물을 만났을 때에는 그 장애물을 회피하도록 빠르게 경로를 다시 계획하고, 이 과정을 반복하여 로봇이 목표점에 신속히 도착할 수 있도록 한다. 이 과정에서 새로운 휴리스틱 함수, 우선순위 큐 관리 함수, 그리고 엣지 비용 저장 방식을 적용하여 D* Lite 알고리즘의 성능을 향상시키고자 한다.

본 논문의 구성은 다음과 같다. II 장에서는 기존 D* Lite 알고리즘을 비롯하여 온라인 경로 계획 알고리즘 동작의 설

* 책임저자(Corresponding Author)

논문접수: 2011. 2. 10., 수정: 2011. 3. 22., 채택확정: 2011. 4. 19.

강태호, 김병국: KAIST 전기 및 전자공학부

(th.kang0501@gmail.com/bkkim@ee.kaist.ac.kr)

※ 본 연구는 (고려대학교 로봇자율주행기술 전문인력양성센터를 통한) 지식경제부/한국산업기술진흥원 융복합형 로봇전문인력양성사업의 지원으로 수행되었음.

명을 통해 전반적인 이해를 돕고, III 장에서는 이동 로봇이 경로를 계획 추종함에 있어서 센서 처리부와 모터 제어부에 대해 기술한다. IV 장에서는 앞에서 살펴본 알고리즘의 시뮬레이션을 통해 결과를 분석하고, V 장에서는 본 연구의 결론과 추후 과제에 대해 알아본다.

II. 온라인 경로 계획 알고리즘

1. D* Lite 알고리즘[11]

다양한 온라인 경로 계획 알고리즘 중 D* Lite 알고리즘은 특히 증분(incremental) 탐색과 휴리스틱(heuristic) 탐색이 같이 사용된 대표적인 알고리즘이다. 휴리스틱 탐색이란 임의의 위치에서 목표점까지의 거리라는 형태의 정보를 이용하여 탐색 방향을 정하는 것이다.

몇 년 전부터 실제 로봇의 온라인 경로 계획에서 널리 활용되고 있는 D* 알고리즘[10]이 이해하기에 어려움이 있고, 알고리즘의 확장성이 떨어진다는 단점에 착안하여 개발된 D* Lite 알고리즘[11]은 몇 가지 중요한 거리 개념을 열쇠값으로 사용한다.

가장 기본이 되는 값은 $g(s)$ 으로, 이것은 목표점으로부터 어떤 격자 s 까지의 거리를 나타낸다. 또 다른 기본 값인 $rhs(s)$ 는 D* Lite 알고리즘의 가장 큰 특징 중 하나인 증분 탐색이라는 특징을 나타낸다.

$$rhs(s) = \begin{cases} 0 & \text{if } s = s_{start} \\ \min_{s' \in Pred(s)} (g(s') + c(s', s)) & \text{otherwise} \end{cases} \quad (1)$$

위에서 볼 수 있듯이 격자 s 가 시작점이 아닌 경우에는 그 주변 8개 격자의 g 값과 c 값을 기반으로 하는 값을 $rhs(s)$ 가 가지게 된다. 다시 말해서 이전 탐색 주기의 값을 이용하여 현재 탐색 주기의 효율성을 높이는 것이다.

$c(s, s')$ 는 s 에서 s' 까지의 비용을 나타낸다.

$$c = \begin{cases} 1 & \text{if traversable in } x, y \text{ axis} \\ \sqrt{2} & \text{if traversable in diagonal axis} \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

직관적으로 생각할 수 있는데, 가로, 세로 방향인 경우에는 이동에 1의 비용이 필요하고 대각선 방향인 경우에는 $\sqrt{2}$ 의 비용이 필요하며, 이동 가능하지 않은 경우에는 무한대의 값을 가지게 된다.

$Pred(s)$ 는 현재 주기의 s 로 이동할 수 있는, 이전 주기의 모든 격자를 나타낸다. 반대로 $Succ(s)$ 는 현재 주기의 s 에서 다음 주기에 이동할 수 있는 모든 격자를 나타낸다. 개념상으로는 다르지만 격자 자체로 보면 이 둘은 결국 8개의 동일한 격자의 집합이다.

은 열쇠값 수정자(key modifier)를 말한다. 이동 로봇이 격자 한 칸씩 움직일 때마다 나중에 추가되는 우선순위 큐의 항목들은 불가피하게 작은 열쇠값을 가지게 되는데, 열쇠값 수정자를 활용하면 이러한 문제를 해결할 수 있다.

