Journal of Institute of Control, Robotics and Systems (2016) 22(3):217-225
http://dx.doi.org/10.5302/J.ICROS.2016.15.0201 ISSN:1976-5622 eISSN:2233-4335

RRT 와 SPP 경로 평활화를 이용한 자동주행 로봇의 경로 계획 및 장애물 회피 알고리즘

Path Planning and Obstacle Avoidance Algorithm of an Autonomous Traveling Robot Using the RRT and the SPP Path Smoothing

박 영 상, 이 영 삼*
(Yeong-Sang Park[1] and Young-Sam Lee[1,*])
[1]Department of Electrical Engineering, Inha University

Abstract: In this paper, we propose an improved path planning method and obstacle avoidance algorithm for two-wheel mobile robots, which can be effectively applied in an environment where obstacles can be represented by circles. Firstly, we briefly introduce the rapidly exploring random tree (RRT) and single polar polynomial (SPP) algorithm. Secondly, we present additional two methods for applying our proposed method. Thirdly, we propose a global path planning, smoothing and obstacle avoidance method that combines the RRT and SPP algorithms. Finally, we present a simulation using our proposed method and check the feasibility. This shows that proposed method is better than existing methods in terms of the optimality of the trajectory and the satisfaction of the kinematic constraints.

Keywords: path planning, autonomous traveling robot, RRT, SPP curve, path smoothing

I. 서론

최근 드론 및 무인 자동차와 같은 자동 로봇(autonomous robot)이 중요한 이슈로 떠오르고 있다. 자동 로봇이란 높은 자율성을 가진 로봇을 뜻하는 말로써, 예측할 수 없는 상황 즉 동적 환경에서도 자가적인 결정을 내리고 행동하는 로봇을 말한다. 이러한 자동 로봇에서는 경로 계획(path planning)이 매우 중요한 부분을 차지하는데, 이는 경로 혹은 움직임에 따라 로봇의 자율성이 얼마나 높은지 나타낼 수 있는 일종의 척도가 되기 때문이다.

경로 계획 알고리즘에서 가장 중점을 두는 부분은 최단 거리를 가지는 경로를 생성하는 것이다. 기존의 경로 계획 알고리즘에는 그래프(graph) 또는 트리(tree) 자료구조를 이용하여 최단 거리 문제를 해결하는 Dijkstra [1], A* [2,3], 그리고 D* [4,5] 알고리즘 등이 있다. 이 알고리즘들은 전체 지도를 모두 정점(node)으로 세분화 한 후, 해당 알고리즘에 따라 최단 거리를 찾아나간다. 그래프 또는 트리를 이용한 경로 계획 알고리즘들은 실제 최적의 결과를 구하게 되어 있다는 장점이 있으나, 복잡한 환경에서는 전체 정점을 모두 탐색 할 수도 있게 되어 계산량이 많아진다는 비효율성을 가지고 있다.

이와 같은 문제점을 해결하기 위해 몇 가지의 무작위 알고리즘(randomized algorithm)이 연구 되었다. 무작위 알고리즘은 무작위 샘플링(random sampling)을 기반으로 하는 알고리즘으로, 전체 지도에서 무작위로 정점을 추출하여 각각의 알고리즘을 적용하는 방법이다. 이 방법은 그래프 또는 트리 자료구조를 이용한 방법에 비해 최적의 경로를 찾아낼 수는 없으나, 계산량이 적고, 빠른 시간 안에 도착점까지의 경로를 생성해 낼 수 있다는 장점이 있다. 이 방법을 이용하는 알고리즘은 대표적으로 randomized potential field, probabilistic roadmap [6], RRT [7,8] 알고리즘 등이 있다. 그 중에서도 RRT는 Voronoi region에서 착안한 빠른 공간 탐색 방법을 이용하여 최근 각광받고 있는 이론이다.

자동 로봇에서는 경로의 평활화(smoothing)도 경로 계획 못지 않은 중요성을 가지고 있다. RRT를 이용한 경로 계획을 실행하면 흔들림(jittering)을 가진 경로가 생성되는데, 이를 최적화시키기 위한 가장 단순한 평활화 기법으로는 경로에서 장애물이 없는 두 지점을 직선으로 연결하는 방법이 있다. 하지만 이렇게 평활화 된 경로는 추종하는 로봇의 실제 동역학을 만족시키지 못한다. 이를 해결하는 몇 가지의 평활화 기법으로는 SPP 곡선[9], clothoid 곡선[10], B-spline 곡선[11] 등을 이용하는 방법들이 있다.

상기한 경로 생성과 경로 평활화는 기존 연구에서는 주로 분리하여 연구가 진행되었지만, 실제 주행 로봇에 적용할 때는 분리해서 생각 할 수 없는 관계이다. 이를 개선하기 위해 최단 거리 경로와 로봇의 동역학을 모두 만족시키기 위한 몇 가지 기존 연구 결과도 있었으나, 즉각적인 후처리를 필요로 하여 예측 불가능한 계산 부하가 발생하거나[12], 최단 거리 경로를 충분히 만족시키지 않는 결과를 도출하였다[13].

이 논문에서는 RRT를 통하여 전역적 경로 계획을 선행하고, SPP를 통하여 계획된 경로의 평활화(smoothing) 및 이륜 로봇의 기구학적 제약조건이 고려된 경로 생성 방법을 제안한다. Section 2에서는 RRT 및 SPP에 대한 이론을 소개하고, Section 3에서는 제안된 방법을 사용하기 위해서 추가적으로

* Corresponding Author
Manuscript received December 1, 2015 / revised January 11, 2016 / accepted January 19, 2016
박영상, 이영삼: 인하대학교 전기공학과
(pys0728k@hotmail.co.kr/lys@inha.ac.kr)
※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 ICT융합 고급인력과정지원사업의 연구결과로 수행되었으며(IITP-2015-H8601-15-1003) 또한 한국전력공사의 재원으로 기초전력연구원의 2015년 선정 기초연구개발과제의 지원을 받아 수행된 것임(과제번호 : R15XA03-12).

사용해야 하는 두 가지 방법을 제안한다. Section 4에서는 RRT와 SPP를 통합한 경로 생성 및 장애물 회피 알고리즘을 제안한다. Section 5에서는 시뮬레이션 결과를 살펴보고 마지막으로 Section 6에서는 결론을 내린다.

II. 관련 이론

1. Rapidly Exploring Random Tree (RRT)

RRT는 트리(tree) 자료구조를 이용한 경로 계획의 일종으로 출발점을 기준으로 지도의 무작위 정점을 샘플링하여 도착점까지의 정점 및 간선(edge)을 구하는 방법이다[7,8].

알고리즘 1. RRT 생성 알고리즘.
Algorithm 1. The RRT generation algorithm.

$GENERATE\_RRT(x_0, K, \Delta t)$
1 $T.\text{init}(x_0)$;
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} = RANDOM\_STATE()$;
4 $x_{near} = NEAREST\_NEIGHBOR(x_{rand}, T)$;
5 $u = SELECT\_INPUT(x_{rand}, x_{near})$;
6 $x_{new} = NEW\_STATE(x_{near}, u, \Delta t)$;
7 $T.\text{add\_node}(x_{new})$;
8 $T.\text{add\_edge}(x_{near}, x_{new}, u)$;
9 Return T

알고리즘 1은 RRT를 생성하는 알고리즘을 의사코드(pseudocode)로 나타낸 것이다. 먼저 지도에서 무작위 정점 x_{rand}를 추출한다. 기존 트리 T에 있는 정점 중에서 x_{rand}와 가장 가까운 정점 x_{near}을 찾고, x_{near}에서 x_{rand}에 도달하기 위한 입력 u를 입력 셋(set) U에서 찾는다. x_{near}, u를 이용하여 Δt만큼 시간이 지났을 때의 실제 상태 x_{new}를 구한다. 이 x_{new}는 x_{near}의 자식 정점(children node)으로 저장되고 간선, 정점의 정보가 트리 T에 저장된다.

