무인 자동차를 위한 실시간 모션 플래닝 계산 기법

Dubins path를 이용한 RRT* 기반의 실시간 무인 자동차 모션 플래닝

RRT* based motion planning for real-time autonomous vehicle using Dubins path

김동혁, Donghyuk Kim*, 이정환, Junghwan Lee**, 윤성의, Sung-Eui Yoon***

요약˘˘모션 플래닝 분야에서 실시간으로 주변을 감지하고 움직이는 로봇의 경우 변화하는 장애물들에 대한 동적인 감지 및 분석과 제한된 경로 계산 시간 등의 제약이 따른다. 특히 무인 자동차와 같이 제어의 제약을 고려한 충돌이 없는 유효 경로 생성은 그 자체 만으로도 중요한 이슈이다. 본 연구에서는 샘플링 기반의 알고리즘인 RRT* 를 이용한 무인 자동차의 경로 생성기법에 대한 연구와 실시간 경로 생성을 위해 고려되는 부가적인 요소들의 유기적인 결합에 대해 소개한다.

Abstract˘˘In motion planning field, it is necessary for the mobile robots which have ability to recognize the surrounding environments and move by itself, to detect, analyze the obstacles and compute feasible trajectory in real time. Especially, in case of car-like robots which have control constraints, it is challenging to only generate a feasible trajectory in a limited time. In this paper, we extend sampling based algorithm RRT* for autonomous vehicle with control constraints to takes advantage of dubins path, and its simplicity for easing trajectory calculation. In addition, we show some simple collision detection technique from dubins path.

핵심어: Real-time Motion Planning, RRT, Dubins path, Autonomous Vehicle.

*주저자 : 한국과학기술원 전산학과 석사과정 e-mail: bishopak@gmail.com
**공동저자 : 한국과학기술원 전산학과 박사과정 e-mail: goolbee@gmail.com
***교신저자 : 한국과학기술원 전산학과 교수 e-mail: sungeui@gmail.com

1. 서론

모션 플래닝 분야에 대한 연구는, 군사적 혹은 학술적 목적의 각종 로봇에서 실제로 쓰이고 인간의 직접 제어가 불가능한 화성 탐사선 등에서 필수적으로 요구되어 왔다.

무인 자동차 분야의 경우, 2006, 2007 년도에 개최된 DARPA Grand Challenge[1, 2]를 통해 많은 기술들이 소개되었고, 특히 최근 구글(Google)에서 발표한 Google self-driving car 는 현재로서 가장 상용화에 가까이 다가간 모델로 평가 받고 있다. 이러한 무인 자동차의 경우 장애물 정보가 매 순간마다 다른 동적 환경에서의 경로 생성을 요구하기 때문에 필연적으로 계산 시간에 민감하게 된다. 때문에 Dynamic A* [3]와 같은 휴리스틱 기법이 선호되어왔고 일반적으로 RRT[4]와 같은 sampling based method 의 경우 optimality 의 보장성과 계산 속도의 문제로 적용에 어려움이 있었다. 하지만 RRT*[5]의 등장으로 optimality 에 대한 문제가 해결되고 각종 가속 기법에 대한 후속 연구로 인해 RRT 를 이용한 실시간 모션 플래닝에 대한 연구도 활발히 진행되고 있다.

본 연구에서는 이러한 optimality 를 보장하는 RRT* 를 기반으로, 실시간으로 입력되는 점구름으로 이루어진 장애물 정보에 대해 Collision-free 한 경로의 실시간 생성기법과 가속을 위한 여러 방법들의 조합에 대한 논의를 담고 있다.

2. 문제 정의

본 연구에서 차량은 Dubins car[6]로 가정하였으며, 구성 공간은 $\mathrm{X} \subset \mathrm{SE}(2) = R^2 \times S^1$ (평면 상 위치와 회전), 제어 입력 공간은 $\mathrm{U} \subset R^n$ 로 정의된다. 차량의 움직임은 오직 일정한 속도 v로 전진만을 고려하며, 제어 입력 공간 U는 차량의 steering angle 에 맵핑가능하며 일반적인 차량의 경우 $\left[-\frac{\pi}{\mathrm{k}}, \frac{\pi}{\mathrm{k}}\right], \mathrm{k} > 2$ 를 갖는다. 시간 $t_{\geq 0}$ 에 대한 차량의 움직임 변화는 다음과 같이 수식으로 나타낼 수 있다.

$$x = (x,\ y,\ \theta),\quad \dot{x}(t) = f(x(t), u(t)),\quad x \subset X,\quad u \subset U \quad (1)$$

