개선한 휴리스틱 함수를 사용한 A* 기반 안전한 경로 계획

남승우, 유경민, 박재원, 김성현, 김명섭*

고려대학교

{nam131119@korea.ac.kr, rudals2710, 2018270614, pb1069, *tmskim}@korea.ac.kr

A* based safe path planning with improved heuristic function

Seung-woo Nam, Gyeong-Min Yu, Jae-Won Park, Ui-Jun Baek, Myung-Sup Kim*

Korea Univ.

요 약

경로 계획(Path planning)은 자율주행에서 핵심적인 기술로, 에이전트가 목적지까지 안전하고 효율적으로 이동할 수 있도록 해준다. 경로 계획을 위해 Dijkstra, A*[1], RRT와 같은 다양한 알고리즘들이 제안되어왔다. 빠른 이동을 위한 경로 길이 최소화도 중요하지만, 동시에 목적지까지의 안전한 이동을 보장하는 경로를 고려하는 것도 매우 중요하다. 급격한 회전이 포함되거나 장애물에 너무 가까운 경로는 실제 로봇 내비게이션에서는 적합하지 않을 수 있으며, 로봇이 해당 경로를 따라가기가 어려울 수도 있다. 본 연구에서는 A* 알고리즘의 휴리스틱 함수를 개선하여 곡률을 줄이고 장애물에 가까이 가지 않는 경로를 계획하는 방법을 제안한다. 새로운 곡률 비용 함수와 장애물 비용 함수를 휴리스틱에 도입함으로써, 제안된 방법은 기존의 향상된 A* 알고리즘보다 곡률이 줄어들고 장애물과의 간격이 더 확보된 경로를 생성할 수 있다.

Ⅰ. 서 론

자율 주행 기술 중 경로 계획은 에이전트가 목적지까지 이동하기 위해 필요한 경로를 계획하는 기술이다. 이 기술은 로봇, 공학, 항공 등 다양한 분야에서 사용하고 있으며, 특히 효율성, 시간 절약, 비용 절감 등의 목표를 달성하는 데 필수적인 요소로 작용한다. 경로 계획 문제는 흔히 지도 데이터, 장애물, 거리 등을 고려해 최단 거리를 찾는 문제로 정의하며, 경우에 따라 복잡한 제약 조건과 변수도 고려할 수 있다.

1.1 A* 알고리즘

경로 계획에서는 Dijkstra, A*, RRT 등 다양한 알고리즘이 제안되었다. Dijkstra 알고리즘은 탐욕적 전략을 기반으로 탐색을 진행하기 때문에 실용성을 고려하지 않고 최단 경로만 고려한다. 이는 지도를 완전히 탐색해야 하므로 계산량이 많고, 효율이 낮으며, 충돌 회피 능력이 약하다. Dijkstra 알고리즘의 높은 계산량 문제를 극복하기 위해 제안된 A* 알고리즘은 최단 경로 탐색 문제를 해결하기 위한 대표적인 휴리스틱 기반 알고리즘이다. A* 알고리즘은 시작 노드에서 목표 노드까지의 총 비용을 평가하기 위한 비용 함수 $F(n)$을 기반으로 경로를 계획한다. $F(n)$은 식 (1)과 같다.

$$F(n) = G(n) + H(n) \quad (1)$$

$G(n)$은 시작 노드에서 현재 노드 n까지 이동하는 데 소요된 실제 경로 비용으로, 탐색 과정에서 이미 지나온 경로의 누적 비용을 계산한다. $H(n)$은 현재 노드 n에서 목표 노드까지 이동하는 데 소요될 것으로 예상되는 비용을 추정하는 휴리스틱 함수이다. $H(n)$은 문제에 따라 정의하며, 경로 계획 문제에서는 목표 노드까지의 거리를 예측하기 위해 거리 측도 식을 사용한다. 대표적으로 맨해튼 거리(Manhattan Distance), 대각선 거리(Diagonal Distance), 유클리디안 거리(Euclidean Distance)가 존재한다.

