ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 23-01-01
j.inst.Korean.electr.electron.eng.Vol.27,No.1,1~11,March 2023
https://doi.org/10.7471/ikeee.2023.27.1.1

자율이동체의 주행 시험을 위한 선분과 원호로 이루어진 경로 자동 생성 방법

A method for automatically generating a route consisting of line segments and arcs for autonomous vehicle driving test

조 세 형*★

Se-Hyoung Cho*★

Abstract

Path driving tests are necessary for the development of self-driving cars or robots. These tests are being conducted in simulation as well as real environments. In particular, for development using reinforcement learning and deep learning, development through simulators is also being carried out when data of various environments are needed. To this end, it is necessary to utilize not only manually designed paths but also various randomly and automatically designed paths. This test site design can be used for actual construction and manufacturing. In this paper, we introduce a method for randomly generating a driving test path consisting of a combination of arcs and segments. This consists of a method of determining whether there is a collision by obtaining the distance between an arc and a line segment, and an algorithm that deletes part of the path and recreates an appropriate path if it is impossible to continue the path.

요 약

자율주행 자동차 또는 자율주행 로봇의 개발을 위해서는 경로 주행 시험이 필요하다. 이러한 시험은 실제 환경뿐만 아니라 시뮬레이션 환경에서도 수행되고 있다. 특히 강화학습과 딥러닝을 이용한 개발을 위해서 다양한 환경의 데이터가 필요한 경우에 시뮬레이터를 통한 개발도 이루어지고 있다. 이를 위해서는 수작업으로 설계된 경로뿐만 아니라 무작위로 자동으로 설계된 다양한 경로의 활용이 필요하다. 이러한 시험장 설계는 실제 건설, 제작에도 활용할 수 있다. 본 논문에서는 원호와 선분의 조합으로 이루어진 주행 시험 경로를 무작위로 생성하는 방법을 소개한다. 이는 원호와 선분의 거리를 구하여 충돌 여부를 판별하는 방법과 경로를 계속해서 이어 나가는 것이 불가능할 경우 경로 일부를 삭제하고 적절한 경로를 다시 만들어 나가는 알고리듬으로 이루어진다.

Key words : Dubins path, distance calculation, collision detection, path planning, mobile robot, circular arc

Ⅰ. 서론

로봇, 자동차와 같은 자율 주행 시스템은 지난 몇 년간 큰 관심을 받아 수십억 달러 규모로 시장에 큰 영향을 미칠 것으로 전망된다. 그리하여 supervised learning에 의존하는 시스템이 연구되었다. [1]은 단일 전면 카메라의 원시 픽셀을 조향 명령에 직접 매핑하도록 CNN(컨볼루션 신경망)을 훈련하였고, [2]는 대규모 크라우드 소

* Division of Information and Communications Engineering, Sunmoon University
★ Corresponding author
E-mail : chosh@sunmoon.ac.kr, Tel : +82-41-530-2316
Manuscript received Nov. 28, 2022; revised Dec. 10, 2022; accepted Jan. 9, 2023.

싱 비디오 데이터를 이용한 일반적인 차량 동작 모델 학습 구조를 제안하였다. 그러나 supervised learning 기술의 한계로 완전 자율 자동차의 개발은 지연되고 있다. Deep reinforcement learning 접근 방식은 에이전트가 환경과 상호 작용하고 실수로부터 학습하도록 하므로 자율 주행 차량에 필요한 것과 같은 복잡한 제어 작업을 해결할 가능성이 있다. 그런데 reinforcement learning은 실내, 도심, 레이싱 트랙과 같은 실제 환경에서 만날 수 있는 다양한 상황에 대한 시험과 실패 경험이 필수적이므로 먼저 시뮬레이션 환경에서 학습한 후 실제 환경에 적용된다. 주행 시뮬레이션에 많이 사용되는 도구로는 CARLA와 AirSim이 있는데 도시 레이아웃, 건물, 차량을 제공하며 센서의 사양, 환경 조건, 구동부 제어, 맵 작성 등을 지원한다. AirSim은 AI 연구를 위한 플랫폼으로 드론, 지상 차량 등을 위한 시뮬레이터이다. 자율 차량용 딥 러닝, 컴퓨터 비전 및 강화 학습 알고리즘을 실험하는 데 사용할 수 있다. 도로 레이아웃으로는 20개의 도시 블록과 API가 있는 약 12km의 도로를 제공하지만, 자동 생성 기능은 없다[3]-[5]. 또 다른 시뮬레이션 플랫폼인 CARLA를 이용한 연구에서도 실세계를 모사한 가상환경에서 딥러닝 기반 자율주행을 위한 다양한 학습데이터를 생성하고 검증하는 방안을 제시함으로써, 기존의 실세계에서 다양한 자율주행 학습데이터를 구축하는 데 요구됐던 시공간적 제약과 비용 문제를 해결할 가능성을 보여주었다. CARLA는 도로 레이아웃 생성에 RoadRunner와 같은 호환 설계 도구를 사용할 수 있으나 사용자가 직접 디자인하여야 하며 자동 생성은 지원하지 않고 관련 연구는 도로 레이아웃 생성이 아니라 이미 있는 레이아웃에 경로와 이벤트 발생 시나리오를 설정하는데 국한된다[6]. 그 외의 다른 도구를 사용한 예에서는 심층 강화 학습 접근 방식으로 자율 주행 차량이 도심이 아닌 레이싱 트랙 환경과 상호 작용하고 실수로부터 학습하도록 한다[7][8]. 여기에서도 사용자가 직접 디자인한 도로 레이아웃이 사용되었다. Deep Q-Network와 같은 RL 기반의 자율 주행 시스템 개발에서 일반화 성능을 높이기 위해서는 다양한 형상의 도로 레이아웃을 자동 생성하는 알고리듬의 개발이 요구되나 관련 연구가 현재에는 확인되지 않는다. 따라서 본 논문에서는 무작위 형상의 도로 레이아웃을 자동 생성하는 알고리듬을 소개한다.

차량, 바퀴 달린 로봇, 비행기, 배 등의 이동체는 종종 최소 선회 반경을 가지므로 이들의 최적 이동 경로 계획에는 최대곡률의 원호와 직선을 연결한 최단 경로인 Dubins path를 활용한다[9]-[15]. Dubins path는 원호와 접선분으로 이루어져 있어서 이동체의 움직임 경로를 표현하기에 적합하다. 이들의 경로 계산에는 기하학적 방법[16]과 분석적 방법[17]이 사용된다. Dubins path에 대한 연구는 장애물 회피와 목적지 도착을 위한 경로 생성에 집중되어있다[18]-[25]. Dubins path는 장애물을 중심으로 하는 원의 호를 따라서 둘러 지나가는 경로를 생성하는데도 계산적으로 유리하며 주행로를 설계하고 건설하는 데도 유용하다. 따라서 원호와 선분을 조합하여 도로 레이아웃을 생성하는 방법의 효용성이 높을 것으로 보이며 이는 후속 연구를 통해서 계속 소개하고자 한다.