각 격자가 가지는 열쇠값($k1, k2$)는 다음과 같은 값을 지니게 된다.

$$\begin{aligned} k1 &= \min(g(s), rhs(s)) + h(s_{start}, s) + k_m \\ k2 &= \min(g(s), rhs(s)) \end{aligned} \quad (3)$$

$k1$ 은 $g(s)$ 와 $rhs(s)$ 중 작은 값에다가 휴리스틱 값, 그리고 열쇠값 수정자 값까지 더한 값을 가진다. $k2$ 는 일종의 보완적인 값으로, 두 개 이상의 격자가 동일한 $k1$ 값을 가질 때 우선 순위를 정하기 위해 사용한다. 여기에 보이는 $h(s_{start}, s)$ 는 휴리스틱 탐색을 위한 값으로 시작점에서 특정 격자 s 까지의 거리를 나타낸다. 구체적인 값은 아래에서 설명한다.

D* Lite 알고리즘은 세 가지 중요한 개념을 기반으로 한다.

- 체비셰프 거리(Chebyshev distance) 기반 휴리스틱 함수
- 이진 최소 힙을 이용한 우선순위 큐 관리[12]
- 4차원 구조의 엣지 비용

아래에서 각 개념을 간략히 살펴볼 수 있다. 그 다음에 각각을 개선시킨 후의 개념에 대해 설명한다.

1.1 체비셰프 거리 기반의 휴리스틱 함수

체비셰프 거리는 아래의 식을 통해 계산할 수 있다.

$$h(i, j) = \max(|s_x - i|, |s_y - j|) \quad (4)$$

즉, 기준점으로부터 임의의 격자까지의 가로와 세로의 거리 차이 중 큰 값을 휴리스틱 값으로 정하는 것이다. 이와 같은 방법을 사용하면 가로나 세로 방향 이동이나 대각선으로의 이동이 동일한 값을 가지게 된다.

1.2 이진 최소 힙을 이용한 우선순위 큐 관리

D* Lite 알고리즘은 이진 최소 힙(binary min heap)을 사용하여 우선순위 큐에 항목을 삽입하거나 제거한다. 그림 1의 이진 최소 힙의 각 항목이 가지는 값은 네 가지로 구성이 되어 있음을 알 수 있는데, [$k1, k2$]는 위에 언급했던 첫 번째와 두 번째 열쇠값을 뜻하고, (x, y)는 각 격자의 전체 맵 상에서 좌표를 나타낸다. 각 격자는 열쇠값에 따라 우선순위 큐에서 정렬되고, 정렬된 상황에서 이동 가능한 격자의 좌표값을 탐색한다.

이진 최소 힙의 규칙은, 부모(상위) 노드의 열쇠값이 자식(하위) 노드의 열쇠값보다 항상 작다는 것이다. 이것 외에는 다른 규칙이 없기 때문에 같은 계층에 있는 노드 사이에는 정해진 관계가 없다. 이것을 ‘부분적으로 정돈되어 있다 (partially ordered)’라고 표현한다.

D* Lite 알고리즘의 우선순위 큐는 크게 삽입, 추출, 제거라는 세 가지 기능이 필요한데 각 기능을 복잡도 측면에서 살펴보면 삽입은 $O(\log(Nx*Ny))$, 추출도 $O(\log(Nx*Ny))$ 의 복잡도를 가진다[8]. 그러나 제거의 경우, 기능 자체의 복잡도는

그림 1. 이진 최소 힙의 예제.

Fig. 1. Example of binary minimum heap.

삽입, 추출과 동일하지만 제거를 하려는 특정 항목이 이진 최소 힙에 존재하는지를 탐색하는 비용이 $O(N_x \times N_y)$ 를 가진다는 단점이 있다. 여기에서 N_x 는 전체 맵의 최대 x 값, N_y 는 최대 y 값을 나타내고, 결론적으로 $N_x \times N_y$ 는 우선순위 큐의 전체 격자의 수를 나타내게 된다.

1.3 4차원 구조의 엣지 비용(Edge cost)

D* Lite 알고리즘에서는 하나의 격자에서 다른 인접한 격자로 이동할 때 그 엣지 비용을 $c(u, s')$ 로 나타낸다. 이는 다시 쓰면 $c(x_u, y_u, x_{s'}, y_{s'})$ 이 되는데 $x_{s'} \in N_x, y_{s'} \in N_y$ 이기 때문에 이를 MATLAB 등으로 구현하게 되면 4차원의 구조가 된다.

2. Modified D* Lite 알고리즘

2.1 개선된 세 가지 주요 개념

위에서 살펴보았던 세 가지 주요 개념은 다음과 같이 새롭게 개선될 수 있다.

- 유클리드 거리(Euclidean distance) 기반의 휴리스틱 함수 [13].