이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(00230661, 하이브리드 양자키분배 방법 및 망 관리 기술 표준개발) 및 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(00235509, ICT융합 공공 서비스• 인프라의 암호화 사이버위협에 대한 네트워크 행위기반 보안관제 기술 개발)을 받아 수행된 연구임.

경로 계획 분야에서 현재까지 A* 알고리즘을 개선한 방법론들이 제안되어왔다. 휴리스틱 함수 개선[2]부터, 탐색 방법[3], 비용 평가 함수 개선[4] 등 다양한 방법론이 제안되었다. 하지만 다양한 환경에서 테스트하지 않았으며, 특정 환경에서는 경로 계획 실패하는 현상까지 발생한다. 이러한 문제를 해결하기 위해, 휴리스틱 함수에 곡률 비용 함수와 새로운 장애물 비용 함수를 추가하여 장애물을 안전하게 회피하며 최소 시간을 가지는 최적 경로를 계획하는 A* 알고리즘을 제안하고자 한다.

Ⅱ. 본론

자율 주행에서 경로 계획은 중요한 도구로 높은 시장 수요, 넓은 응용 가능성, 그리고 큰 개발 잠재력으로 인해 많은 연구자들의 주목을 받고 있다. 그러나 기존 A* 알고리즘은 거리 측도 식만 사용하기 때문에 급격한 회전이 필요하며 장애물에 인접하여 회전하는 경로가 계획될 수 있다. 본 연구에서는 경로 계획 문제를 해결하기 위해 개선된 A* 알고리즘을 제안한다.

2.1 곡률 비용 함수

기존 A* 알고리즘이 생성한 경로는 급격한 회전이 필요할 수 있다. 만약 이 경로를 에이전트가 사용할 경우, 에이전트는 경로를 제대로 따라가지 못할 수 있으며, 이는 이동 비용이 증가할 수 있다. 따라서 경로 계획 시 곡률을 감소시키는 것이 필수적이다. 이러한 곡률 감소를 위해 기존 휴리스틱 함수에서 곡률 비용 함수를 추가한다. 기존 휴리스틱 함수에 곡률 비용 함수를 추가한 식은 식 (2)와 같다.

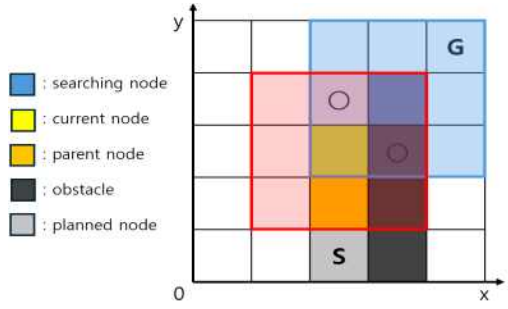

그림 1. 장애물 확인을 위한 각 노드에 대한 인접 노드 확인

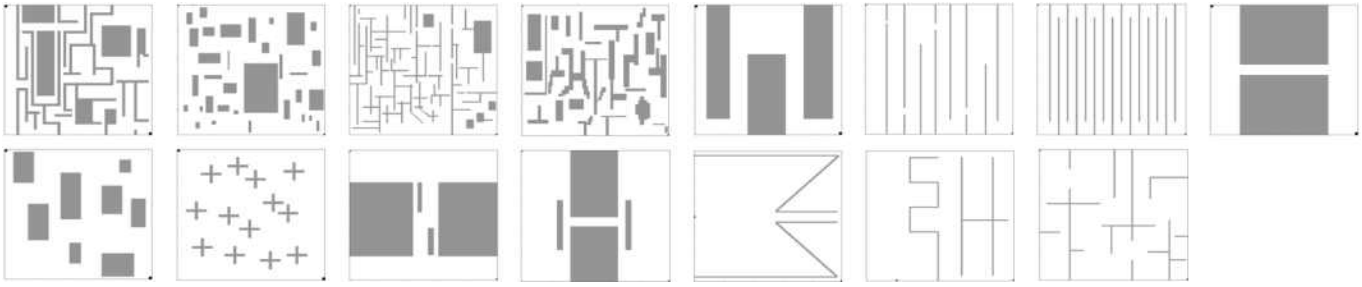

그림 2. 15가지의 맵에 대한 정보.