2장에서는 원호와 선분으로 이루어진 경로의 거리 계산, 충돌 검사 방법을 설명한다. 3장에서는 원호와 선분으로 이루어진 경로를 생성하는 알고리듬과 경로 생성 결과를 소개한다.

II. 원호와 선분으로 이루어진 경로의 충돌검사를 위한 거리 계산

1. 점과 선분, 원호 사이의 거리

가. 점과 선분 사이의 거리

점과 직선 사이의 거리는 점으로부터 직선에 내린 수선의 발까지의 거리이다. 수선의 발이 선분 위에 있으면

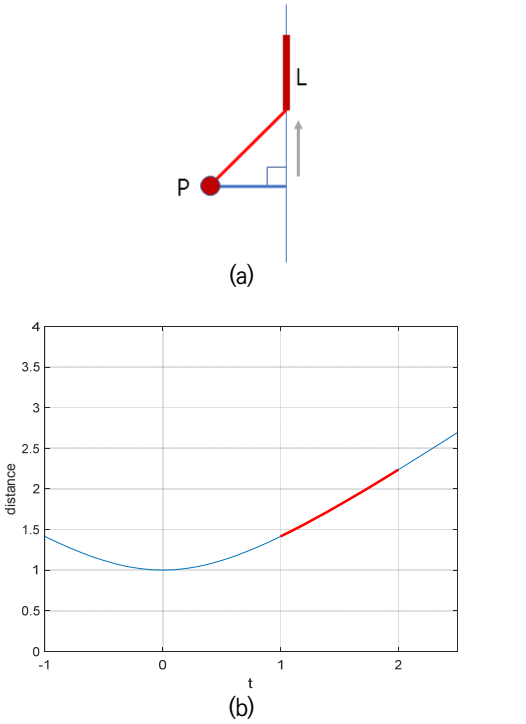


Fig. 1. Distance between a point and a line segment.
그림 1. 점과 선분 사이의 거리

점과 선분 사이의 거리는 점과 직선 사이의 거리와 같게 된다. 수식 1은 점 P와 매개변수방정식으로 표현된 직선 L 그리고 그 사이의 거리 d를 보여준다. 이때 직선의 매개변수의 범위를 제한함으로써 선분이 표현된다. 그림 1(a)에서는 점에서 직선에 내린 수선의 발이 선분 위에 있지 않으므로 매개변수 t 범위의 두 끝 1과 2 중 수선의 발에서 가까운 1을 취하면 선분 L 위의 점 중 점 P와 최단 거리를 이루는 점을 구할 수 있다. 그림 1(b)는 직선의 매개변수 t에 따른 거리 변화를 보여준다. t=0일 때 점 P와 최단 거리를 이루며 t가 이로부터 멀어질 때는 거리가 계속 증가하므로 t=1인 끝점이 점 P와의 거리를 최소로 하는 것을 확인할 수 있다.

$$P = (0, 0) \tag{1}$$
$$Q(t) = (1, t), (1 \le t \le 2)$$
$$d(t) = \sqrt{1^2 + t^2}$$

나. 점과 원호 사이의 거리

점과 원 중심 사이의 거리가 d이고 원의 반지름이 r일 때, 사이의 거리는 |d-r|로 구해진다. 점과 원 중심을 잇는 선분이 원호를 지나면 점과 원호 사이의 거리는 점과 원 사이의 거리와 같게 된다.

수식 2는 점 P와 매개변수방정식으로 표현된 원 A, 그리고 그 사이의 거리 d를 보여준다. 이때 원의 매개변수의 범위를 제한함으로써 원호가 표현된다. 그림 2(a)에서는 점과 원을 최단 거리로 있는 선분이 원호 위에 있지 않으므로 매개변수 범위의 두 끝 45°와 335° 중 원의 중심에서 바라본 각변위가 가까운 335°를 취하면 원호 A 위의 점 중 점 P와 최단 거리를 이루는 점을 구할 수 있다. 그림 2(b)는 원의 매개변수 φ에 따른 거리 변화를 보여준다. φ=0°일 때 점 P와 최단 거리를 이루며 φ가 ±180°까지는 멀어지는 동안 거리가 계속 증가하므로 φ=-25°=335°인 끝점이 점 P와의 거리를 최소로 하는 것을 확인할 수 있다.

$$P = (3, 0) \tag{2}$$
$$Q(\phi) = (2\cos\phi, 2\sin\phi), (45^\circ \le \phi \le 335^\circ)$$
$$d(\phi) = \sqrt{(2\cos\phi - 3)^2 + (2\sin\phi)^2}$$

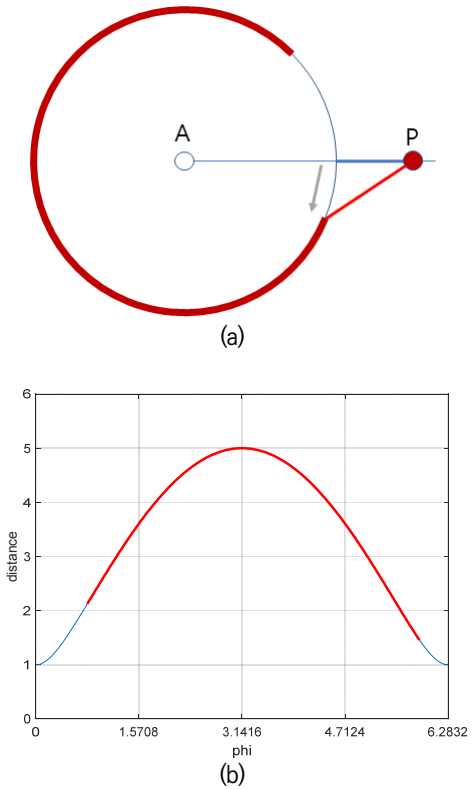


Fig. 2. Distance between a point and an arc.
그림 2. 점과 원호 사이의 거리

다. 선분과 선분 사이의 거리

선분과 선분 사이의 거리는 그의 연장선인 직선들 사이의 거리와 다를 수 있다. 직선들 사이의 최단 거리 점, 다시 말해서 두 직선을 최단 거리로 잇는 선분의 양 끝점이 직선 위에 놓인 선분의 범위에서 벗어나 있는 경우에 그렇다.