앞서 언급한 것과 같이 D* Lite 알고리즘은 실제 거리와 차이가 있는 체비셰프 거리 계산을 했기 때문에, 실제 로봇으로 실험을 하기 위해서는 이를 보완할 필요가 있다. 제안하는 휴리스틱 함수를 생각할 때 주의할 점은 지형이 격자 구조로 되어있다고 생각하기 때문에, 로봇은 격자 하나 단위로만 움직일 수 있고, 대각선이 아닌 비스듬한 사선 방향으로로는 이동할 수 없다는 것이다. 그러므로 기준점과 계산하려는 격자까지의 거리 중에 대각선으로 움직일 수 있는 최대 거리와, 남은 거리를 가로나 세로 방향으로 이동하는 비용의 합을 격자구조에서의 유클리드 거리 기반의 휴리스틱 함수로 다음과 같이 설정한다.

$$h(i, j) = \sqrt{2} \times \min(\text{abs}(s_x - i), \text{abs}(s_y - j)) + 1 \times \text{abs}(\text{abs}(s_x - i) - \text{abs}(s_y - j)) \quad (5)$$

k1 \ k2	(magnitude order...)	
1	(1.4, 5.2)	(1.7, 2.4)
2	(2.1, 1.9)	(2.4, 3.1)
3		
...		
$2 \times \max(N_x, N_y)$		

그림 2. 2차원 구조를 이용한 우선순위 정렬 방법.

Fig. 2. Priority managing method using 2-dimensional structure.

그림 3. 3차원 구조를 가지는 엣지 비용. 숫자는 index d.

Fig. 3. Edge cost of 3-dimensional structure.

- 2차원 구조의 우선순위 큐 관리 함수

기존 D* Lite 알고리즘은 이진 최소 힙을 데이터 구조로 사용함으로써 특정 항목을 제거할 때 $O(N_x \times N_y)$ 의 복잡도를 가지게 되는 단점이 있었다. 본 연구에서는 그림 2와 같이 2차원 구조를 이용한 우선순위 정렬 방법을 제안한다. 이것은 부분적으로만 정렬된 이진 최소 힙과는 달리 모든 항목이 우선순위에 따라 정렬이 되어있기 때문에 특정 항목을 제거할 때 $O(\max(N_x, N_y))$ 의 복잡도를 가지는 장점이 있다.

- 3차원 구조를 가지는 엣지 비용

기존 4차원 구조에서 3차원으로 줄일 수 있다. 왜냐하면 D* Lite 알고리즘은 엣지 비용을 두 격자 사이의 값으로 고려했지만, 이 논문에서는 기준이 되는 하나의 격자가 그림 3과 같이 그 주변의 인접한 8개의 격자에 대해 가지는 값만 고려하면 충분하기 때문이다.

격자 구조의 특성 상 하나의 격자를 기준으로 그 주변으로는 항상 8개의 격자만 존재할 수 있기 때문에, 각 주변 격자에 1부터 8까지의 index를 매길 수 있고, 엣지 비용을 계산할 때에는 $c(u_x, u_y, d)$ 같이 기준 격자와 해당 방향의 값만으로 표시할 수 있다는 장점이 있다.

특히 기존 연구에서는 최댓값이 $c(N_x, N_y, N_x, N_y)$ 이었음에 비해, 새로운 엣지 비용은 단순한 한 차원의 감소뿐 아니라 $c(N_x, N_y, 8)$ 에서 볼 수 있는 것처럼 세 번째 차원이 8개만 필요하다는 추가적인 장점이 있다.

2.2 우선순위 큐 관리 알고리즘

2.2.1 2차원 구조의 우선순위 큐 관리 함수와 우선순위 큐 관리 알고리즘

위에서 다룬 두 번째 개선 사항을 우선순위 큐 관리 알고리즘과 연결시켜서 좀 더 자세한 설명을 하면 다음과 같다. 기본적인 원리는, $(k1, k2)$ 라는 우선순위 값에 대해, 먼저 $k1$ 이 가질 수 있는 모든 값으로 구분이 된 $2 \times \max(N_x, N_y)$ 개의 행으로 정렬한다. 그리고 $k2$ 를 이용하여 남은 정렬이 이루어진다. 각 열에 포함된 항목끼리는 이중 연결 리스트로 연결되어 있다. 이러한 구조를 이용하면 앞에서 살펴본 이진 최소 힙과는 달리 모든 항목이 전체적으로 정렬이 된다는 큰 장점이 있다.