RRT는 여러 장점들이 있으나, 가장 중요한 장점은 탐색하지 않은 위치를 더욱 효과적이고 빠르게 탐색해 나간다는 것이다. 이는 RRT가 Voronoi bias를 가지고 있다는 성질이 기인하는 장점이다. 알고리즘 1의 의사코드에 따라서 알고리즘이 진행될 때, 트리 T 내부의 한 정점이 x_{near}로 선택 될 확률은 그 정점의 Voronoi region의 크기와 비례한다. Voronoi region의 크기가 크다는 것은 빈 공간이 크다는 것을 의미하므로 결국 빈 공간을 탐색 할 확률이 커지게 되고, 이것이 바로 Voronoi bias를 의미한다.

2. Single Polar Polynomial (SPP) 곡선

SPP 곡선은 경로의 평활화 기법으로 사용되는 이론으로써 곡률이 연속적인 경로를 생성하여 로봇의 기구학적 제약조건을 만족하는 경로를 생성 하는 기법이다[9].

경로 계획을 통해 생성된 경로들은 일반적으로 직선만으로 이루어져 있거나, 직선과 원호를 결합한 형태로 이루어져 있는 경우가 많다. 그러나 이러한 경로를 로봇이 완벽하게 추종할 수 있는 경우는 많지 않다. 실제 로봇에는 기구학적 제약조건이 있으나, 경로 계획을 통해 생성된 경로가 이를 고려하지 않았기 때문이다.

제약조건을 만족하는 곡선을 얻기 위해서, 곡선 상의 거리에 대해 독립변수들이 균등하게 변할 수 있는 극 좌표계에서

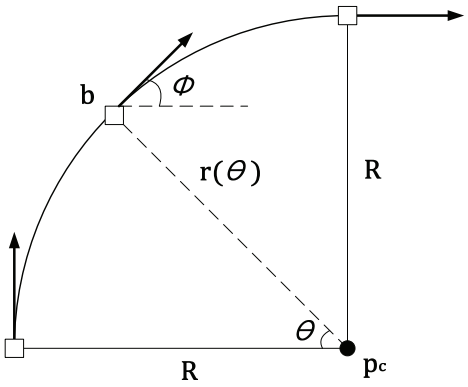


그림 1. 주행 로봇의 회전각 θ 에 따른 SPP 곡선의 반경 변화.
Fig. 1. The radius change of the SPP curves for different rotation angle θ.

SPP를 이용하여 곡선을 표현할 수 있다. 그림 1은 임의의 SPP 곡선 b를 나타낸 그림이다. SPP 곡선이 반지름이 R이고 p_c를 원의 중심으로 하는 원호를 대체한다고 할 때, 로봇의 회전각 θ에 따라 변화하는 반지름을 나타내는 변수 $r(\theta)$와 θ에 대한 1차 도함수 $\dot{r}(\theta)$, 2차 도함수 $\ddot{r}(\theta)$는 (1)과 같고, (2)와 같은 경계조건을 만족해야 한다.

$$\begin{aligned} r(\theta) &= a_0 + a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4 + a_5\theta^5 + \cdots \\ \dot{r}(\theta) &= \frac{dr}{d\theta} = a_1 + 2a_2\theta + 3a_3\theta^2 + 4a_4\theta^3 + 5a_5\theta^4 + \cdots \\ \ddot{r}(\theta) &= \frac{dr^2}{d^2\theta} = 2a_2 + 6a_3\theta + 12a_4\theta^2 + 20a_5\theta^3 + \cdots \end{aligned} \quad (1)$$

$$\begin{cases} r(\theta) = R,\ \dot{r}(\theta) = 0,\ k = 0, \text{ at } \theta = 0 \\ r(\theta) = R,\ \dot{r}(\theta) = 0,\ k = 0, \text{ at } \theta = \mu \end{cases} \quad (2)$$

여기서 φ는 헤딩각(heading angle) 이고, s는 원주거리일 때, 곡률 k는 다음과 같다.

$$k(\theta) = \frac{d\varphi}{ds} = \frac{r^2(\theta) + 2\dot{r}^2(\theta) - r(\theta)\ddot{r}(\theta)}{\left(r^2(\theta) + \dot{r}^2(\theta)\right)^{\frac{3}{2}}} \quad (3)$$

(3)에 (1), (2)를 대입하여 계수 $a_0 \sim a_5$를 구하면 다음과 같다.

$$a_0 = R,\ a_1 = 0,\ a_2 = \frac{R}{2},\ a_3 = -\frac{R}{\mu},\ a_4 = \frac{R}{2\mu^2},\ a_5 = 0 \quad (4)$$

그러므로 다음과 같은 식을 얻을 수 있고, 이는 SPP 곡선의 궤적을 나타낸다.

$$r(\theta) = R\left(1 + \frac{\theta^2}{2} + \frac{\theta^3}{\mu} + \frac{\theta^4}{2\mu^2}\right) \quad (5)$$

상기 SPP 곡선을 사용하더라도 출발점 및 도착점의 헤딩각 φ에 따라서 1-segment, 즉 하나의 곡선 혹은 직선만으로는 경로를 생성하지 못할 수도 있다. 이 때는 2-segments를 사용하여 경로를 생성해야 하는데, 이를 판별하는 기준이 출발점과 도착점이 대칭적인지 혹은 비대칭적인지 확인 하는 것이다. 대칭성을 판별하기 위해 β를 다음과 같이 정의한다.

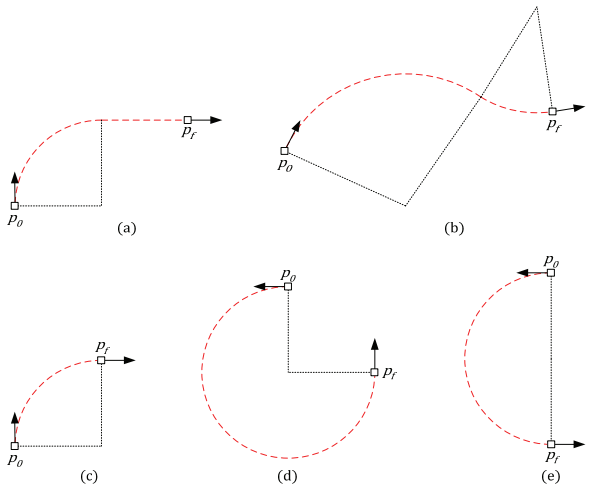


그림 2. 비대칭적인 두 점을 연결하는 경로 (a), (b)와 대칭적인 두 점을 연결하는 경로 (c)~(e).

Fig. 2. Path (a) and (b) which are connecting asymmetrical two points and (c) ~ (e) which are connecting symmetrical two points.

$$\beta = \tan^{-1}\left(\frac{y_f - y_0}{x_f - x_0}\right) \tag{6}$$

출발점 $p_0 = (x_0, y_0, \varphi_0)$ 와 도착점 $p_f = (x_f, y_f, \varphi_f)$ 가 다음을 만족하면 두 점이 대칭적이라고 한다.

$$\varphi_0 - \beta = -(\varphi_f - \beta) \tag{7}$$

그림 2와 같이 대칭적인 두 점을 연결하는 경로는 1-segment로 생성하고 비대칭적인 두 점을 연결하는 경로는 2-segments를 사용하여 생성하면 된다.