수식 3은 매개변수방정식으로 표현된 두 직선 L_1, L_2와 그 사이의 거리 d를 보여준다. 이때 직선의 매개변수 t_1, t_2의 범위를 정함으로써 선분이 표현된다. 선분과 선분 간의 거리는 두 매개변수 t_1, t_2의 함수이다. 두 직선 사이의 최단 거리 점은 기하학적 방법으로도 쉽게 구할 수 있다. 그림 3(a)에서는 두 직선 사이의 최단 거리 점인 교점을 보여준다. 교점이 선분 위에 있지 않으므로 매개변수 t_1과 t_2의 주어진 범위 안에서 최단 거리를 이루는 두 점을 다시 찾아야 한다. 그림 3(e)는 직선의 매개변수 t_1과 t_2에 따른 두 점 사이의 거리를 보여준다. t_1=0, t_2=0일 때 전역 최솟값을 가지며 어느 방향으로든 이로부터 멀어질 때는 계속해서 증가한다. 색깔로 표시된 부분은 매개변수의 범위와 경계를 보여준다. 매개변수의 허용된 영역이 전역 최솟값을 포함하지 않으면 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 3 (e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 3 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면 t_1=1, t_2=1일 때 거리가 최

소가 되는 것을 확인할 수 있다.

$$L_1(t_1) = (t_1, 0), (1 \le t_1 \le 2) \quad (3)$$

$$L_2(t_2) = (0, t_2), (1 \le t_2 \le 2)$$

$$d(t_1, t_2) = \sqrt{t_1^2 + (-t_2)^2}$$

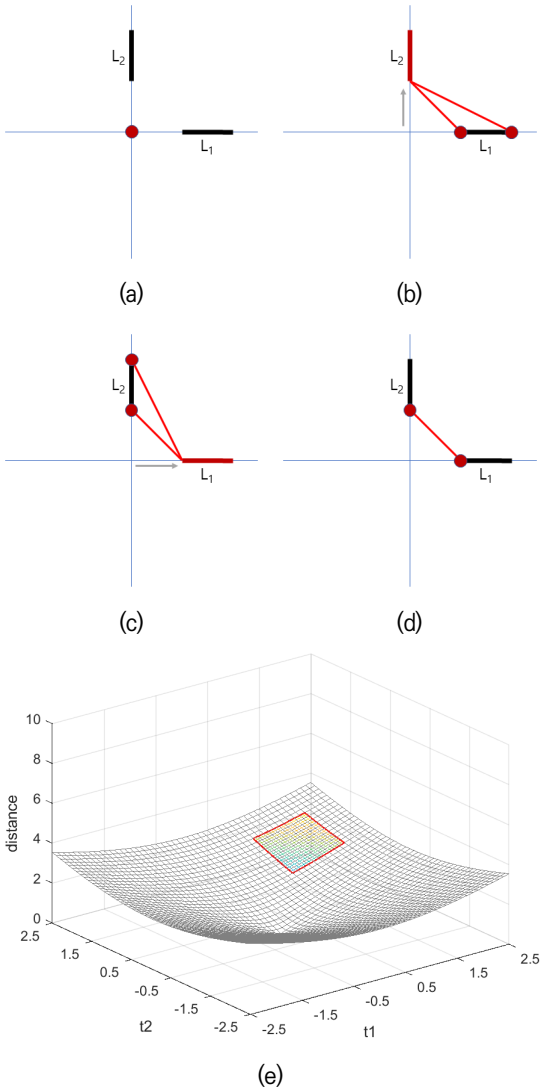


Fig. 3. Distance between the segments on two straight lines that meet each other.

그림 3. 서로 만나는 두 직선 위의 선분 사이의 거리

그림 4는 서로 평행한 선분 사이의 거리를 구하는 과정을 보여준다. 이 경우 전역 최솟값은 직선 위에 수없이 많은 값이 존재한다. 그림 4(a)는 그중 하나를 보여주며 기하학적 방법으로 쉽게 구할 수 있다. 매개변수의 허용된 영역이 전역 최솟값을 포함하지 않으므로 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 4 (e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 4 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면 t_1=0, t_2=0.5일 때 거리가 최소가 되는 것을 확인할 수 있다.

$$L_1(t_1) = (0, t_1), (-1 \le t_1 \le 0) \quad (4)$$

$$L_2(t_2) = (1, t_2), (0.5 \le t_2 \le 1.5)$$

$$d(t_1, t_2) = \sqrt{(-1)^2 + (t_1 - t_2)^2}$$

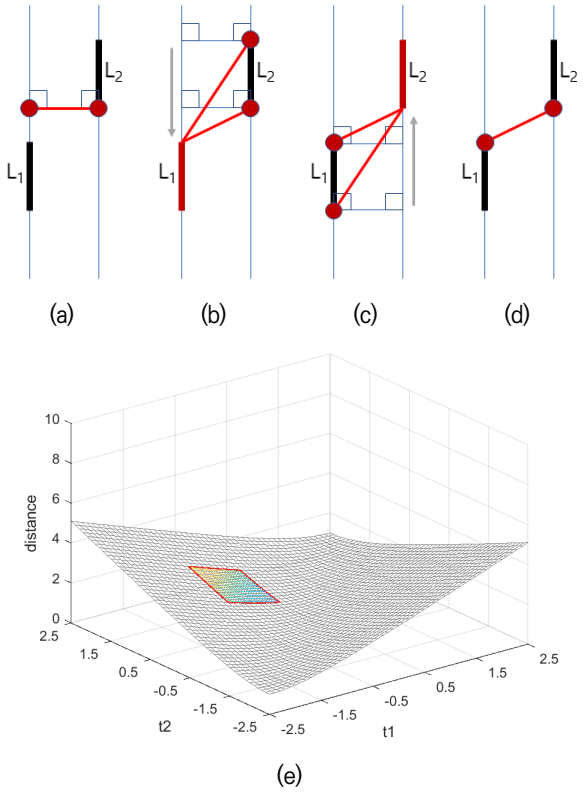


Fig. 4. Distance between the segments on two straight lines that are parallel to each other.

그림 4. 서로 평행한 두 직선 위의 선분 사이의 거리

선분 사이의 거리를 구할 때는 네 개의 경계를 모두 확인할 필요는 없다. 거리가 어느 방향으로든 전역 최솟값에서 멀어질수록 증가하므로 전역 최솟값에서 보이는, 가까운 두 경계만 확인하면 된다. 따라서 앞서 설명한 방법보다 더 효율적인 방법이 존재한다[26]. 이 방법은 parameter 공간에서만 설명되어있으므로 이해도를 높이기 위해서 기하학적으로 표현하면 그림 5와 같다. 가장 많은 경우를 고려해야 하는 최악의 경우를 보여준다.

그림 5(a)와 같이 먼저 직선의 최단 거리 점을 구한다. 두 직선이 만나는 경우에는 교점을 구한다. 두 직선이 꼬인 위치에 있어서 만나지 않는 경우에는 두 직선에 동시에 수직인 직선과의 교점을 구한다. 그다음 그림 5(b)와

같이 선분 하나를 골라잡아서 가까운 끝점을 취하고 다른 직선에 내린 수선의 발을 구한다. 그다음 그림 5(c)와 같이 다른 선분에 대해서 가까운 끝점을 취하고 반대편 직선에 내린 수선의 발을 구한다. 마지막으로 그림 5(d)와 같이 수선의 발에서 가까운 끝점을 취하여 거리를 구한다. 도중에 두 점이 모두 선분 위에 있게 되는 경우에는 바로 거리를 구하면 된다.