그렇기 때문에 우선순위 큐에 특정 항목이 존재하는지 탐색을 해야 하는 ‘제거’의 경우, 특정 항목의 $k1$ 을 탐색하는데 걸리는 복잡도는 $O(1)$, 그리고 $k2$ 를 탐색하는데 $O(\max(N_x, N_y))$ 의 복잡도를 가진다. $k1$ 의 각 열에 대해 $k2$ 가 가질 수 있는 최댓값이 $\max(N_x, N_y)$ 로 한정되기 때문이다[14]. 이는 $O(N_x \times N_y)$ 를 가지던 이진 최소 힙에 비하면 획기적인 감소량을 알 수 있다.

다만 삽입과 추출의 경우에도 제거와 마찬가지로 $k1$ 행에서 $O(\max(N_x, N_y))$ 만큼 걸려서 $k2$ 를 찾기 때문에, 이진 최소 힙과 비교할 때 복잡도가 증가한다는 단점이 있다.

그러나 추출과 삽입은 D* Lite 알고리즘의 이진 최소 힙에 비해 복잡도가 다소 증가한다는 단점에도 불구하고, 제거의 경우에 급격한 복잡도의 감소를 확인할 수 있다는 점에서 제안하는 데이터 구조가 개선되었다고 볼 수 있다.

2.2.2 우선순위 큐 관리 알고리즘의 구현

제안하는 2차원 구조의 우선순위 큐 관리 함수를 사용한

우선순위 큐 관리 알고리즘은 다음과 같이 구현할 수 있다. 먼저 새롭게 등장한 용어들을 살펴본다.

Tocc는 Total occupancy, 즉 우선순위 큐에 포함된 전체 항목의 개수를 나타낸다.

Rocc[2*max(Nx,Ny)] [1]는 Row occupancy, 다시 말해 각 행에 포함된 항목의 개수를 뜻한다. k1으로 구분되는 각 행에 포함되는 항목들은 최대 2*max(Nx,Ny)개까지 가능하다.

Rhp[2*max(Nx,Ny)] [1]는 Row highest priority 를 의미하며, 각 행의 최고 우선순위를 가지는 항목을 가리킨다. 이중 연결 리스트 구조의 head와 유사한 개념으로 생각할 수 있다.

Rlp[2*max(Nx,Ny)] [1]는 Row lowest priority 이고, 각 행의 최저 우선순위를 가지는 항목을 나타낸다. 이중 연결 리스트에서의 tail과 같다고 볼 수 있다.

위와 같은 추가적인 저장 변수를 활용함으로써 우선순위 큐 항목의 정렬이 보다 손쉽게 빠르게 진행될 수 있다. 이 변수들을 활용한 우선순위 큐 관리 알고리즘 중 제거, 삽입, 추출이 그림 4-6에 나타나 있다.

■ Remove(x,y)

```
Search Linked List data for (x,y)
If matched
  go to the row of k1
  switch(x,y)
    case head:
      remove connection
      let its next be rhp
    case tail:
      remove connection
      let its prev be rlp
    case body:
      remove connection
  row occupancy--, total occupancy--
```

그림 4. 제거 함수의 pseudo code.

Fig. 4. Pseudo code of remove function.

■ Insert(x,y,k1,k2)

```
If (x,y) is obstacle
  do not insert
Else
  put x,y,k1,k2 value into Linked List data
  switch(k1)
    case head:
      make connection
      let rhp point this
    case tail:
      make connection
      let rlp point this
  (Continued)
  case body:
    search for adequate position
    make connection
    if(identical k1)
      compare k2 and make connection
  row occupancy++, total occupancy++,
  Linked List data index++
```

그림 5. 삽입 함수의 pseudo code.

Fig. 5. Pseudo code of insert function.

■ Extract()=Pop()

```
Get rhp from Linked List data
Let it be (x,y)
Remove(x,y)
```

그림 6. 추출 함수의 pseudo code.

Fig. 6. Pseudo code of extract function.

Procedure calculateKey(s)

Return

$[k1; k2] = [\min(g(s), rhs(s)) + h(s_{start}, s) + k_m; \min(g(s) + rhs(s))]$

Procedure initialize()

$U = \emptyset$

$k_m = 0$

for all $s \in S, rhs(s) = g(s) = \infty$

$rhs(s_{goal}) = 0$

U.Insert($s_{goal}, CalculateKey(s_{goal})$)

initialize h with Euclidean distance.

initialize occ_cell.