$$F(n) = G(n) + H(n) + C(n) \quad (2)$$

식 (2)에서 n은 노드이며, $C(n)$이 곡률 비용 함수이다. 곡률 비용 함수 $C(n)$은 탐색 노드와 현재 노드, 현재 노드의 부모 노드 총 3개의 노드를 사용하여 경로 곡률을 계산한다. 헤론 공식을 사용하여 세 노드로 구성되는 삼각형 넓이를 구하고, 구한 넓이를 활용하여 곡률을 계산한다. 자세한 식은 식 (3)~(5)와 같다.

$$s = \frac{Euclidean(n_{i-2},n_{i-1}) + Euclidean(n_{i-1},n_i)}{2} + \frac{Euclidean(n_{i-2},n_i)}{} \quad (3)$$

식 (3)에서 $Euclidean(n_{i-1},n_i)$는 현재 노드 n_{i-1}과 후보 노드 n_i 사이의 유클리디안 거리이다. 현재 노드의 부모 노드 n_{i-2}, 현재 노드 n_{i-1}, 후보 노드 n_i 총 세 노드 각각 사이의 거리를 구하고 이를 기반으로 s값을 구한다.

각 노드 간 유클리드 거리와 식 (3)에서 구한 s를 사용하여 삼각형의 면적 Δ을 계산한다. 삼각형의 면적을 구하는 식은 식 (4)와 같다.

$$\Delta = \sqrt{s \cdot (s - Euclidean(n_{i-2},n_{i-1}) \cdot \sqrt{(s - Euclidean(n_{i-1},n_i) \cdot \sqrt{(s - Euclidean(n_{i-2},n_i)}} \quad (4)$$

삼각형의 면적 Δ을 사용하여 곡률을 계산한다. 계산 식은 식 (5)와 같다.

$$C(n) = \frac{4 \cdot \Delta}{Euclidean(n_{i-2},n_{i-1}) \cdot Euclidean(n_{i-1},n_i) \cdot Euclidean(n_{i-2},n_i)} \quad (5)$$

앞서 언급한 식을 사용하여 구한 곡률 값은 비용 함수에 더해지며, 이는 경로 탐색 시 곡률이 커지는 경로는 비용이 증가하게 되므로, 급격한 회전보다 완만한 경로를 가지는 후보 노드를 선택하며 경로를 계획하게 된다.

본 연구에서는 곡률 비용을 효율적으로 사용하기 위해 곡률 계산에 사용하는 노드 수를 설정할 수 있도록 하였다. 개선한 곡률 함수 식인 $C(n,k)$은 식 (6)과 같다.

$$C(n,k) = \frac{\sum_{j=1}^{k} C(n_{i-j+1})}{k} \quad (k \geq 1) \quad (6)$$

k는 1 이상의 값으로 설정하는 상수로, k가 1일 때 기존 곡률 함수와 동일하며, k가 2일 경우 탐색 노드 n_i 기준 곡률 값과 현재 노드 n_{i-1} 기준 곡률 값을 구하여 구한 곡률들의 평균 값을 사용한다. 이를 통해 현재 노드 기준 이전 노드들의 곡률을 고려하며 탐색을 진행할 수 있다.