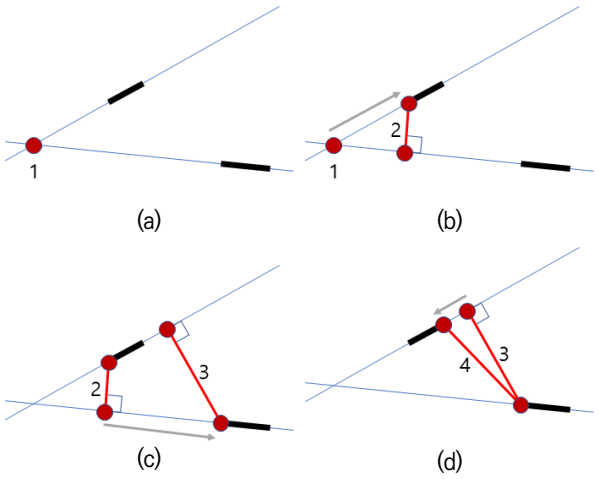


Fig. 5. Distance between the line segments on two straight lines.

그림 5. 두 직선 위의 선분 사이의 거리

다시 언급하자면 이 방법[26]은 더 효율적이지만 선분과 선분 사이의 거리를 구할 때만 활용할 수 있으므로 본 논문에서는 원호와의 거리도 고려할 수 있는 일반화된 방법을 소개한다.

라. 선분과 원호 사이의 거리

두 직선 위에 놓인 두 점 사이의 거리는 최단 거리 점에서 멀어질수록 증가한다. 따라서 그림 5와 같이 최단 거리 점을 선분의 끝점 중 가까운 것과 비교하여 유효한 점을 찾다 보면 선분 사이의 최단 거리 점을 구할 수 있다. 반면 원주상에 놓인 점과 선분 사이의 거리 표현에는 삼각함수가 포함되며 일정 범위 안에서 가까워졌다가 멀어지기를 반복하므로 원주상의 각변위가 최단 거리 점에서 멀어지더라도 최단 거리는 오히려 더 짧아지는 경우가 존재한다.

수식 5는 매개변수방정식으로 표현된 원 A와 직선 L, 그 사이의 거리 d를 보여준다. 이때 직선의 매개변수 φ, t의 범위를 정함으로써 원호와 선분이 표현된다. 원호와 선분 간의 거리는 두 매개변수 φ, t의 함수이다. 원과 직선 사이의 최단 거리 점은 기하학적 방법으로 쉽게 구할

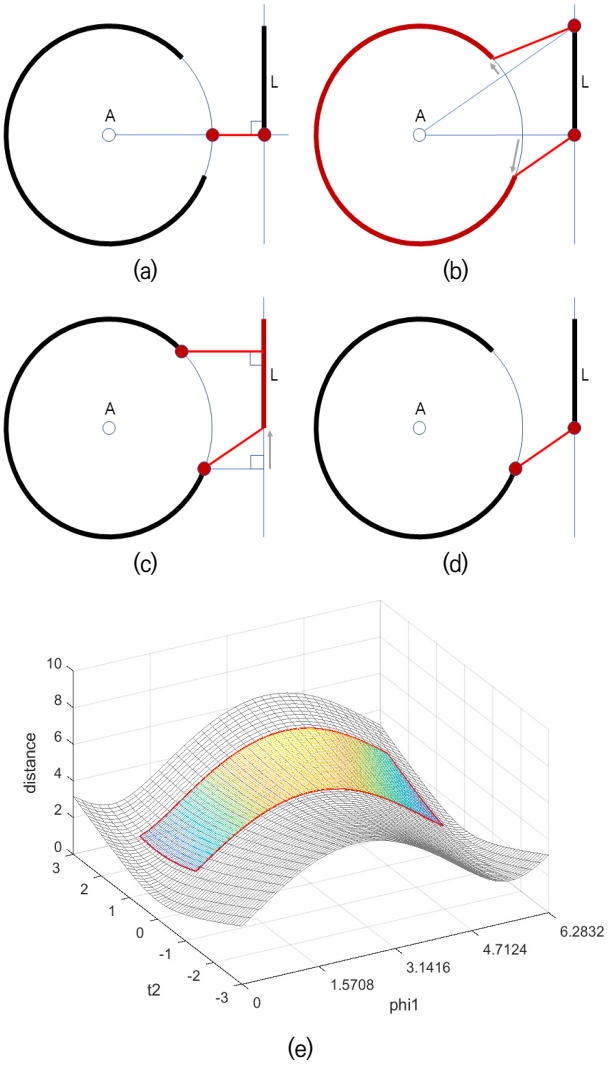


Fig. 6. Distance between a line segment and an arc of a straight line and a circle that are separated from each other.

그림 6. 서로 떨어져 있는 직선과 원의 일부인 선분과 원호 사이 거리

수 있다. 그림 6(a)에서는 원과 직선 사이의 최단 거리 점을 보여준다. 이 점이 원호 위에 있지 않으므로 매개변수 φ, t의 주어진 범위 안에서 최단 거리를 이루는 두 점을 다시 찾아야 한다. 그림 6(e)는 매개변수 φ, t에 따른 두 점 사이의 거리를 보여준다. φ=0°, t=0일 때 전역 최솟값을 가지며 직선을 따라서는 어느 방향으로 멀어지든 계속해서 증가한다. 반면 원주를 따라서는 증가와 감소가 반복된다. 색깔로 표시된 부분은 매개변수의 범위와 경계를 보여준다. 매개변수의 허용된 영역이 전역 최솟값을 포함하지 않으면 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 6 (e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 6 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면

φ=335°, t=0일 때 거리가 최소가 되는 것을 확인할 수 있다. 원호와의 거리를 구할 때는 선분 간의 거리를 구할 때와는 다르게 직사각형 모양으로 이루어진 네 경계를 모두 확인해야 한다.

$$A(\phi) = (2\cos\phi, 2\sin\phi), \quad (45^\circ \le \phi \le 335^\circ) \tag{5}$$

$$L(t) = (3, t), \quad (0 \le t \le 2)$$

$$d(\phi,t) = \sqrt{(2\cos\phi - 3)^2 + (2\sin\phi - t)^2}$$

수식 6은 매개변수방정식으로 표현된 원 A와 직선 L, 그 사이의 거리 d를 보여준다. 원과 직선 사이의 최단 거리 점은 기하학적 방법으로 쉽게 구할 수 있다. 그림 7(a)에서는 원과 직선 사이의 최단 거리 점을 보여준다. 이 경우에는 두 개가 존재한다. 이 점이 선분 위에 있지 않으므로 매개변수 φ, t의 주어진 범위 안에서 최단 거리를 이루는 두 점을 다시 찾아야 한다. 그림 7(e)는 매개변수 φ, t에 따른 두 점 사이의 거리를 보여준다. 이 경우에는 두 개의 같은 크기의 전역 최솟값을 가지며 직선을 따라서는 어느 방향으로 멀어지든 계속해서 증가한다. 반면 원주를 따라서는 증가와 감소가 반복된다. 색깔로 표시된 부분은 매개변수의 범위와 경계를 보여준다. 매개변수의 허용된 영역이 두 전역 최솟값 중 어떤 것도 포함하지 않으면 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 7 (e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 7 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면 φ=315°, t=-1일 때 거리가 최소가 되는 것을 확인할 수 있다.