여기서 제어 입력 u 는 Dubins steering 을 표현하며, 차량에 따라 다른 시작 상태 x_{init} 와 종료 상태 x_{goal} 의 관계는

$$x_{goal} = x_{init} + \int \dot{x}(t)dt \quad (2)$$

로 정의된다. 또한 전체 구성 공간 X 에 대하여, X_{obs} 와 X_{goal} 은 각각 장애물로 인해 위치 불가능한 공간 집합과 목표 종료 상태의 집합을 의미하며, 반대로 위치 가능한 공간은 $X_{free} \coloneqq X \setminus X_{obs}$ 로 정의된다. 이러한 정의 하에, 주어진 x_{init}, x_{goal} 에 대하여 수식 (2)를 만족하며, 어떤 한 순간 t 에 대하여 $\mathrm{x(t)} \in X_{free}$ 를 만족하는 일련의 제어 입력 u를 계산하는 것으로 모션 플래닝 문제를 수식화 할 수 있다.

그림 1 현대 자동차에서 주최한 자율 주행 자동차 경진대회에서 사용된 각종 센서를 장착한 한국과학기술원 팀의 차량 사진

3. RRT* 알고리즘과 Dubins path 경로 생성 함수

3.1 RRT* 알고리즘

RRT* 알고리즘은 점진적 샘플링 기반의 모션 플래닝 알고리즘으로, RRT 와 달리 해가 점진적으로 최적해로의 수렴성을 보장하기 위해 고안된 알고리즘이다. RRT 는 샘플된 구성 상태의 부모 노드를 결정하기 위해 RRT 내의 노드 중 가장 가까운 노드로 연결하며 이 관계는 RRT 가 종료되기까지 변하지 않는다. 반면 RRT*의 경우 부모를 결정하는 단계에서 시작 상태 x_{init} 으로 부터 각 노드까지의 비용을 고려하여 최적의 부모를 결정한다.

알고리즘 1 : RRT*

$\mathrm{V} \leftarrow \{x_{init}\}; E \leftarrow \emptyset;$

while the termination condtion is not satisfied **do**:

$\mathrm{G} \leftarrow (\mathrm{V}, \mathrm{E});$

$x_{sample} = Sample();$

$(\mathrm{V}, \mathrm{E}) \leftarrow Extend(G, x_{sample})$

알고리즘 2 : Extend(G, x_{sample})

$V' \leftarrow V; E' \leftarrow E;$
for i = 0, … , n **do**:
  $x_{nearest} \leftarrow$ **Nearest**(G, x_{sample});
  $t_{nearest,sample} \leftarrow$ **Trajectory**$(x_{nearest}, x_{sample})$;
  if isCollisionFree$(t_{nearest,sample})$:
    $V' \leftarrow V' \cup \{x_{sample}\}$;
    $x_{parent} \leftarrow x_{nearest}$;
    $c_{min} \leftarrow$ **Cost**$(x_{nearest})$ + **Cost**$(t_{nearest,sample})$;
    $X_{near} \leftarrow$ **NearNeighbors**(G, x_{sample}, δ);
    for all $x_{near} \in X_{near}$:
      $t_{near,sample} \leftarrow$ **Trajectory**(x_{near}, x_{sample});
      if isCollisionFree$(t_{near,sample})$:
        if Cost(x_{near}) + **Cost**$(t_{near,sample}) < c_{min}$:
          $c_{min} \leftarrow$ **Cost**(x_{near}) + **Cost**$(t_{near,sample})$;
          $x_{parent} \leftarrow x_{near}$;
    $E' \leftarrow E' \cup \{(x_{parent}, x_{sample})\}$;
    Rewire$(E', X_{near}, x_{sample})$;

Rewire 함수는 X_{near} 에서 x_{sample} 의 최적의 부모를 찾는 과정을 역으로 하여, x_{sample} 을 최적의 부모로 갖는 노드를 X_{near} 에서 찾아 연결한다. 앞서 사용된 **Nearest** 와 **NearNeighbors** 함수에서 사용된 거리 함수는 유클리디안 거리가 사용되었으며 **Cost**는 **Trajectory**에서 계산한 실제 경로의 길이로 계산되었다. **NearNeighbors**(G, x_{sample}, δ) 에서 δ는 x_{sample}로부터 거리가 δ이하인 이웃 노드 집합을 계산하는 함수이며, 이 지금까지 생성된 노드 개수 |V|에 비례하여 결정된다.

알고리즘 3 : Rewire$(E, X_{near}, x_{sample})$

for all $x_{near} \in X_{near}$:
  $t_{sample,near} \leftarrow$ **Trajectory**$(x_{sample,near})$;
  if isCollisionFree$(t_{sample,near})$:
    if Cost(x_{sample}) + **Cost**$(t_{sample,near})$ < **Cost**(x_{sample}):
      $E \leftarrow E \cup \{(x_{sample}, x_{near})\} - \{(Parent(x_{near}), x_{near})\}$;

3.2 Dubins path 경로 생성 함수

알고리즘 2. 에서 언급된 **Trajectory**(x_a, x_b)는 차량의 제어 제약을 만족하며 상태 x_a 로부터 x_b로 가는 Dubins path[6, 7] 기반의 유효 경로를 생성하여 반환하는 함수다. 임의의 상태 짝 $x_a, x_b \in X$에 대해서 주어진 공간 X = X_{free}임을 가정시, 최단 경로인 Dubins path 가 항상 존재하며 총 6 개 카테고리로 분류된다. 각 패턴은 선분 혹은 최소회전 반경을 따르는 부분 원호의 3 개 조합으로 표현된다.