2.2 장애물 비용 함수

기존 A* 알고리즘은 장애물에 인접하여 회전하는 경로를 계획한다. 이러한 경로는 로봇이 경로를 따라가면서 회전하는 도중 장애물과 충돌할 수 있으며, 이는 자율 주행 성능에 큰 악영향을 끼친다. 따라서 장애물에 인접한 상태에서 회전하지 않기 위해 장애물 비용 값을 계산한다. 탐색 노드 n_i와 현재 노드 n_{i-1} 각각 8방향의 인접 노드가 서로 겹치는 위치에 장애물이 있으며 노드 n_i, n_{i-1}, n_{i-2} 가 한 직선에 존재하지 않을 때 비용을 추가한다. 그림 1은 두 노드 각각의 인접 노드가 겹치는 위치에 장애물이 존재하는 예시를 보여주는 그림이다. 시작 노드 S에서 목표 노드 G까지 이동하는 경로를 계획할 때, 탐색 노드 (4,4)의 파란색 영인 8방향의 인접 노드와 현재 노드의 인접 노드(3,3)으 빨간색 영역인 8방향의 인접 노드 8개가 겹치는 위치는 동그라미 표시가 되어있는 (3,4), (4,3)이다. 두 위치 중 (4,3) 노드에 장애물이 존재하며 세 노드가 한 직선에 있지 않으므로 비용을 추가한다. 장애물 비용 함수까지 추가한 최종 식은 식 (7)과 같다.

$$F(n) = G(n) + H(n) + C(n,k) + O(n) \quad (7)$$

Ⅲ. 실험

제안한 알고리즘을 테스트하고 비교하기 위해 2차원 그리드 환경의 맵에서 경로를 계획하고자 한다. 2차원 그리드 환경은 경로 계획 알고리즘의 평가를 위한 표준적인 테스트베드로 사용된다. 컴퓨팅 자원은 ubuntu 20.04 Desktop 운영체제, RAM 24GB, Intel® Core™ i5-9600KF CPU @ 3.70GHz이며 Python을 사용하여 경로 계획 알고리즘을 구현하였다. 맵의

그림 3. 6번 맵에서 각 알고리즘에 대한 경로 계획 결과. 왼쪽부터 A*, AdaptA*[1], EBHSA*[2], XiaoA*[3], proposed A*

Map No.	경로 내 점 수					장애물과 가까운 점 수					경로 평균 곡률					경로 곡률 편차					소요 시간				
	A*	AdapA*	EBHSA*	XiaoA*	Proposed	A*	Adapt A*	EBHSA*	XiaoA*	Proposed	A*	Adapt A*	EBHSA*	XiaoA*	Proposed	A*	Adapt A*	EBHSA*	XiaoA*	Proposed	A*	Adapt A*	EBHSA*	XiaoA*	Proposed
1	**135**	**135**	-	138	141	124	122	-	124	82	**0.14**	0.16	-	0.23	0.15	0.21	0.21	-	0.29	**0.19**	**0.01**	0.03	-	0.11	0.06
2	**129**	**129**	153	132	169	67	41	**0**	45	4	0.1	0.12	0.09	0.12	**0.03**	0.15	0.16	0.13	0.19	**0.09**	**0.01**	0.17	0.03	0.78	0.03
3	**222**	226	-	225	238	157	95	-	147	**78**	0.13	**0.12**	-	0.29	0.12	0.21	0.18	-	0.31	**0.18**	**0.07**	0.21	-	0.31	0.35
4	**145**	**145**	166	148	157	93	64	**0**	60	45	0.21	0.15	0.1	0.22	**0.08**	0.15	0.17	0.2	0.24	**0.13**	**0.01**	0.15	0.07	0.74	0.15
5	**128**	**128**	146	128	132	105	85	**0**	99	8	0.06	0.11	0.1	0.12	**0.06**	0.13	0.18	0.2	0.21	**0.11**	**0.01**	0.04	0.04	0.14	0.04
6	**313**	316	-	317	329	234	86	-	161	**12**	0.05	0.07	-	0.26	**0.04**	0.15	0.16	-	0.29	**0.11**	**0.16**	0.28	-	0.89	0.51
7	**1234**	1248	1291	1241	1279	1176	174	**0**	811	27	0.03	0.04	0.05	0.26	**0.03**	0.13	0.12	0.14	0.27	**0.12**	**0.22**	0.32	0.33	0.92	0.58
8	**76**	**76**	81	78	84	43	21	**0**	16	0	0.05	0.14	0.71	0.06	**0.04**	**0.1**	0.17	0.13	0.11	0.11	**0.01**	0.03	0.01	0.11	0.02
9	70	**67**	75	68	80	43	29	**0**	16	10	0.10	0.13	0.16	0.11	**0.04**	0.15	0.16	0.19	0.12	**0.11**	**0.01**	0.06	0.01	0.2	0.01
10	**54**	**54**	61	56	57	14	14	**0**	14	12	**0.07**	0.11	0.19	0.11	0.10	0.14	0.17	0.2	0.14	**0.13**	**0.01**	0.04	0.01	0.21	0.02
11	**140**	146	148	141	155	65	74	**0**	35	1	0.04	0.13	0.04	0.11	**0.02**	0.09	0.19	0.11	0.21	**0.07**	**0.01**	0.15	0.01	0.56	0.04
12	**131**	**131**	138	133	157	62	61	**0**	33	2	0.04	0.08	0.07	0.05	**0.03**	0.09	0.14	0.14	0.12	**0.09**	**0.01**	0.15	0.0	0.54	0.04
13	**143**	**143**	146	144	144	44	48	**0**	2	44	0.01	0.03	0.04	0.15	**0.01**	**0.06**	0.10	0.12	0.25	0.07	**0.06**	0.26	0.11	1.1	0.25
14	**242**	**242**	251	249	254	139	104	**0**	106	2	0.02	0.03	0.04	0.18	**0.01**	0.08	0.1	0.13	0.27	**0.06**	**0.08**	0.31	0.5	1.57	0.33
15	**166**	**166**	-	166	170	34	48	-	16	**15**	0.06	0.05	-	0.12	**0.05**	0.12	0.13	-	0.22	**0.12**	**0.16**	0.28	-	1.04	0.53