1-segment를 이용한 경로는 직선 또는 SPP 곡선 1개로 이루어져 있으며, 2-segments를 이용한 경로는 직선 + SPP 곡선, SPP 곡선 + 직선 또는 SPP 곡선 2개로 이루어져 있다. 그림 2의 경로에서 곡선부는 SPP 곡선이 아닌 원호로 표현되어 있으나 실제 SPP 곡선을 나타내며, 각 경로가 생성될 조건은 논문 [10]을 참조하였다.

III. 추가적인 방법

제안된 방법을 사용하기 위하여 기존 사용되었던 RRT 및 경로 평활화 방법들을 일부 수정하여 적용하여야 한다. 이를 위하여 다음의 2가지 수정된 방법들을 제시한다.

1. 도착점 편향 RRT

RRT는 Voronoi bias에 따라서 빈 공간을 빠르게 탐색 할 수 있고, 장애물이 있는 공간이나 막혀있는 공간에서도 샘플링의 무작위성으로 인하여 회피 및 탈출 할 수 있다. 그러나 기본적으로 비용함수에 의한 bias가 생성되는 그래프 자료구조를 이용한 경로 생성 알고리즘에 비해 경로의 수렴 가능성이 낮다는 단점이 있다. 이 단점을 상쇄시키기 위하여 일정 확률로 도착점에 편향된 bias를 인가한다. 이에 관한 연구들도 진행되고 있으나[14], 제안된 방법에 적용하기 위해서는 추가적인 알고리즘이 필요하다. 도착점 편향에 대한 알고리즘을 다음과 같이 제시한다.

알고리즘 2. 도착점 편향 RRT를 이용한 경로 생성 알고리즘.

Algorithms 2. The path generation algorithm using the goal biased RRT.

```
GENERATE_MODIFIED_RRT(x_0, x_f, p, ε, Δt)
1   T.init(x_0);
2   for k = 1 to ∞ do
3       n = NORMALIZED_RANDOM_NUMBER();
4       if n < p
5           x_rand = x_f;
6       else
7           x_rand = RANDOM_STATE();
8       x_near = NEAREST_NEIGHBOR(x_rand, T);
9       u = SELECT_INPUT(x_rand, x_near);
10      x_new = NEW_STATE(x_near, u, Δt);
11      T.add_node(x_near);
12      T.add_edge(x_near, x_new, u);
13      if ‖x_f − x_k‖ < ε
14          T.path_to_root(x_f, T_result);
15          break;
16  Return T_result
```

알고리즘 2는 도착점으로 편향된 RRT 경로 생성 알고리즘이다. 기존 RRT와의 차이점은 Line 3~7과 13~15이다. 먼저 Line 3~7은 확률 p에 따라 x_{rand}가 x_f로 선택되도록 하는 일종의 bias를 적용한 부분이다. 확률 p는 빠른 수렴성과 장애물 회피간의 trade off 관계에 따라 결정 할 수 있다. 일반적인 RRT는 Voronoi bias에 따라 탐색하지 않은 지역에 대한 방향성이 부여되나, 목표점에 대한 방향성이 부여되지 않은 상태이기 때문에 수렴이 굉장히 늦다. 이러한 문제를 해결하기 위하여 일정 확률 p만큼의 도착점에 대한 방향성을 부여한다. 이 방향성이 너무 작다면 목표점에 대한 방향성이 매우 낮을 수 있다. 반면에 이 방향성이 너무 크다면 지역 최소점(Local minima)에 빠졌을 때 도착점의 방향으로만 무작위 샘플링을 진행하므로 탈출하기 위한 시간이 오래 걸린다. 장애물이 적은 상황에서는 확률 p를 크게 잡아서 빠르게 수렴하도록 선정 할 수 있고, 장애물이 많은 상황에서는 확률 p를 작게 잡아서 지역 최소 점에서 충분히 빠져 나오도록 선정할 수 있다.

Line 13~15는 제안된 방법에 적용하기 위해 추가된 알고리즘이다. 샘플 k에서 로봇의 상태 x_k가 x_f에 도달 했다고 볼 수 있는 반경 ε 근처에 들어가면 RRT 생성을 마치고, 도착점에서부터 부모 노드를 거슬러 올라가서 출발점까지 도달하는 실제 경로를 추출하여 경로에 해당하는 정점 셋 T_{result}를 얻는다. 반경 ε 은 RRT의 특성상 도착점 근처에 도달 하였더라도 정확히 도착점이 아니라면 수렴한 것으로 보지 않고 계속 알고리즘을 진행하기 때문에, 수렴하였다고 볼 수 있는 범위 내에서 계산량을 줄이기 위하여 적용되었다.

그림 3과 4는 RRT에 수렴 반경 ε 을 적용하지 않았을 때와 적용 하였을 때의 시뮬레이션 결과이다. 시뮬레이션은 출발점 (0,0)과 도착점 (400,400)을 입력으로 하여 진행하였고 수렴 반경 ε 은 50으로 설정하였다. 실선으로 이루어진 원은 RRT가 한 스텝당 무작위로 샘플링 한 위치를 나타내며 파선으로 이루어진 원은 수렴 반경을 나타낸다. 무작위 샘플링을

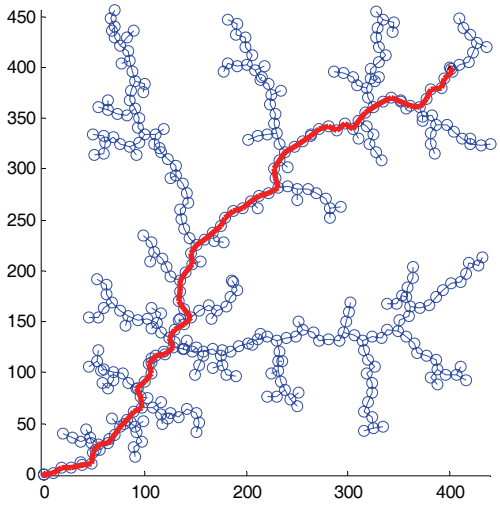


그림 3. 수렴 반경 ε 을 적용하지 않았을 때의 RRT 결과.
Fig. 3. The result of the RRT obtained when the convergence region ε is not used.

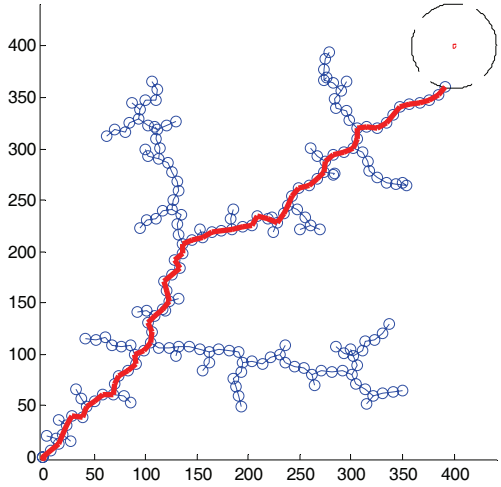


그림 4. 수렴 반경 ε 을 적용하였을 때의 RRT 결과 ($\varepsilon = 50$).
Fig. 4. The result of the RRT obtained when the convergence region ε ($\varepsilon = 50$) is used.