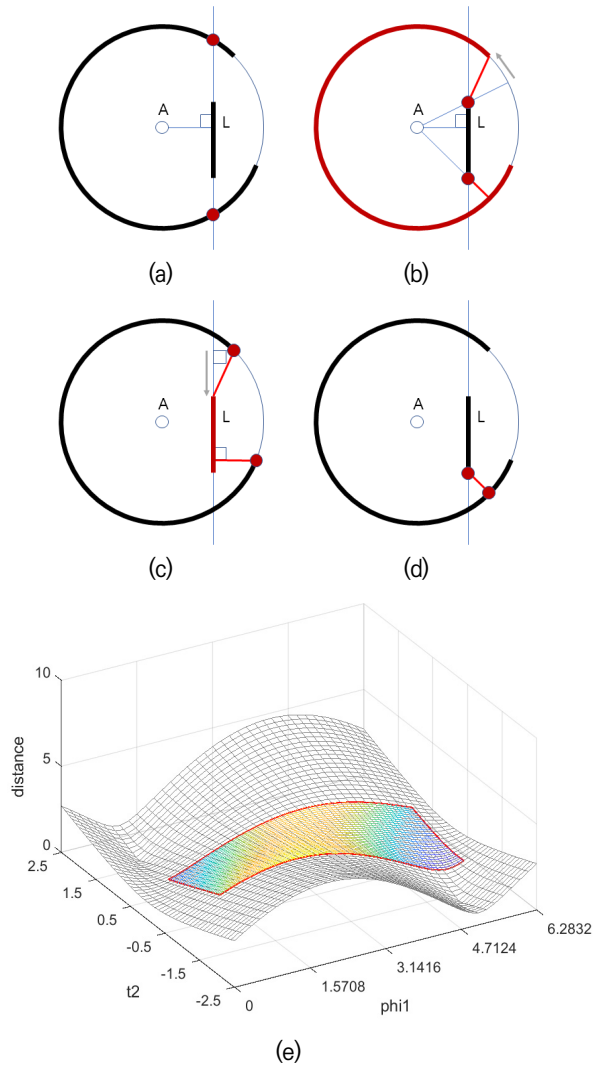


Fig. 7. Distance between the line segment and the arc of a straight line and a circle that meet each other.

그림 7. 서로 만나는 직선과 원의 일부인 선분과 원호 사이 거리

$$A(\phi) = (2\cos\phi, 2\sin\phi), \quad (45^\circ \le \phi \le 335^\circ) \tag{6}$$

$$L(t) = (1, t), \quad (-1 \le t \le 0.5)$$

$$d(\phi,t) = \sqrt{(2\cos\phi - 1)^2 + (2\sin\phi - t)^2}$$

마. 원호와 원호 사이의 거리

수식 7은 매개변수방정식으로 표현된 원 A_1, A_2와 그 사이의 거리 d를 보여준다. 이때 원의 매개변수 φ_1, φ_2의 범위를 정함으로써 원호가 표현된다. 원호 간의 거리는 두 매개변수 φ_1, φ_2의 함수이다. 두 원 사이의 최단 거리 점은 기하학적 방법으로 쉽게 구할 수 있다. 그림 7(a)에서는 두 원 사이의 최단 거리 점을 보여준다. 이 점이 원 A_1의 호 위에 있지 않으므로 매개변수 φ_1, φ_2의 주어진 범위 안에서 최단 거리를 이루는 두 점을 다시 찾아야 한다. 그림 7(e)는 매개변수 φ_1, φ_2에 따른 두 점 사이의 거리를 보여준다. 한 개의 전역 최솟값을 가지며 원주를 따라서 증가와 감소가 반복된다. 색깔로 표시된 부분은 매개변수의 범위와 경계를 보여준다. 매개변수의 허용된 영역이 전역 최솟값을 포함하지 않으면 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 7 (e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 7 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면 거리가 최소가 되는 점을 찾을 수 있다.

$$A_1(\phi_1) = (2\cos\phi_1, 2\sin\phi_1), \quad (45^\circ \le \phi_1 \le 335^\circ) \tag{7}$$

$$A_2(\phi_2) = (5 + 2\cos\phi_2, 2\sin\phi_2), \quad (45^\circ \le \phi_2 \le 335^\circ)$$

$$d(\phi_1,\phi_2) = \sqrt{(2\cos\phi_1 - 5 - 2\cos\phi_2)^2 + (2\sin\phi_1 - 2\sin\phi_2)^2}$$

수식 8은 매개변수방정식으로 표현된 원 A_1, A_2와 그 사이의 거리 d를 보여준다. 그림 9(a)에서는 두 원 사이의 최단 거리 점을 보여준다. 이 경우에는 두 개가 존재한다. 이 점 중 어떤 것도 원 A_1의 호 위에 있지 않으므로 매개변수 φ_1, φ_2의 주어진 범위 안에서 최단 거리를 이루는 두 점을 다시 찾아야 한다. 그림 9(e)는 매개변수

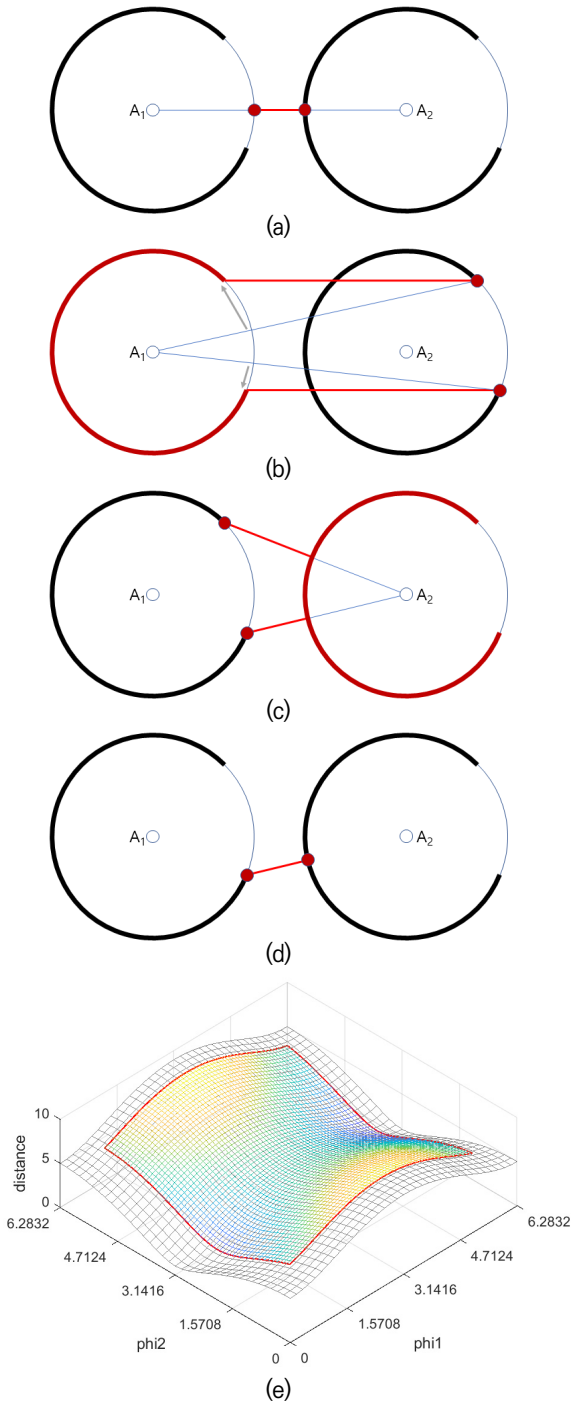


Fig. 8. Distance between arcs of circles that are apart from each other.