RLR 과 LRL 은 상대적으로 출현 빈도가 낮고, 차량이 유턴과 같은 큰 선회를 할 경우 최적 경로로 고려되어 본 연구에서는 고려되지 않았다. **Trajectory** 함수가 호출될 경우 위에서 열거한 4 가지 패턴 중 장애물과의 충돌이 없으며 경로의 길이가 최저인 패턴이 선택되어 그 정보를 RRT* 상의 노드에 저장되고, 모든 경로가 생성된 이후 차량에 전달할 제어 입력 계산을 위해 이용된다.

3.3 경로 충돌 검사

본 연구에서 가정한 무인 자동차의 경우 매 순간마다 차체의 레이저 센서로부터 읽어 들인 점 구름 정보를 토대로 장애물이 포함된 환경을 인지한다. Dubins path 의

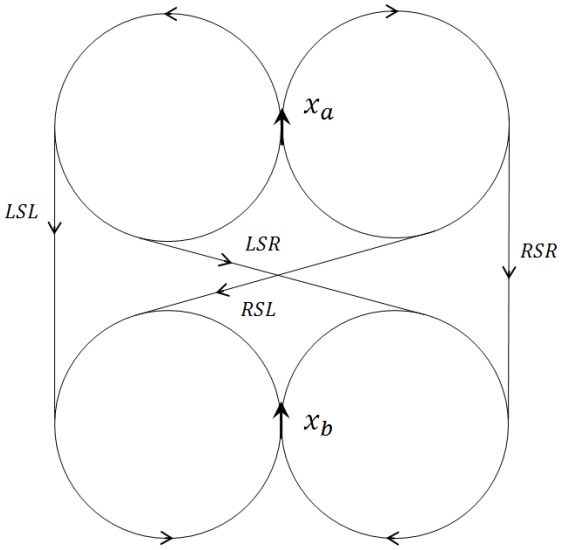

그림 2. 주어진 임의의 두 상태 x_a 로 부터 x_b 로 가는 최단 경로의 분류인 RSL(R:오른쪽 회전, S:직진, L: 왼쪽 회전), LSR, RSR, LSL, RLR, LRL 중 RLR, LRL 을 제외한 나머지의 예

특징 및 장애물 정보의 단순함에 기인하여 각 점에 대해 그림 3 의 방식으로 충돌 검사를 단순화 할 수 있다. x와 y는 각각 수식 3 에 표기된 바와 같으며, w, h, ρ 는 각각 차체의 폭, 길이, 최소 회전 반경을 나타낸다.

$$x = -2\rho; \quad y = \sqrt{4(\frac{w^2}{4} + w\rho + h^2)} \quad (3)$$

차량의 형태를 직사각형으로 가정하였을 때, 선분 형태의 경우는 선분과 점 사이 거리로 충돌 여부를 측정할 수 있으며, 원호의 경우 해당 원호의 원점으로부터 원호의 시작과 끝 지점 범위 내 각도에 들어오는 장애물들에 대해 원호의 원점과의 거리 계산을 통해 충돌 체크의 간소화가 가능하다. 점 구름의 크기가 클 경우, 충돌 가능성이 있는 점 집합을 줄이기 위해 앞서 언급된 **NearNeighbors** 을 이용한 가속이 가능하다. 차량이 움직임에 따라 점유하는 공간을 2차원 상의 물체로 고려해 보면, 이는 원의 집합으로 근사가 가능하며, 경로상에 일정 간격마다 원을

배치한 것과 동일한 형태로 나타내진다. 이 원의 반지름을 **NearNeighbors** 함수 호출 시의 δ 로 이용하여, 충돌 가능성이 있는 점 집합에 대해서만 충돌 체크를 수행하는 것으로 계산 시간을 낮출 수 있다.

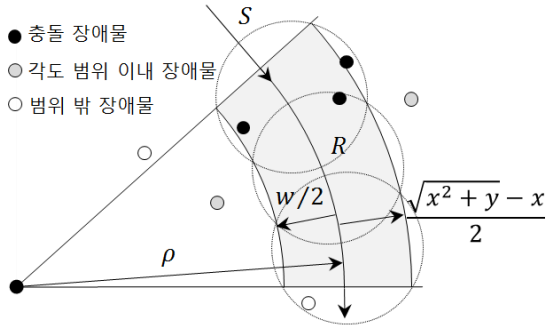

그림 3. 원호로 구성된 차량 경로와 충돌체크의 예.