사용하는 알고리즘인 RRT의 특성으로 인하여 수렴 반경이 커질수록 트리의 가지(branch), 즉 샘플링 횟수가 적다는 것을 알 수 있다. 그러나 정확한 도착점 (400,400)에는 미치지 못하므로 두 상관관계를 적절히 고려하여 수렴 반경 ε 을 선정하여야 한다. 수렴 반경을 정하는 부분인 Line 13은 일반적인 유클리드 거리(Euclidean distance)를 기준으로 작성하였으나, 확률적 정보가 존재할 경우 마할라노비스 거리(Mahalanobis distance) [15]를 사용하여 더 적절한 수렴반경을 지정할 수도 있다.

2. SPP 곡선 생성을 위한 전처리 평활화

도착점으로 편향된 RRT로 생성된 출발점부터 도착점까지의 정점 셋 N_{RRT} 는 후처리를 거치지 않으면 흔들림을 가지고 있는 최적이 아닌 경로를 나타내게 된다. 이를 제거하기 위해 먼저 가장 기본적인 평활화 기법을 이용하여 $N_{SMOOTHED}$ 를 생성한 후, SPP 곡선을 위한 새로운 정점 셋 N_{SPP} 를 만든다.

알고리즘 3. SPP 곡선을 생성하기 위한 전처리 평활화.
Algorithm 3. The pre-smoothing for the generation of the SPP curve.

$RRT\_PRE\_SMOOTHING(N_{RRT}, d_{factor}, Obs)$

1 $N_{SMOOTHED}.\text{init}(N_{RRT,1})$;
2 $K = \text{length}(N_{RRT})$;
3 **for** $k = 1$ **to** K **do**
4 $(x_0, y_0) = N_{RRT,k}$;
5 $(x_f, y_f) = N_{RRT,k+1}$;
6 $d_{inter} = \dfrac{\left\| N_{RRT,k+1} - N_{RRT,k} \right\|}{d_{factor}}$;
7 $\varphi = \tan^{-1}\left(\dfrac{y_f - y_0}{x_f - x_0}\right)$;
8 **for** $l = 1$ **to** d_{factor} **do**
9 $x_{inter} = x_0 + l \cdot d_{inter} \cos(\varphi)$;
10 $y_{inter} = y_0 + l \cdot d_{inter} \sin(\varphi)$;
11 s = $COLLISION\_CHECK(Obs, x_{inter}, y_{inter})$;
12 **if** s == TRUE
13 $N_{SMOOTHED}.add\_node(N_{RRT,k+1})$;
14 Return $N_{SMOOTHED}$

알고리즘 3은 가장 간단한 평활화 기법을 나타낸 표이다. N_{RRT}에서 먼저 인접한 정점의 거리를 원하는 만큼 등분 한다. 이후 출발점에서부터 도착점까지 간선의 길이를 등분한 만큼 증가시켜 나가면서 장애물 충돌 검사를 실행한다. 장애물과 충돌한다면 두 인접한 정점 중에서 나중 샘플의 정점을 $N_{SMOOTHED}$에 저장한다. 이 평활화 알고리즘을 이용하면 장애물과 충돌하지 않는 선에서 그 사이의 정점들이 간략화 되어 더 최적인 경로가 생성된다.

IV. 방법

Section 4에서는 상기 RRT와 SPP 곡선을 융합하여 RRT로 계획된 경로를 평활화하고 이륜로봇의 기구학적 제약조건을 만족하면서도 장애물을 회피하는 경로 생성 방법을 제시한다.

1. N_{SPP}의 선정 및 N_{SPP}를 경유하는 SPP 곡선 생성

$N_{SMOOTHED}$ 를 SPP 곡선 생성에 이용하려면 먼저 새로운 정점 셋 N_{SPP} 를 만들어야 한다. 그림 5는 임의의 출발점과 도착점에서 정점의 위치에 따른 SPP 곡선 경로가 어떻게 생성되는지 도시한 그림이다. 타원은 출발점과 도착점을 나타내고 마름모는 SPP 곡선의 중심, 사각형은 $N_{SMOOTHED}$, 원은 N_{SPP}를 나타낸다. 점선은 기존의 $N_{SMOOTHED}$ 를 그대로 사용할 경우의 경로이고, 일점쇄선은 N_{SPP} 를 사용할 경우의 경로이다. 기존의 $N_{SMOOTHED}$ 를 그대로 사용할 경우에는 생성된 경로가 최적 경로의 외부로 이탈하여 전체 경로의 길이가 길어지는 것을 볼 수 있고, 반대로 N_{SPP}를 사용할 경우에는 최적 경로의 내부로 이탈하여 전체 경로의 길이가 짧아지는 것을 볼 수 있다.

그림 6은 N_{SPP} 가 기존 $N_{SMOOTHED}$ 에서 얼마나 떨어져 있느냐에 따라서 SPP 곡선 경로가 어떻게 생성되는지 나타낸 그림이다. 주행 로봇을 구동시키는 환경과 개발자가 중요하게 생각하는 상황에 따라 N_{SPP} 를 생성하는 기준이 달라진다. N_{SPP} 가 $N_{SMOOTHED}$ 에 가까울수록 경로에서 덜 이탈하는 모습을 볼 수 있다. 이는 장애물이 많은 환경에서는 N_{SPP} 를 $N_{SMOOTHED}$ 에 가깝게 정해서 기존 경로를 최대한 추종하게

하면 장애물과의 충돌을 방지 할 수 있다는 뜻이다. 반대로 장애물이 많지 않은 환경에서는 N_{SPP} 를 $N_{SMOOTHED}$ 에서 멀도록 지정하여 경로를 단축하도록 할 수도 있다.

제안된 방법에서 초기 N_{SPP} 는 그림 5와 같이 $N_{SMOOTHED}$ 에 있는 출발 정점과 도착 정점은 그대로 사용하면서, 연속한 두 정점의 중간 지점을 새로운 정점으로 생성하여 사용하였다. 이와 같은 방법은 전체 경로의 길이를 가장 짧게 하고, 곡률이 작은 곡선을 생성하여 로봇의 기구학적 제약조건을 최대한 만족하는 초기 경로를 생성한다.

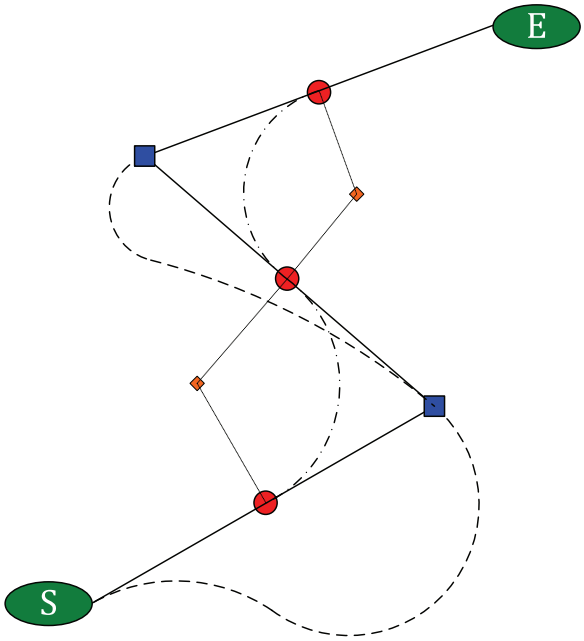


그림 5. $N_{SMOOTHED}$ 와 N_{SPP} 를 사용한 경로의 차이.
Fig. 5. The difference of the path between $N_{SMOOTHED}$ and N_{SPP}.