Procedure updatevertex(u)

if ($u \neq s_{goal}$), $rhs(u) = \min_{1 \leq d \leq 8} (c(u_x, u_y, d) + g(s'))$

if ($u \in U$), **U.Remove**(u)

if ($g(u) \neq rhs(s)$), **U.Insert**(u, CalculateKey(u))

Procedure computeshortestpath()

while (**U.Topkey**() < calculatekey(s_{start}) or $rhs(s_{start}) \neq g(s_{start})$)

$k_{old} = U.Topkey()$

$u = U.Pop()$

if ($k_{old} < calculatekey(u)$)

U.Insert(u, calculatekey(u))

elseif ($g(u) > rhs(u)$)

$g(u) = rhs(u)$

for all $s \in Succ(u)$, updatevertex(s)

else

$g(u) = \infty$

for all $s \in Succ(u) \cup \{u\}$, updatevertex(s)

Procedure main()

$s_{last} = s_{start}$

initialize()

computeshortestpath()

while ($s_{start} = s_{goal}$)

$s_{start} = \arg \min_{1 \leq d \leq 8} (c(x_{start}, y_{start}, d) + g(s'))$

Move to s_{start} .

recalculate h.

(Continued)

if any new occ_cell

$k_m = k_m + h(s_{last}, s_{start})$

$s_{last} = s_{start}$

for all **new occ_cell** with changed edge

Update the edge $c(u_x, u_y, d)$.

add new occ_cell.

updatevertex(u)

computeshortestpath()

그림 7. Modified D* Lite 알고리즘의 pseudo code.

Fig. 7. Pseudo code of modified D* Lite algorithm.

2.3 Modified D* Lite 알고리즘의 설명

제안한 내용들을 적용한 전체 알고리즘은 그림 7과 같다.

III. 센싱 및 제어부

본 장에서는 이동 로봇의 온라인 경로 계획의 구현에 필수적인, 입력 데이터를 제공하는 LRF 센서 처리 입력부와 경로 추종을 위한 모터 제어 출력부에 대해 기술한다.

1. LRF 센서 처리부

그림 8에 보인 바와 같이 전역 좌표계에서 (x_r, y_r, θ_r) 에 위치한 이동 로봇상의 LRF 센서 (x_{sen}, y_{sen}) 는 로봇 좌표계에서 스캔 데이터를 생성해 내고, 이로부터 판별된 장애물 정보(원으로 표시됨)를 전역 좌표계로 변환한 후, 이를 포함한 장애물 그리드(사각형으로 표시됨)를 생성해낸다. 그리고 로봇이 유한한 크기를 가지므로, 이를 한 그리드 이내로 s만큼 축소함과 동시에 장애물 그리드를 동일한 양 s만큼 확장시킬 필요가 있다. 한 그리드의 크기가 s의 크기와 동일하도록 설정한다면, (x_u, y_u) 가 장애물일 경우 $Succ(x_u, y_u)$ 도 확장된 장애물로 정해야 한다.

그림 9의 왼쪽 부분과 같은 환경에서 로봇이 오른쪽 방향으로 하나의 격자의 폭을 가지는 경로를 통과할 수 있는 것처럼 보이지만, 실제로는 로봇이 지나가기에 협소한 경로가 된다. 오른쪽 부분에서와 같이 로봇의 크기가 한 그리드 이

그림 8. 이동 로봇 좌표계와 전역 좌표계.

Fig. 8. Mobile robot coordinate system and global coordinate system.

그림 9. 장애물 확장.

Fig. 9. Obstacle expansion.

그림 10. 가능한 이동 로봇의 궤적.

Fig. 10. Set of probable trajectory of mobile robot.

내 크기로 축소하는 대신 장애물을 한 격자씩 확장시킴으로써, 이러한 경우를 배제할 수 있다.

장애물 격자의 확장까지 완료되었으면, 이 정보를 이전 장애물 격자 정보와 비교하여 새로운 장애물이 나타났는지 확인해야 한다. 이러한 새로운 장애물 정보 여부 판단은 간단한 비교를 통해 수행할 수 있다. 이전 주기의 장애물 격자 정보를 기억하고 있는 상태에서, 현재 주기에 감지된 장애물 격자 정보를 이것과 비교하여 차이가 있다면 바로 그 차이 나는 부분이 새로운 장애물 격자 정보로 인식되는 것이다.

2. 모터 제어부

다음으로는 모터 제어 출력부를 기술한다.

모터의 속도 프로파일은 선속도 및 각속도 모두 사다리꼴(trapezoidal) 프로파일을 사용한다. 경로계획부에서 이동할 다음 그리드가 결정되면, 그림 10과 같이 회전 후 직진을 수행하는 선속도 및 각속도 프로파일을 생성하고, 모터 제어부는 매 제어 주기마다 모터의 속도제어를 수행한다.

IV. 시뮬레이션 결과

본 장에서는 컴퓨터 시뮬레이션을 통해 이동 로봇을 위한 온라인 경로 계획 알고리즘의 동작을 검증해 보았다.