4. 실험 결과

그림 4 의 결과는 Labview 로 구현된 시뮬레이터가 사용되었으며 빨간색은 장애물 정보를 나타내며 매번 입력되는 정보가 갱신될 때 마다, 기존에 계산한 경로의 유효성을 검사(새로 감지된 장애물과의 충돌 여부)하여 유효하지 않는 경우 새로 계산된 경로를 취하며, 유효한 경우 기존 경로와 비용을 비교하여 더 최적의 경로를 취하도록 구현되었다. 초기에는 시작 상태와 종료 상태를 연결하는 직선 방향으로 차량이 이동하며, 실시간으로 충돌이 없는 유효 경로를 계산해낸다.

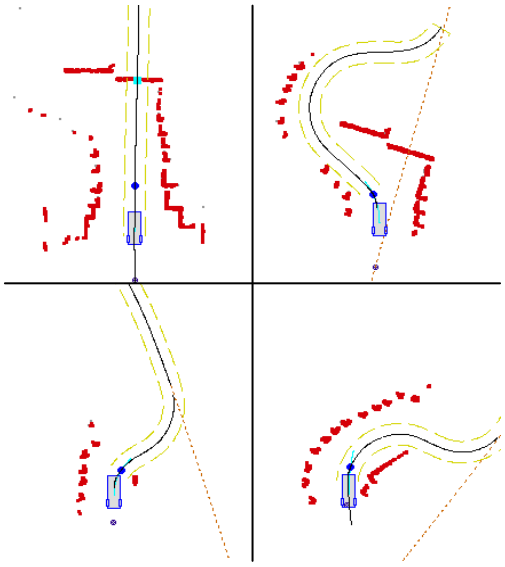

그림 4. 경로 생성 시뮬레이션 결과. 좌측 상단 이미지를 기준으로 시계 방향 순서. 점선은 초기에 주어진 경로이며 실선이 모션 플래너로 계산된 경로이다.

모션 플래너 프레임웍으로 OMPL[8]이 사용되었으며, 인접 이웃 검색 알고리즘으로는 GNAT[9] 이 사용되었다.

5. 결론 및 향후 연구

본 연구에서는 간략화된 Dubins car 모델과 선분과 원호만으로 구성된 경로를 생성하는 Dubins path 에 기반한 모션 플래닝 방법을 보였다. 하지만 실제 주행하는 차량의 경우 경로가 단순히 짧은 것을 떠나, 그 구간을 어느 속도로 주행하는지에 따라 차량이 시작 상태에서 종료 상태에 도달하는 비용이 결정된다. 이러한 점을 고려할 경우 차량의 상태의 차원은 더욱 올라가고 커브 구간의 경우 속도 이외에도 타이어와 지면의 마찰, 다운 포스(Down force) 등의 계산을 통해 차량 안전성을 확보해야만 실질적인 최적 경로, 속도 제어 결과를 얻을 수 있다. 본 연구의 결과가 보인 것은 저속 차량에 대한 계산 비용이 낮은 모션 플래너로서 의미를 가지며, 향후에는 차량 동역학이 가미된 모션 플래닝 연구와 알고리즘 가속을 위한 공간 분석에 대해 연구할 계획이다.

감사의 글

본 연구와 관련하여 많은 영향을 준 한국과학 기술원 무인 자동차 팀 소속의 교수님들과 애써주신 학생 분들께 감사하다는 말을 전합니다. 또한, 연구 진행에 도움을 주신대용량 그래픽스 연구실 동료들에게도 감사의 뜻을 표합니다.

참고문헌

[1] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. P. How,“Motion planning in complex environments using closed-loop prediction,”presented at the AIAA Guid., Nav., Control Conf. Exhibit, Honolulu, HI, Aug. 2008, AIAA-2008-7166.

[2] D. Ferguson, T. M. Howard, and M. Likhachev. Motion planning in urban environments: Part ii. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1070–1076, 2008

[3] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, “Anytime Dynamic A*: An Anytime, Replanning Algorithm” in Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), 2005.

[4] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State Univ. Oct. 1998.

[5] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” in Proc. Robotics: Science and Systems(RSS), 2010.

[6] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed and terminal position tangents." American J. of Math., no. 79, pp. 497-516, 1957.

[7] S. Karaman, E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. IEEE Conf. on Decision and Control, 2010.

[8] Ioan A. Şucan, Mark Moll, Lydia E. Kavraki, The Open Motion Planning Library, IEEE Robotics & Automation Magazine, December 2012

[9] BRIN, S. 1995. Near neighbor search in large metric spaces. In Proceedings of the 21st Conferenceon Very Large Databases (VLDB'95), 574–584.