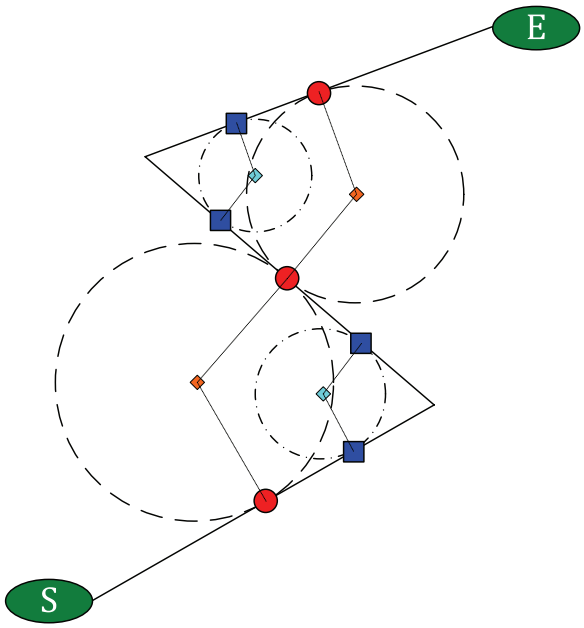


그림 6. N_{SPP} 의 위치에 따라 생성된 경로들의 차이.
Fig. 6. The difference of the path according to the location of the N_{SPP}.

2. 장애물 충돌 판단

Section 4-1의 방법을 사용하면 기구학적 제약조건을 만족하는 경로가 생성되지만 SPP 곡선을 사용한 경로는 장애물 회피 기능을 가지고 있지 않기 때문에 장애물 회피 알고리즘을 사용하여야 한다. 장애물 회피 알고리즘을 사용하기 위해서는 먼저 장애물이 현재 경로를 통과하는 로봇과 충돌 했는지를 판단할 수 있는 기준이 필요하다. 제안된 방법은 먼저 장애물의 반경이 SPP 곡선의 반경보다 더 큰지 확인하고, 장애물이 경로와 충돌할 가능성이 있는 상태인지 확인한 후, 충돌할 가능성이 있는 상태라면 실제 충돌하는 경로인지 확인하는 세 가지 판단을 내린다. 이 세 가지 판단을 내릴 기준을 위하여 세 가지 가정을 세운다.

첫 번째, 장애물의 모양을 원으로 가정하고 실제 장애물보다 더 크게 설정한다. 장애물은 모양에 따라 크게 벽, 물체의 2가지로 나눌 수 있다. 그 중 물체는 일반적으로 유한한 구간의 폐쇄된 도형의 형태를 가지고 있는데, 이로 인해 장애물을 해당 장애물이 내접하는 원으로 가정하는데 큰 문제가 없다. 따라서 제안된 방법은 모든 장애물을 고려하는 것이 아닌 물체만으로 이루어진 장애물들을 고려한다. 이렇게 가정된 원형 장애물은 장애물의 원점으로부터 표면까지 일정한 거리를 가지고 있고, 곡률이 일정하다는 성질로 인하여 SPP 곡선을 사용하여 장애물을 회피할 때, 특성을 고려하기 쉬워진다. 또, 추후 사용할 알고리즘의 특성상 장애물과 경로의 접점이 생기더라도 실제 로봇이 충돌하지 않도록 장애물을 더 크게 설정한다. 두 번째, SPP 곡선을 원으로 가정한다. SPP 곡선은 (2)에서 알 수 있듯이, 중심 p_c 에서부터 시작 정점과 종료 정점까지의 거리가 반경 R 으로 일정하다. 이는 곧 반경이 R 인 원호로 가정 할 수 있다는 뜻이며, Section 4-3의 장애물 회피 알고리즘을 이용하기 위하여 반경이 R인 원으로 가정하였다. 마지막으로, SPP 곡선 1개에 대해서만 장애물 충돌 판단을 하였다. 제안된 방법은 Section 2-2에 의해 모든 경로가 직선과 SPP 곡선으로 이루어져 있으며, 직선부는 RRT에서 생성한 경로를 그대로 추종하면 장애물에 충돌하지 않는다. 결국 SPP 곡선 1개에 대한 장애물 회피 문제로 간략화 할 수 있다. 상기 세 가지의 가정을 통하여 SPP 곡선과 장애물의 관계는 두 원의 관계로 성립하게 된다.

장애물 충돌을 판단하는 조건은 다음과 같다. 먼저, 장애물의 반경과 SPP 곡선의 반경을 비교한다.

그림 7은 장애물의 반경이 SPP 곡선의 반경보다 클 때의 한 가지 상태를 도시한 그림이다. RRT로 생성된 직선 경로 a 와 b는 장애물을 회피 하도록 생성되기 때문에, 장애물과 직선 a, b 사이에 각각 2개의 교점이 생기는 상태는 일어나지 않는다. 그러므로 그림 7의 상태가 가장 최악의 상황으로 생각 할 수 있으나, 반경이 R 보다 큰 직선 a, b의 내접원 중에서 SPP 곡선과 교차하는 내접원은 없으므로 장애물의 반경이 SPP 곡선의 반경보다 크다면 충돌할 가능성은 없다는 것을 알 수 있다.

두 번째로, 장애물과 SPP 곡선의 관계로부터 충돌 가능성을 판별한다. 그림 8은 장애물과 초기 SPP 곡선의 중심 거리에 따른 관계를 나타낸 그림이다. 이점쇄선은 실제 SPP 곡선을 나타내고, 반지름이 R 인 실선으로 이루어진 원은 초기

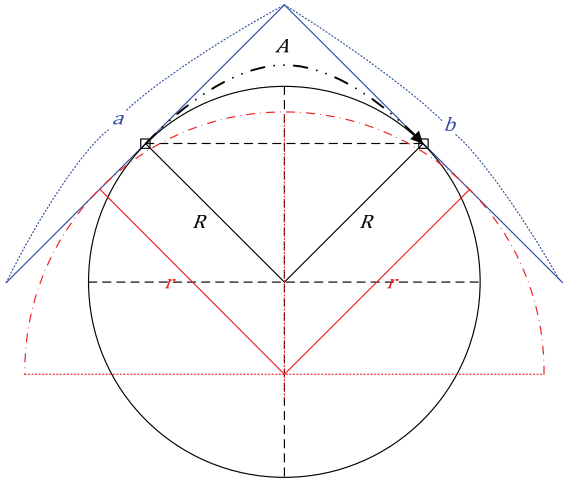


그림 7. 장애물의 반경이 SPP 곡선의 반경보다 큰 경우.

Fig. 7. The case where the radius of an obstacle is bigger than the radius of the SPP curve.