1. 시뮬레이션 환경

시뮬레이션은 그림 11과 같이 20x20 크기의 지형에서 진행

그림 11. Environment 설명을 위한 지형.

Fig. 11. Example terrain for explanation of 'Environment'.

된다. 시작점은 (4,4) 위치이고 목표점은 (14,14) 위치로 정했다. 초기 장애물 격자도 임의로 주어진다.

유클리드 격자 거리 기반으로 계산을 하면 가로, 세로 방향으로 한 칸 이동할 때와 대각선 방향으로 이동할 때 1:1.4의 비율을 가져야 한다. 그런데 소수점을 다루기 시작하면 시뮬레이션이 필요 이상으로 복잡해지기 때문에, 여기에서는 10:14로 계산을 하였다. 이렇게 되면 엣지 비용 역시 10배로 고려해야 한다는 점을 잊으면 안 된다.

II 장에서 설명한 Modified D* Lite 알고리즘이 경로 계획을 하게 되고, 센서와 모터를 각각 모델링하여 실제 이동 로봇이 센서로 주변 환경을 감지한 후 모터가 동작하여 일정 거리 이동 및 각도 회전하는 과정을 반복하며 시작점에서 목표점까지 이동한다.

2. 전체 알고리즘

장애물 위치 정보를 포함하는 Environment라는 부분에 대해 먼저 설명을 하면 다음과 같다. Environment는 시간에 따른 장애물에 대한 정보를 표 1과 같이 나타내었으며, 전체 시뮬레이션을 순차적으로 진행시키면서 각 시점에서 필요로 하는 장애물 정보를 제공하는 역할을 한다.

n번째 environment가 호출될 때마다 해당 n을 가지는 각 행의 나머지 4개 정보를 센서부에 제공하게 된다. 동일한 n을

표 1. 각 시점에 주어지는 장애물 정보.

Table 1. Obstacle information given at each step.

n	Obstacle index O_i_n (Max=N)	Status (1=enter, 0=exist, -1=exit)	dx	dy
0	2	1	0	0
0	3	1	0	0
0	4	1	0	0
4	1	1	0	0
10	5	1	0	0
11	5	0	0	1
12	5	-1	0	0

그림 12. 전체 시뮬레이션 블록 다이어그램.

Fig. 12. Total simulation block diagram.

가지는 행이 하나 이상이면 모두 해당된다. 장애물 인덱스 O_i_n 는 장애물의 번호로서 최대 N개의 장애물이 존재할 수 있다. Status는 세 가지 값을 가질 수 있는데, 1이면 주변 환경에 나타났다는 것이고, 0이면 이전 단계에서 나타나서 현재 단계에도 그대로 존재한다는 것이고, -1이면 사라진다는 것이다. 그리고 dx, dy는 장애물이 x축 방향, y축 방향 각각으로 움직인 변화량을 나타낸다. 이상과 같은 environment의 내용을 포함하여, 전체 시뮬레이션의 블록 다이어그램을 그리면 그림 12와 같다.

3. Modified D* Lite 알고리즘 시뮬레이션 결과

위와 같은 환경에서 MATLAB으로 일단 온라인 경로 계획 알고리즘인 Modified D* Lite 알고리즘 시뮬레이션을 수행한 결과는 그림 13과 같다. 이것은 보완 사항을 적용시킨 경로 계획 알고리즘이 정상적으로 동작하는지 확인할 수 있다는 점에 그 의미가 있다.

그러나 로봇이 최단 경로를 따라가는 도중에 그림 14와 같이 (5,4) 격자가 장애물임을 인식했다면, 로봇은 새로운 경로를 다시 계획해야 한다. 이 경우, 잠시 제 자리에 멈추고 새로 동적 계획을 수행한다.

그림 15는 새로운 경로를 계획한 후, 경로를 따라가는 모습이다. 목표점에 도달할 때까지 추가적인 장애물이 발견되지 않았다.

그림 13. 새로운 장애물이 나타나기 전 최단 경로 탐색.

Fig. 13. Shortest path before new obstacle appears.

그림 14. (5,4)격자가 새로운 장애물이라는 것을 발견.

Fig. 14. Found (5,4) grid is obstacle.

그림 15. 새로운 경로를 계획한 후 목표점에 도달.

Fig. 15. Reached goal grid after planning new path.

4. 전체 시뮬레이션 결과

다음으로는 온라인 경로 계획 알고리즘을 포함하여, 센서부와 모터 제어부까지 포함한 전체 시뮬레이션 결과를 확인함으로써, 그림 16과 같이 실제 장애물을 감지하고 이동 로봇이 물리적으로 움직이는 것을 알아볼 수 있다.