표. 1. 실험 결과

크기는 50 x 50 크기와 100 x 100 크기의 맵 두 종류이며 각각 5가지, 10가지의 맵으로 구성하였다. 맵 형태는 그림 2와 같다. 왼쪽 위 맵이 1번 맵이며 오른쪽 방향 순서가 된다. 100 x 100 크기의 그리드 맵은 2-4, 6, 7, 11-15번 맵이며, 50 x 50 크기의 그리드 맵은 1,5,8-10번 맵이다. 출발지와 목적지는 13,14번 맵을 제외하면 각각 왼쪽 위 모서리 끝 점 (0,100)과 오른쪽 아래 모서리 끝 점(100,0)이다. 13,14번 맵의 목적지는 다른 맵과 같으며 13번 맵의 출발점은 (0,50)이고 14번 맵의 출발점은 (20,0)이다. 맵 종류는 크게 세 가지로 나누었으며, 1~4번 맵은 복잡한 환경, 5~7번 맵은 노드 탐색이 많은 환경, 8~15번 맵은 넓은 환경으로 구성하였다.

제안한 A* 알고리즘의 성능 비교를 위해 현재까지 제안된 A* 알고리즘과의 비교 실험을 진행하였다. 비교 알고리즘은 A*, AdaptA*[2], EBHSA*[3], XiaoA*[4]으로 총 4가지이며 제안한 알고리즘인 Proposed A* 알고리즘까지 해서 5가지의 알고리즘을 15가지의 맵에서 경로 계획을 진행하였다. 그림 3은 4번 맵에서 각 알고리즘으로 경로를 계획한 결과이다. 왼쪽부터 차례대로 A*, AdaptA*, EBHSA*, XiaoA*, Proposed A* 알고리즘으로 계획한 결과이다. A*, AdaptA*, XiaoA* 알고리즘은 장애물에 인접한 상태에서 회전을 하지만 EBHSA*, Proposed A* 알고리즘은 장애물에 인접하여 회전하지 않는 경로를 계획한 것을 볼 수 있다. EBHSA* 알고리즘에서는 양방향 탐색으로 인해 특정 구간에서 비효율적인 경로가 계획되었다. 전체 실험 결과는 표 1에 정리하였다.