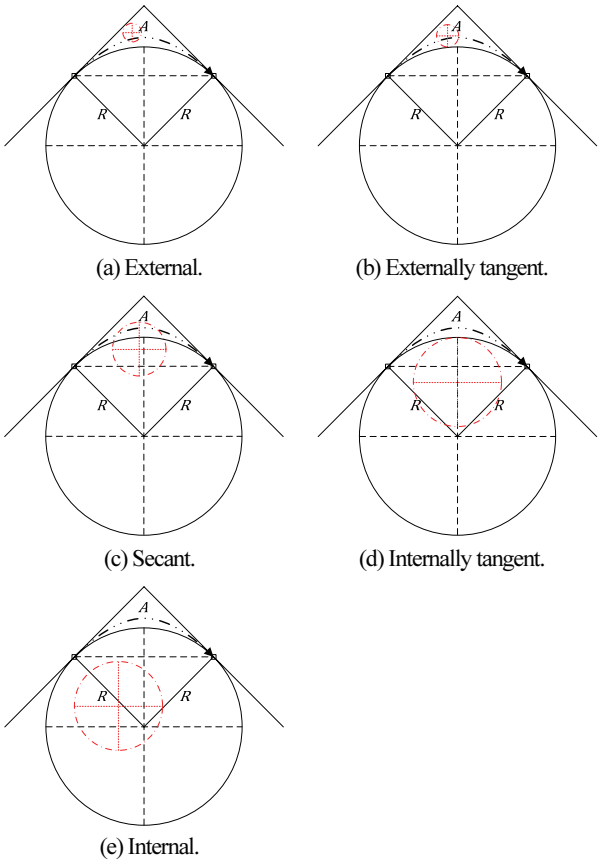
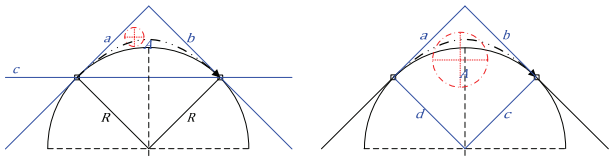


그림 8. 장애물과 초기 SPP 곡선의 관계.

Fig. 8. The relation between an obstacle and the initial SPP curve.

SPP 곡선을 나타내며, 일점쇄선으로 이루어진 원은 장애물을 나타낸다. 그림 8을 통하여 두 원의 관계에서 로봇과 장애물이 충돌할 수 있는 관계는 두 원이 만남일 때로 고려할 수 있다. 가정에 의하면 외부, 외접일 때는 충돌하지 않으나, 실제 SPP 곡선은 가정한 원보다 반지름이 더 크기 때문에 충돌할 가능성이 생기므로 외부, 외접일 때도 충돌하는 것으로 판단해야 한다.



(a) External and externally tangent. (b) Secant.

그림 9. 장애물이 초기 SPP 곡선과 충돌하는 영역.

Fig. 9. The region of the collision between an obstacle and the initial SPP curve.

마지막으로, 상기한 충돌 가능성 판단에서 충돌할 가능성이 있는 외부, 외접 그리고 만남 상태일 경우, 실제로 장애물과 SPP 곡선이 충돌하는지 확인 할 수 있는 조건을 만들어야 한다.

그림 9에서 실제 SPP 곡선과 충돌하는 장애물의 위치는 각각의 영역 A에 해당한다. 그러므로 외부, 외접일 때는 그림 9(a)와 같이 직선 a, b, c로 이루어진 영역 A에 원의 중심이 있다면 충돌하는 것으로 처리하였고, 만남일 때는 그림 9(b)와 같이 직선 a, b, c, d로 이루어진 영역 A에 원의 중심이 있다면 충돌하는 것으로 처리하였다.

3. 장애물 회피 알고리즘

상기 Section 4-2를 통해 장애물이 경로와 충돌하는 상태가 되면 그림 10과 같이 반경이 R'인 SPP 곡선을 새로 만들어야 한다. 이때 가장 최적의 R'는 기존 RRT로 생성된 경로와 내접이면서, 장애물과 내접의 관계를 가지고 있어야 한다. 초기 SPP 곡선을 나타내는 원의 중심을 (x, y)라고 하고, 반경을 R이라고 할 때, 새로운 SPP 곡선을 나타내는 원의 중심 (x', y')과 R'는 다음과 같은 수식을 연립하여 얻을 수 있다.

$$\begin{cases} R' = l\sin\sigma = (\sqrt{(x'-x_s)^2+(y'-y_s)^2})\sin\sigma \\ R'-r = d = \sqrt{(x'-x_{obs})^2+(y'-y_{obs})^2} \\ R' = \left\| (p_s + (l\cos\sigma)\cdot u_f) - p' \right\| \end{cases} \quad (8)$$

p_{obs} 는 장애물의 중심, p_s 는 $N_{SMOOTHED}$ 의 한 정점, 그리고 $\boldsymbol{u_0}$ 와 $\boldsymbol{u_f}$ 는 각각 시작 정점의 단위 방향 벡터와 종료 정점의 단위 방향 벡터를 나타낸다. (8)을 연립하여 생성된 (x', y')와 R'를 사용하여 새로운 시작 정점과 종료 정점을 구할 수 있고, 이를 통해 SPP 곡선을 생성하면 장애물에 내접하는 최적의 SPP 곡선을 생성 할 수 있다.

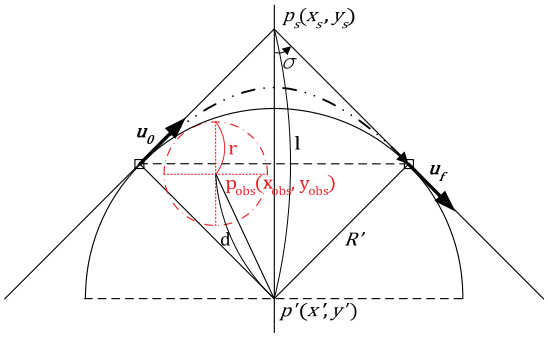


그림 10. 장애물에 충돌하지 않는 최적의 SPP 곡선.

Fig. 10. The optimal SPP curve that does not collide with an obstacle.

V. 시뮬레이션

첫 번째 시뮬레이션은 출발점 (0,0)과 도착점 (400,400)을 입력으로 하여 진행하였다. 장애물은 임의의 네 점 (180,180), (310,230), (150,350), (300,350)을 중심으로 하고 반경 50인 원을 사용하였다. $x_{rand} = x_f$ 일 확률 변수 p는 0.5로 지정하였다. 프로그램을 구동한 PC는 클럭속도가 3.40GHz인 Intel Core i7-4770 프로세서를 기반으로 하고 8GB 메모리를 가지고 있다.

그림 11은 시뮬레이션 결과이다. 원은 장애물을 나타내며, 실선은 제안된 방법으로 생성된 경로를 나타내고, 점선은 기존 RRT에 간단한 직선 평활화만을 적용한 경로를 나타낸다. 직선부에서는 기존 RRT와 큰 차이가 없지만, 곡선부에서 제안된 방법은 부드러운 곡선을 그리며 제약조건을 만족시키는 것을 볼 수 있다.

두 번째 시뮬레이션은 장애물 회피 알고리즘의 성능을 판별하기 위해서, (180,180)에 있는 장애물의 위치를 (188,203)으로 변경하여 장애물과 경로가 충돌하도록 조정하였다. RRT는 기본적으로 절대 장애물에 부딪히지 않는 장애물 회피 기능을 가지고 있기 때문에 직선구간에서의 장애물 회피는 고려할 필요가 없다. 그러므로 곡선구간에서의 장애물 회피, 즉 SPP 곡선이 생성되는 구간에서만 장애물 회피 기능이 적용되어 있으며, 이 구간에서 장애물 회피가 되는 것을 증명한다면 전 구간에서의 장애물 회피가 보장된다. 추가적으로 첫 번째 시뮬레이션에서 구해진 $N_{SMOOTHED}$ 를 이용하여 기존 경로와 같은 경로를 생성하도록 함으로써 RRT가 경로를 변경하지 않게 하였다.