그림 16. 모터 제어를 포함한 시뮬레이션.

Fig. 16. Simulation including motor control.

그림 17. (10,10), 45도 방향에서의 로봇 주변 장애물 감지.

Fig. 17. Sensing obstacles at (10,10) in direction of 45.

그림 18. (12,11), (12,12) 장애물 인식하고 새 경로 계획.

Fig. 18. Planning new path after sensing (12,11), (12,12) grid are obstacles.

센서는 HOKUYO URG-04LX 모델의 사양을 사용했다. 최대 감지 범위는 4m, 감지 각도 범위는 240°, 측정 각도 단위는 0.36° 이다. 그림 17과 같이 거리 상의 범위와 각도 상의 범위에 적합하게 센서 동작을 모사했음을 확인할 수 있다. 그림 17은 x축 방향을 기준으로 45° 반 시계 방향으로 회전한 로봇이 (10,10)에 위치하고 있는 상황을 나타낸다.

그림 18과 같이 장애물로 (12,11), (12,12) 격자가 나타나면, 이를 회피하여 새로운 경로를 계획하게 된다.

5. 기존 알고리즘과의 비교

기존의 D* Lite 에 대한 Modified D* Lite 알고리즘의 효율성을 메모리 소요량, 알고리즘 계산 속도, 생성된 경로의 최적성 측면에서 비교하였다.

먼저 메모리 소요량의 비교를 수행하였다. 기존의 D* Lite 에서 가장 메모리 소요가 큰 부분이 s에서 s'까지 엣지 비용 $c(s, s')$ 로써, NxN 그리드 환경에 대하여 s, s' 모두 2차원의 양이므로 N^4 의 메모리가 소요된다. 반면 제안된 Modified D* Lite 알고리즘의 경우, $8N^2$ 의 메모리가 소요된다. 구체적으로, 제안된 Modified D* Lite 알고리즘에서는 $N=20$ 인 경우, 1/50, $N=1000$ 인 경우 1/1250의 메모리만이 필요하다.

다음으로 알고리즘의 계산속도를 비교하였다. 두 알고리즘에서 모두 가장 빈번하게 사용되는 부분이 우선순위 큐인데, 기존의 D* Lite에서는 이진 최소 힙을 사용하며, 경로의 격자 점들에 대하여 같은 빈도로 큐에의 삽입, 추출, 제거가 이루어지며, 삽입, 추출에 각각 $\log N^2$, 제거에 N^2 의 계산량이 필요하다. 반면, 제안된 Modified D* Lite 알고리즘의 경우 2차원 구조의 우선순위 큐를 사용하므로, 삽입, 추출, 제거에 각각 N의 계산량이 필요하다. 구체적으로, 제안된 Modified D* Lite 알고리즘에서는 $N=20$ 인 경우 약 14%, $N=1000$ 인 경우 약 0.3%의 큐 관리를 위한 계산량만이 소요된다.

끝으로 생성된 경로의 최적성을 비교하였다. 휴리스틱 함수로써 기존의 D* Lite에서는 체비셰프 거리를 사용하고, 제안된 Modified D* Lite 알고리즘의 경우 유클리드 거리를 사용하였다. 체비셰프 거리에서는 수평, 수직, 대각선의 격자이동이 모두 1의 비용을 가지는 반면, 유클리드 거리에서는 수

평, 수직 이동은 1, 대각선 이동은 $\sqrt{2}$ 의 비용을 가지므로, 훨씬 정확한 이동 비용을 사용하여 경로를 생성해내므로, 최단 거리의 경로를 생성해 낼 수 있다. 구체적으로 (1,1)에서 (4,2)로 이동하는 경우 기존의 D* Lite에서는 (2,1), (3,1)을 경유하는 경로, (2,2), (3,1)을 경유하는 경로 둘 모두를 이동 비용 3으로서 최적이라고 판단할 수 있지만, 제안된 Modified D* Lite 알고리즘의 경우 (2,1), (3,1)을 경유하는 경로는 이동 비용이 $2+\sqrt{2}$, (2,2), (3,1)을 경유하는 경로는 이동 비용이 $3\sqrt{2}$ 로 판단하므로, (2,1), (3,1)을 경유하는 경로만을 이동 비용이 $2+\sqrt{2}$ 인 최적의 경로라고 판단하므로, 최단 거리의 경로를 구할 수 있는 장점이 있다.

종합하면, 기존의 D* Lite에 비하여 제안된 Modified D* Lite 알고리즘은, 적은 메모리 소요량, 향상된 계산 속도로, 최적의 경로를 생성해 낼 수 있는 장점이 있다.