실험 결과, Proposed A* 알고리즘은 장애물과 가까운 점 수, 경로 평균 곡률, 경로 곡률 편차에서 좋은 성능을 보여주었다. 경로 내 점 수에서는 A* 알고리즘이 좋은 성능을 보여주었다. 기존 A* 알고리즘에서는 급격한 회전을 허용하기 때문에 다른 알고리즘보다 경로 내 점 수가 적은 것을 확인할 수 있다. 장애물과 가까운 점 수에서는 EBHSA* 알고리즘이 확장 거리로 인해 최고의 성능을 보여주었지만, 확장 거리로 인하여 좁은 구역을 이동하지 못하여 1,3,6,15번 맵에서 경로를 계획하지 못하는 현상이 발생하였다. 따라서 EBHSA* 알고리즘을 제외한 알고리즘 중에서는 Proposed A* 알고리즘이 장애물과 가까운 점의 수에서 가장 좋은 성능을 보여주었다. 경로 곡률에서는 모든 환경에서 Proposed A* 알고리즘이 우수한 성능을 보여주었다. 경로 평균 곡률에서는 복잡한 환경인 1, 3번 맵에서 각각 A*, AdaptSearA* 알고리즘이 Proposed A* 알고리즘보다 좋은 성능을 보여주었지만 최대 0.0049 차이밖에 발생하지 않았다. 경로 곡률 편차에서는 복잡한 환경과 노드 탐색이 많은 환경에서 좋은 성능을 보여주었다. 넓은 환경에서도 8,13번 맵을 제외한 다른 맵에서도 좋은 성능을 보여주었다. 이는 대부분의 환경에서 안전한 경로 계획 관점에서 강인한 성능을 보여주는 모습을 보여주었다. 계산 시간에서는 추가 연산을 하지 않는 A* 알고리즘이 가장 좋은 성능을 보여주었지만, 모든 알고리즘이 대부분 1초를 넘기지 않고 경로를 계획하는 것을 확인할 수 있었다.

Ⅳ. 결론

본 논문에서는 A*알고리즘의 휴리스틱 함수에 곡률 비용 함수와 장애물 비용 함수를 추가한 새로운 경로 계획 알고리즘을 제안하였다. 곡률 비용 함수는 경로의 곡률을 최소화하기 위해 세 점을 헤론 공식을 기반으로 새롭게 곡률을 계산하여 회전이 적은 경로를 생성할 수 있도록 하였으며, 장애물 비용 함수는 현재 노드와 탐색 노드의 인접 노드를 활용하여 장애물에 인접하지 않고 회전하는 경로를 계획할 수 있도록 하였다.

기존 A* 알고리즘 및 기타 개선된 A* 알고리즘과의 비교 실험에서 제안한 알고리즘은 다양한 환경에서도 경로의 부드러움, 장애물과의 거리 측면에서 우수한 성능을 보였다. 특히 복잡한 환경과 노드 탐색이 많은 환경에서도 제안한 알고리즘이 안전한 경로를 생성할 수 있음을 확인하였다.

본 연구는 경로 곡률 감소와 장애물 회피를 동시에 고려한 알고리즘의 효과를 입증함으로써 자율 주행, 로봇 공학 등 다양한 응용 분야에서 안전하고 효율적인 경로 계획의 가능성을 제시하였다.

참 고 문 헌

[1] TAN, Chee Sheng; MOHD-MOKHTAR, Rosmiwati; ARSHAD, Mohd Rizal. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. IEEE Access, 2021, 9: 119310-119342.

[2] ZHANG, Jing, et al. Research on effective path planning algorithm based on improved A* algorithm. In: Journal of Physics: Conference Series. IOP Publishing, 2022. p. 012014.

[3] WANG, Huanwei, et al. An efficient and robust improved A* algorithm for path planning. Symmetry, 2021, 13.11: 2213.

[4] SA, Xiao; HUAIYU, Wu; ZHIHUAN, Chen. Research of mobile robot path planning based on improved A* algorithm. In: 2020 Chinese Automation Congress (CAC). IEEE, 2020. p. 7619-7623.