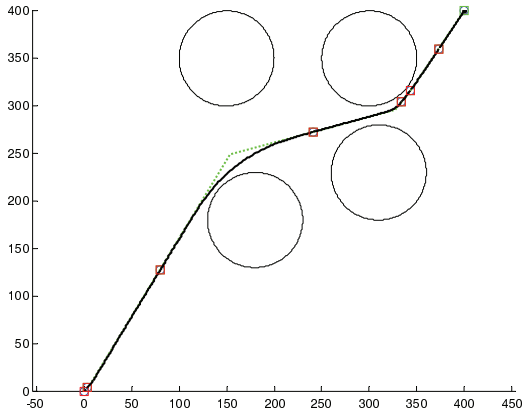


그림 11. 제안된 방법과 기존 RRT 경로의 차이.

Fig. 11. The difference between the proposed method and the original RRT path.

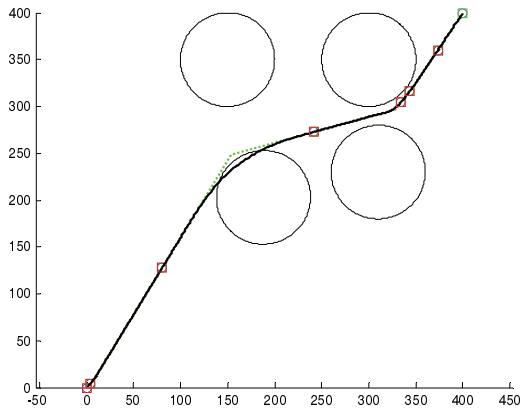


그림 12. 제안된 장애물 회피 알고리즘을 사용하지 않는 경로 생성.

Fig. 12. The path generation not using the proposed obstacle avoidance algorithm.

그림 12는 제안된 장애물 회피 알고리즘을 사용하지 않는 경로 생성 그래프이고, 그림 13과 14는 제안된 장애물 회피 알고리즘을 사용하는 경로 생성 그래프이다. 점선으로 이루어진 원은 새로운 SPP 곡선을 가정한 원을 나타내는데, 장애물에 내접한 최적의 SPP 곡선이 생성되었다는 것을 볼 수 있다.

세 번째 시뮬레이션은 제안된 경로 계획의 성능을 판별하기 위해서 목표 bias 확률 p에 따른 평균 실행시간을 측정하였다.

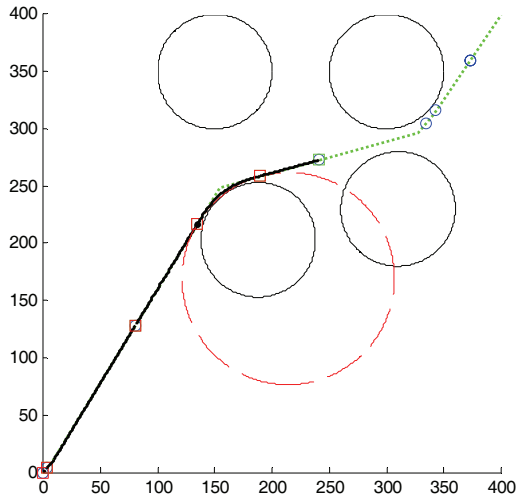


그림 13. 제안된 장애물 회피 알고리즘을 사용하는 경로 생성.

Fig. 13. The path generation using the proposed obstacle avoidance algorithm.

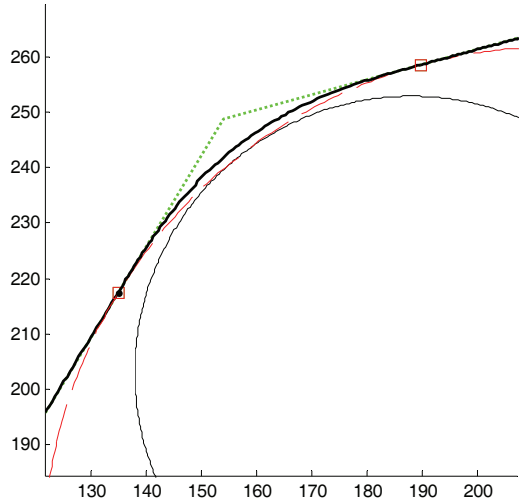


그림 14. 제안된 장애물 회피 알고리즘을 사용하는 경로 생성 (확대).

Fig. 14. The path generation using the proposed obstacle avoidance algorithm (zoom-in).

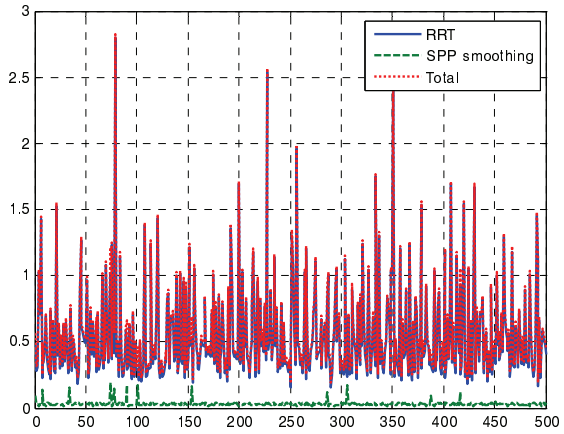


그림 15. 제안된 방법을 500회 반복했을 때의 실행 시간 그래프 ($p = 0.20$).

Fig. 15. The execution time graph when the proposed method is iterated 500 times ($p = 0.20$).

표 1. 목표 bias 확률 p에 따른 총 실행 시간.

Table 1. The total execution time for various goal bias probability p.

p	RRT (s)	SPP smoothing (s)	Total (s)
0.10	0.6303	0.0294	0.6597
0.20	0.5451	0.0312	0.5763
0.25	0.5088	0.0346	0.5435
0.30	0.5299	0.0338	0.5637
0.35	0.5478	0.0343	0.5821
0.40	0.5272	0.0352	0.5623
0.45	0.5833	0.0356	0.6189
0.50	0.6463	0.0349	0.6812

그림 15는 확률 $p = 0.20$일 때, 제안된 방법을 500회 반복한 실행 시간을 나타낸 그래프이다. 평균 실행 시간은 약 0.5763초로, RRT가 무작위 샘플링이기 때문에 편차가 심했다. 기존 RRT는 약 0.5451초의 실행 시간을 가지고 있었으며, SPP 평활화 및 장애물 회피 알고리즘이 전체 실행시간의 약 5.5%로, 무시할 수 있는 수준인 것으로 나타났다.

표 1은 확률 p에 따른 총 실행 시간을 나타낸 표이다. 모두 500회 반복한 평균 값이며, RRT와 SPP 평활화를 나누어서 나타내었다. RRT의 특성 때문에 확률 p와 총 실행 시간의 관계가 선형적이지 않다는 것을 알 수 있다. 실제 제안된 방법을 사용할 환경을 먼저 시뮬레이션 해보고 가장 적절한 확률 p를 선정하는 것이 중요하다.

마지막으로 대표적인 경로 계획 알고리즘인 A* 알고리즘과 제안된 방법의 성능 차이를 알아보았다. A* 알고리즘은 개발자가 선정한 휴리스틱 방법에 따라서 비용함수가 최소가 되는 방향으로 경로를 만드는 알고리즘으로써, 비용함수 $f(n)$은 다음과 같다.

$$f(n) = g(n) + h(n) \tag{9}$$

$g(n)$은 격자 한 칸을 이동할 때의 이동 거리이고, $h(n)$은 목적지까지의 거리이다. 대표적으로 A* 알고리즘에 사용하는 휴리스틱 방법은 맨해튼 방법, 직선 방법 등이 있으며, 이는 $h(n)$에 영향을 미친다.