그림 8. 서로 떨어져 있는 원의 일부인 원호 사이 거리

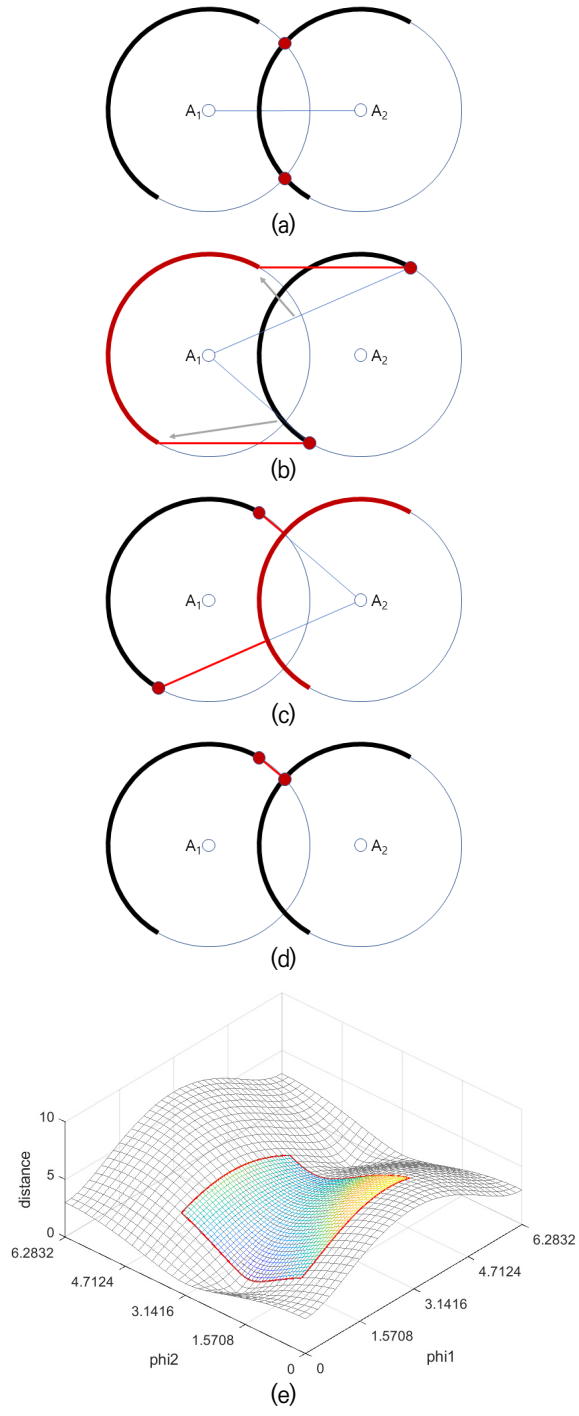


Fig. 9. Distance between arcs of circles that meet each other.

그림 9. 서로 만나는 원의 일부인 원호 사이 거리

φ_1, φ_2에 따른 두 점 사이의 거리를 보여준다. 두 개의 전역 최솟값을 가지며 원주를 따라서 증가와 감소가 반복된다. 색깔로 표시된 부분은 매개변수의 범위와 경계를 보여준다. 매개변수의 허용된 영역이 두 개의 전역 최솟값 중 어떤 것도 포함하지 않으면 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 9 (e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 9 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면 거리가 최소가 되는 점을 찾을 수 있다.

$$A_1(\phi_1) = (2\cos\phi_1, 2\sin\phi_1), \quad (60^\circ \le \phi_1 \le 240^\circ) \tag{8}$$

$$A_2(\phi_2) = (3+2\cos\phi_2, 2\sin\phi_2), \quad (60^\circ \le \phi_2 \le 240^\circ)$$

$$d(\phi_1,\phi_2) = \sqrt{(2\cos\phi_1 - 3 - 2\cos\phi_2)^2 + (2\sin\phi_1 - 2\sin\phi_2)^2}$$

수식 9는 매개변수방정식으로 표현된 원 A_1, A_2와 그 사이의 거리 d를 보여준다. 두 원은 동심원이다. 두 원 사이의 최단 거리를 잇는 선분은 무수히 많이 존재하며 그림 10(a)에서는 그중 하나를 보여준다. 이들 중 어떤 것도 원 A_1과 A2의 호 위에 동시에 있지 않으므로 매개변수 φ_1, φ_2의 주어진 범위 안에서 최단 거리를 이루는 두 점을 다시 찾아야 한다. 그림 10(e)는 매개변수 φ_1, φ_2에 따른 두 점 사이의 거리를 보여준다. 무수히 많은 전역 최솟값을 가지며 원주를 따라서 증가와 감소가 반복된다. 색깔로 표시된 부분은 매개변수의 범위와 경계를 보여준다. 매개변수의 허용된 영역이 전역 최솟값 중 어떤 것도 포함하지 않으면 영역의 최솟값은 영역의 경계 중에 존재한다. 그림 10(e)의 직사각형 모양으로 이루어진 네 경계에서 각각 최솟값을 찾는 과정을 그림 10 (b)(c)에서 보여준다. 네 개의 최솟값 중 가장 작은 값을 취하면 거리가 최소가 되는 점을 찾을 수 있다.

$$A_1(\phi_1) = (2\cos\phi_1, 2\sin\phi_1), \quad (60^\circ \le \phi_1 \le 240^\circ) \tag{9}$$

$$A_2(\phi_2) = (\cos\phi_2, \sin\phi_2), \quad (270^\circ \le \phi_2 \le 360^\circ)$$

$$d(\phi_1,\phi_2) = \sqrt{(2\cos\phi_1 - \cos\phi_2)^2 + (2\sin\phi_1 - \sin\phi_2)^2}$$