V. 결론 및 추후 과제

본 논문에서는 D* Lite 알고리즘에 대해 깊이 있는 분석을 했으며, 분석 결과를 이용하여 보다 개선된 Modified D* Lite 알고리즘을 제안하였고, 시뮬레이션으로 그 성능을 검증했다.

D* Lite 알고리즘이 가지고 있는 세 가지 개선될 여지가 있는 항목 - 체비셰프 거리 기반의 휴리스틱 함수, 이진 최소 힙을 이용한 우선순위 큐 관리, 4차원 구조의 엣지 비용 -에 대해 새로운 해법을 제시했다. 또한 LRF 센서 처리와 모터의 동작을 모델링하여 전체 시뮬레이션 결과가 이를 모두 보여줄 수 있게 하였고, 제안한 알고리즘의 성능을 시뮬레이션을 통해 검증하였다. 기존의 D* Lite에 비하여 제안된 Modified D* Lite 알고리즘은, 적은 메모리 소요량, 향상된 계산 속도로, 최적의 경로를 생성해 낼 수 있는 장점이 있음을 보였다.

추후 과제로는 다음과 같은 사항이 있다. 본 논문에서는 전체 지형을 격자로 나누어 각 격자를 로봇 수행의 최소 단위로 고려했다. 이에 따라 한 격자에서 다른 격자로 이동한 후에 센서 정보를 받아서 활용했는데, 이것은 센서의 스캔 주기보다 비교적 긴 시간이 걸린다. 격자를 좀 더 세밀하게 나누어 한 격자를 이동하는데 걸리는 시간이 센서의 스캔 주기와 비슷해진다면, 센서의 활용도를 높이는 것은 물론 보다 정확한 경로 계획과 장애물 회피 성능을 얻을 수 있을 것이다. 그러나 이 경우 계산량이 증가하므로, 성능 대비 계산량에 대한 연구가 추가적으로 필요하다.

참고문헌

- [1] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "Cooperative sweeping by multiple mobile robots," *Proc. IEEE International Conference on Robotics and Automation*, Minneapolis, MN, pp. 1744-1749, Apr. 1996.
- [2] M. Gerke, "Genetic path planning for mobile robots," *Proc. American Control Conference*, San Diego, CA, pp. 2424-2429, Jun. 1999.
- [3] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on System. Man and Cybernetics*, vol. 34, no. 1, pp. 718-725, Feb. 2004.
- [4] Y. Koren and J. Borenstein, "Potential field methods and their

inherent limitations for mobile robot navigation," *Proc. IEEE Conference on Robotics and Automation*, Sacramento, CA, pp. 1398-1404, Apr. 1991.

- [5] J. Borenstein and Y. Koren, "The Vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278-288, 1991.
- [6] D. An and H. Wang, "VPH: a new laser radar based obstacle avoidance method for intelligent mobile robots," *Proc. 5th World Congress on Intelligent Control and Automation*, Hangzhou, China, pp. 4681-4685, Jun. 2004.
- [7] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, pp. 2383-2388, Oct. 2002.
- [8] J.S. Zelek, "Complete real-time path planning during sensor-based discovery," *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, pp. 1399-1404, Oct. 1998.
- [9] L. Podsedkowski, J. Nowakowski, M. Idzikowski, and I. Vizvary, "A new solution for path planning in partially known or unknown environment for nonholonomic mobile robots," *Robotics and Autonomous Systems*, vol. 34, pp. 145-152, 2001.
- [10] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proc. International Joint Conference Artificial Intelligence*, pp. 1652-1659, 2005.
- [11] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354-363, Jun. 2005.
- [12] K. Loudon, *Mastering Algorithms with C*, O'Reilly, 2000.
- [13] H. M. Choset et al., *Principles of Robot Motion*, MIT press Cambridge, 2005.
- [14] Y. H. Kim, "Design of a robot task planning system with a state partitioning technique," M.S. thesis, Dept. of Electrical Engineering, KAIST, 2007.
- [15] Scanning Laser Range Finder URG-04LX Specifications, HOKUYO, 2005.

강 태 호

1984년 5월 1일생. 2009년 KAIST 전기 및 전자공학과 졸업. 2011년 동 대학원 석사. 관심 분야는 이동 로봇, 온라인 경로 계획.

김 병 국

1952년 10월 5일생. 1975년 서울대학교 전자공학과 졸업. 1975년 한국과학기술원 전기 및 전자 공학과 석사. 1981년 동 대학원 박사. 1981년~1986년 우진계기(주) 연구실장. 1982년~1984년 University of Michigan 방문연구. 1986년~현재 한국과학기술원 전기 및 전자공학과 교수. 관심분야는 실시간 시스템, 로보틱스, 임베디드 제어.