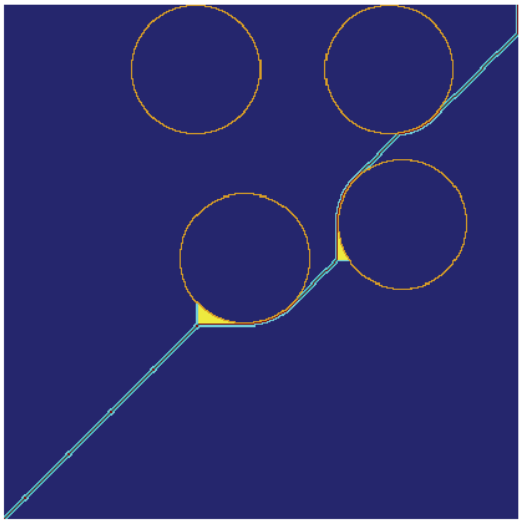


그림 16. A* 알고리즘을 사용한 경로 계획 (맨해튼 거리).

Fig. 16. The path planning using the A* algorithm (Manhattan distance).

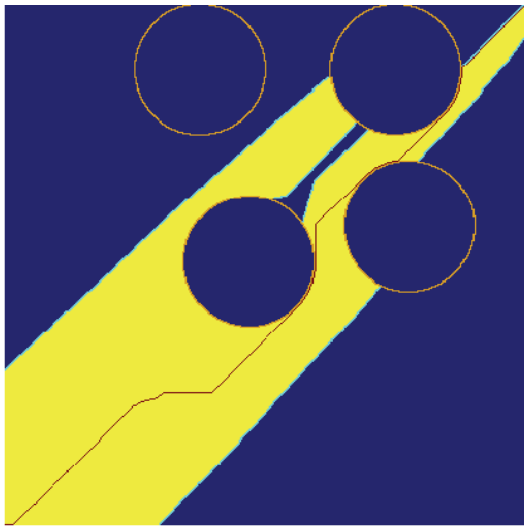


그림 17. A* 알고리즘을 사용한 경로 계획 (직선 거리).

Fig. 17. The path planning using the A* algorithm (Straight distance).

A* 알고리즘의 시뮬레이션을 위해 Alex Andriën의 report [16]에 수록된 프로그램을 사용하였으며, 프로그램을 구동한 PC는 상기 제안된 방법의 시뮬레이션에서 사용한 PC와 같다. 그래프 또는 트리를 이용한 알고리즘은 시뮬레이션을 할 때마다 다른 결과가 출력되는 무작위 알고리즘에 비해 항상 같은 경로를 출력하고 알고리즘 수행시간이 크게 달라지지 않으므로 1회의 시뮬레이션만 수행하였다.

그림 16은 맨해튼 거리를 사용한 A* 알고리즘을 사용한 경로 계획이고, 그림 17은 직선 거리를 사용한 A* 알고리즘을 사용한 경로 계획이다. A* manhattan은 총 2.971초의 실행 시간을 사용하였고, A* straight는 총 397.784초의 실행 시간을 사용하였다. 제안된 방법에 비해 매우 큰 실행 시간을 보여주었으며, 기구학적 제약조건을 만족하지 않는 경로가 생성되어 실제 실험에 적용하기 힘들 것이라는 것을 알 수 있다.

VI. 결론

일반적으로 사용되는 경로계획들은 계산량이 많거나, 직선만으로 이루어진 한계 때문에 계획된 경로를 그대로 로봇에 적용하지 않는다. 이 논문에서는 다른 경로계획들에 비해 계산량 및 계산 속도 측면에서 현저한 향상을 보여주고 있는 RRT를 이용하여 경로계획을 한 뒤, SPP 곡선을 이용한 평활화를 통해 로봇의 제약조건을 만족하는 경로가 생성 될 수 있는 방법을 제안하였다. 제안된 방법을 이용하여 시뮬레이션 한 결과 단순히 기존 RRT만 사용한 결과에 비하여 좋은 성능을 가지면서도 실행 시간이 다른 경로계획들에 비해 적었다. 기존 제시된 방법들에 비해 초기에 전역적으로 경로를 생성하기 때문에 추가적인 계산 부하가 발생하지 않고, 직선 경로를 최대한 이용함으로써 최단거리 경로를 만족하는 장점을 가진다는 것을 확인하였다. 제안된 방법을 통해서 경로계획만이 아닌 실제 로봇에 적용할 경로를 생성할 경우에도 로봇의 기구학적 제약조건을 고려하여 최적의 결과를 얻어낼 수 있을 것으로 기대된다.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, pp. 595-601, 2001.

[2] W. Zeng and R. L. Church, "Finding shortest paths on real networks: the case for A*," *International Journal of Geographical Information Science*, vol. 23, no. 3-4, pp. 531-543, Mar.-Apr. 2009.

[3] G.-Y. Song and J.-W. Lee, "Path planning for autonomous navigation of a driverless ground vehicle based on waypoints," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 20, no. 2, pp. 211-217, 2014.

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

[5] S. Anthony, "The focussed D* algorithm for real-time replanning," *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 1652-1659, Aug. 1995.

[6] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning," *International Journal of Robotics Research*, vol. 16, pp. 759-774, 1996.

[7] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[8] D.-H. Kim, Y.-S. Choi, R.-J. Yan, L.-P. Luo, J. Y. Lee, and C.-S. Han, "Efficient path planning of a high DOF multibody robotic system using adaptive RRT," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 21, no. 3, pp. 257-264, 2015.

[9] I. Jeong and J. Lim, "Trajectory generation for mobile robot," *The Conference of The Institute of Electronics and Information Engineers (in Korean)*, pp. 25-30, Oct. 1992.

[10] Henrie, Joshua and Wilde, Doran, "Planning continuous curvature paths using constructive polylines," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 12, pp. 1143-1157, 2007.

[11] H.-M. Lee, M.-H. Kim, and M.-C. Lee, "A UGV hybrid path generation method by using B-spline Curve's control point selection algorithm," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 20, no. 2, pp. 138-142, 2014.

[12] Y. J. Yoon, J. Kim, and D.-J. Kang, "The study of the autonomous driving method for tracking way point at the unmanned vehicle," *Proc. of 2011 26th Institute of Control, Robotics and Systems (ICROS) Annual Conference (in Korean)*, Gwangju, pp. 1-5, May 2011.

[13] C. Moon and W. Chung, "Motion planning scheme on the basis of RRT for a high-speed navigation of a two wheeled mobile robot," *Proc. of the Annual Spring & Fall Conference of The Korean Society of Mechanical Engineers (in Korean)*, pp. 2178-2183, 2013.

[14] C. Urmson and R. Simmons, "Approaches for heuristiclly biasing RRT growth," *Proc. of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1178-1183, 2003.

[15] P. C. Mahalanobis, "On the generalised distance in statistics," *Proceedings National Institute of Science*, India, vol. 2, no. 1, pp. 49-55, Apr. 1936.

[16] A. Andriën, "Topology optimization versus A*: a comparison for 2D robot path planning," Eindhoven University of Technology, Netherlands, Report, Dec. 2012.

박 영 상

2015년 인하대학교 전자공학과 졸업. 2015년~현재 인하대학교 대학원 전기공학과 석사과정 재학 중. 관심분야는 경로 계획, 플랜트 제어, 알고리즘.

이 영 삼

제어 · 로봇 · 시스템학회 논문지, 제15권 제4호 참조.