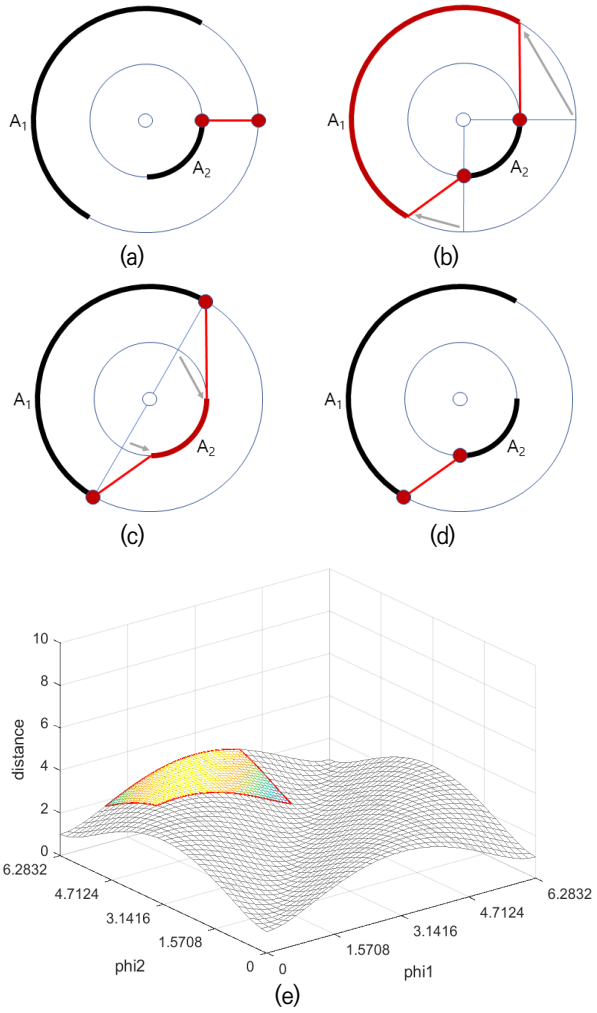


Fig. 10. Distance between arcs that are part of concentric circles.

그림 10. 동심원의 일부인 원호 사이 거리

Ⅲ. 원호와 선분으로 이루어진 경로 생성

그림 11은 원호와 선분으로 이루어진 경로를 보여주며 Algorithm 1은 이러한 경로의 생성 알고리듬을 보여준다. 기존 경로에 우회전, 좌회전, 직진 중에서 임의의 구간을 덧붙여 생성한 후 새로 추가된 구간을 기존에 생성된 경로와 충돌검사 한다. 최근에 생성한 구간이 그림 12와 같이 충돌 경로이면 삭제하고 이전 구간에 penalty를 부여한다. 그리고 어떤 구간에 penalty가 일정 횟수 이상 누적되면 삭제하고 이전 구간에 penalty를 부여한다. 이렇게 하면 경로 생성 중에 구간을 추가로 이어 나가기 어려운 형상이 되었을 때 이전 형상으로 되돌아가서 다른 경로를 다시 생성하도록 시도할 수 있다.

Table 1. Algorithm for path generation consisting of circular arcs and line segments.
표 1. 원호와 선분으로 이루어진 경로 생성 알고리즘

Algorithm 1 Generate_Path

Input: boarder_line, target_path_length
Output: path
Initialization:
1: i=0
LOOP Process
2: **while** path_length 〈 target_path_length
3: path[i] = Generate_Random_Section()
4: **if** Collision_Detected(path, i) == false
5: fail_count[i] = 0
6: **else**
7: fail_count[i] = ∞
8: **while** fail_count[i] 〉 5
9: i=i-1
10: fail_count[i] = fail_count[i]+1
11: **end while**
12: i=i+1
13: **end while**
14: **return** path

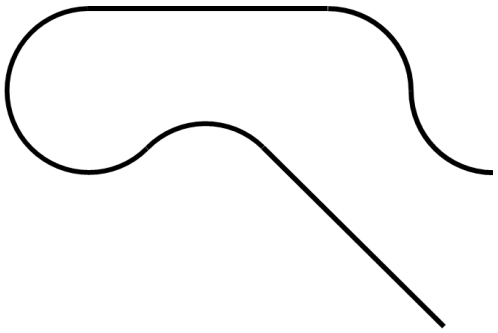


Fig. 11. Path consisting of a line segment and an arc.
그림 11. 선분과 원호로 이루어지는 경로

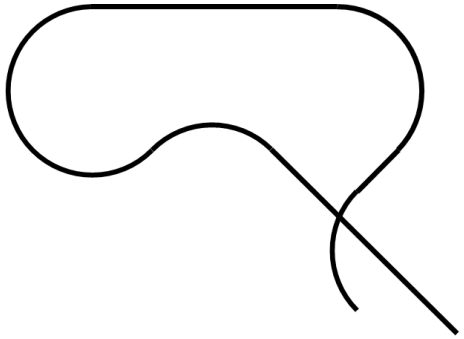


Fig. 12. Collision of a path consisting of a line segment and an arc.
그림 12. 선분과 원호로 이루어지는 경로의 충돌

그림 13은 직선 간에 수직 통과를 허용하여 교차로와 함께 생성한 주행 경로이며 로봇 시뮬레이션 모델의 주행 시험과 강화학습을 위한 에피소드 데이터 획득에 사용되었다.

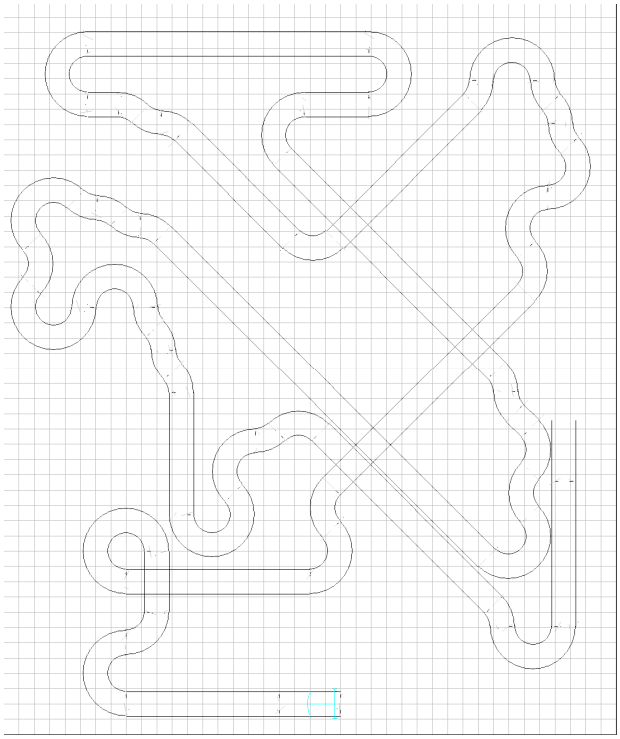


Fig. 13. Driving test path created using line segments and arcs.
그림 13. 선분과 원호를 이용해서 생성한 주행 시험 경로

Ⅳ. 결론

Deep Q-Network와 같은 reinforcement learning 기반의 자율 주행 시스템 개발에서 일반화 성능을 높이기 위해서는 다양한 형상의 도로 레이아웃을 자동 생성하는 알고리듬의 개발이 요구되나 관련 연구가 현재에는 확인되지 않는다. 따라서 본 논문에서는 무작위 형상의 도로 레이아웃을 자동 생성하는 알고리듬을 소개하였다.

먼저 선분과 원호로 이루어진 경로 세그먼트의 충돌검사를 위한 거리 계산 방법을 소개하였다. 선분-선분, 선분-원호, 원호-원호 사이의 거리 계산을 위하여 기하학적 방법으로 전역 최솟값을 찾고 매개변수공간을 검사하여 지역 최솟값을 찾는 방법을 제시하였다.

다음으로 선분과 원호를 무작위로 이어서 경로를 생성하고, 충돌 때문에 경로를 계속해서 이어 나가는 것이 불가능할 경우 문제가 되는 경로 일부를 삭제하고 후속 경로를 다시 만들어 나가는 알고리듬을 제시하였다. 그리고 제안한 방법으로 주행 시험할 수 있는 경로를 생성할 수 있음을 확인하였다.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao and K. Zieba, "End to End Learning for Self-Driving Cars," 2016.
DOI: 10.48550/arXiv.1604.07316

[2] H. Xu, Y. Gao, F. Yu and T. Darrell, "End-to-end Learning of Driving Models from Large-scale Video Datasets," *2017 IEEE Conference on Computer Vision and Pattern Recognition,* 2017.
DOI: 10.48550/arXiv.1612.01079

[3] S. Shah, S. Dey, C. Lovett and A, Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: Hutter, M., Siegwart, R. (eds) Field and Service Robotics," *Springer Proceedings in Advanced Robotics,* 2018.
DOI: 10.48550/arXiv.1705.05065

[4] Kim, Min-Tae and Kim, Byung-Wook, "Autonomous driving technique based on CNN using AirSim," *Proc. Korean Society of Electronics Engineers Conference*, pp.1018-1021, 2020.

[5] Joo, Eun-Oh, Kwag, Ye-Eun, Kim, Min-Soo, "CARLA-based virtual environment training data collection and real-world usability validation," *Proc. KOREA Spatial Information Society*, pp. 141-143, 2022.

[6] Lee, Shinkyung, Sung, Kyungbok and. Min, Kyungwook, "Development of Autonomous Driving Scenario Editor based on Carla," *Proc. Korean Society of Automotive Engineers*, pp.405-406, 2020.

[7] M. Bosello, R. Tse and G. Pau, "Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1tenth LIDAR-Based Races," *Proc. of IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pp.290-298, 2022.
DOI: 10.1109/CCNC49033.2022.9700730

[8] D. Loiacono, A. Prete, P. L. Lanzi and L. Cardamone, "Learning to overtake in TORCS using simple reinforcement learning," *IEEE Congress on Evolutionary Computation*, pp.1-8, 2010.
DOI: 10.1109/CEC.2010.5586191

[9] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. of Mathematics*, vol.145, no.2, pp.367-393, 1990. DOI: 10.2140/pjm.1990.145.367

[10] *"IN MEMORIAM Lester Eli Dubins Professor of Mathematics and Statistics, Emeritus UC Berkeley 1920~2010,"* University of California, 2012.

[11] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American J. of Mathematicsm*, vol.79, no.3, pp.497-516, 1957.
DOI: 10.2307/2372560

[12] Harold H. Johnson, "An application of the maximum principle to the geometry of plane curves," *Proc. American Mathematical Society*, vol.44, no.2, pp.432-435, 1974.
DOI: 10.2307/2040451

[13] J.-D. Boissonat, A. Cerezo and K. Leblond, "Shortest Paths of Bounded Curvature in the Plane," *Proc. IEEE Int. Conf. on Robotics and Automation*, vol.3, pp.2315-2320, 1992.
DOI: 10.1109/ROBOT.1992.220117

[14] Ayala José, Kirszenblat David and Rubinstein Hyam, "A Geometric approach to shortest bounded curvature paths," *J. of Communications in Analysis and Geometry*, vol.26, no.4, pp.679-697, 2018.
DOI: 10.48550/arXiv.1403.4899

[15] Ayala José, "Length minimising bounded curvature paths in homotopy classes," *J. of Topology and Its Applications*, vol.193, pp.140-151, 2015.
DOI: 10.1016/j.topol.2015.06.008

[16] Anisi David, "Optimal Motion Control of a Ground Vehicle," *Swedish Research Defence Agency,* pp.1650-1942, 2003.

[17] Xuan-Nam Bui, J.-D. Boissonnat, P. Soueres and J.-P. Laumond, "Shortest Path Synthesis for Dubins Non-Holonomic Robot," *Conf. IEEE Robotics and Automation*, vol.1, pp.2-7, 1994.
DOI: 10.1109/ROBOT.1994.351019

[18] Satyanarayana Manyam and Sivakumar Rathinam, "On Tightly Bounding the Dubins

Traveling Salesman's Optimum," *J. of Dynamic Systems, Measurement, and Control*, vol.140, no.7, 2016. DOI: 10.1115/1.4039099

[19] Satyanarayana G. Manyam, Sivakumar Rathinam, David Casbeer and Eloy Garcia, "Tightly Bounding the Shortest Dubins Paths Through a Sequence of Points," *J. of Intelligent & Robotic Systems*, vol.88, no.2-4, pp.495-511, 2017.
DOI: 10.1007/s10846-016-0459-4

[20] J. -P. Hong, Y. -J. Choi and K. -H. Park, "Path Planning and Mobile Robot Control for the Obstacle Avoidance by Using Dubin's Curve," *roc. Korean Society of Control and Robot Systems Conference*, pp.25-29, 2007.

[21] Dongsin Kim and Keumjin Lee, "Shortest Dubins Path from a Point to a Line," *J. of The Korean Society for Aeronautical and Space Sciences*, pp.631-632, 2019.

[22] You Young Yang and Henzeh Leeghim, "Shortest Path Generation and Tracking Using Dubins Path for Unmanned Vehicles," *J. of The Korean Society for Aeronautical and Space Sciences*, pp.251-252, 2020.

[23] Huiseong Song, Mingu Kim and Youdan Kim, "Rendezvous Path Planning of UAV using Dubins Curve," *J. of The Korean Society for Aeronautical and Space Sciences*, pp.234-235, 2016.

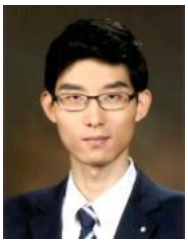
[24] H. -D. Oh, H. -S. Shin and M. -J. Tak, "Integration of Task Assignment and Path Planning for Multi-UAVs Using Dubins Set," *J. of The Korean Society for Aeronautical and Space Sciences*, pp.729-733, 2009.

[25] O, Su-Hun, Ha, Chul-Su, Kang, Seung-Eun, Mok, Ji-hyun, Ko, Sangho and Lee, Yong-Won, "3-Dimensional Path Planning and Guidance using the Dubins Curve for an 3-DOF Point-mass Aircraft Model," *J. of the Korean Society for Aeronautical Science and Flight operation*, vol.24, no.1, pp.1-9, 2016.
DOI: 10.12985/ksaa.2016.24.1.001

[26] Daniel Sunday, *"Practical Geometry Algorithms: With C++ Code,"* 2021.

BIOGRAPHY

Se-Hyoung Cho (Member)



2004 : BS degree in University of Seoul.
2007 : MS degree in Electric and Electronic Engineering, KAIST.
2016 : PhD degree in Robotics program, KAIST.

2017~ : Assistant Professor, Division of Information and Communications Engineering, Sunmoon University.