이화여자대학교 대학원
2019학년도
석사학위 청구논문

동적 환경에서의 충돌 예측 확률을 이용한 지역적 충돌 회피 알고리즘

컴 퓨 터 공 학 과
김 지 선
2020

동적 환경에서의 충돌 예측 확률을 이용한 지역적 충돌 회피 알고리즘

이 논문을 석사학위 논문으로 제출함

2019 년 12월

이화여자대학교 대학원

컴 퓨 터 공 학 과 김 지 선

김 지 선 의 석사학위 논문을 인준함

지도교수 김 영 준 ____

심사위원 박 현 석 ____

오 유 란 ____

김 영 준 ____

이화여자대학교 대학원

목 차

표 목 차

그 림 목 차

논 문 개 요

자율적으로 경로를 계획하면서 주행하는 자율주행 로봇의 핵심 기능은 효율적인 충돌 회피와 경로 생성이다. 지역 경로 계획자(local path planner)는 충돌 회피를 위해 짧은 시간 동안 장애물과의 충돌 없는 경로를 지역적으로 생성하고, 이 과정을 목표 지점에 도달할 때까지 반복한다. 이러한 지역 경로 계획자의 충돌 회피 알고리즘은 동적 환경에서도 효율적으로 작동해야 하며 즉각적인 충돌 회피가 가능해야 한다. 그러나 기존의 지역 경로 계획자는 여전히 정적 장애물만을 고려하거나, 장애물의 정확한 위치 및 속도 정보를 알고 있다는 가정에 따라 로봇의 경로를 계획하므로 실제 로봇에 적용할 때 통신 의존성 등의 한계점을 보인다.

본 논문에서는 충돌 예측 확률에 기반한 동적 환경에서의 장애물 회피 알고리즘을 제안한다. 본 논문에서 제안하는 알고리즘은 효율적인 계산으로 Robot Operating System (ROS)의 지역 경로 계획법에 사용되는 Dynamic Window Approach (DWA)를 기반으로 확률 기반의 동적 장애물 회피 기능을 추가하는 형태로 연구되었다. 제안하는 접근법은 먼저 2차원 LiDAR 센서를 사용하여 주변 장애물의 동적 궤적을 예측하고 그에 따른 충돌 예측 시간을 계산한다. 이를 이용하여 충돌 회피 확률을 계산하고 DWA의 경로 비용함수 최적화에 반영한다. 이로써 로봇이 충돌 가능성이 큰 속도로 주행하는 것을 최소화한다.

제안하는 알고리즘의 성능은 Gazebo 시뮬레이터를 이용하여 동적 장애물이 주

어진 환경에서 터틀봇3의 네비게이션을 이용하여 실험하였다. 기존의 DWA 알고리즘과 본 논문에서 제안하는 알고리즘을 통한 로봇의 충돌 회피 동작, 경로, 그리고 소요 시간을 비교한 결과, 본 논문에서 제안하는 알고리즘이 동적 환경에서 더 유연한 충돌 회피 기동을 보이고 더 효율적인 방법 (거리 및 시간)으로 주행함을 볼 수 있었다.

Ⅰ. 서론

A. 연구 배경

로봇 기술의 대중화에 따라 주로 연구소, 공장 등에만 존재하던 로봇들을 근래에는 더욱 다양한 곳에서 접하게 되었다. 그뿐만 아니라 pick-and-place 매니퓰레이션으로 한정되어 있던 로봇의 주 업무들이 이제는 사용자의 필요에 맞게 다양하게 확장되고 있고, 이에 따라 이동성을 갖춘 이동 로봇의 수요 역시 증가하고 있다. 물류, 창고, 스마트 팩토리 공정에서는 이미 이동 로봇을 활용한 다중 로봇 시스템(multi-robot system)을 구축하고 있으며, 안내 서비스나 배달 서비스 또한 이동성을 갖춘 로봇으로 사용자에게 서비스를 제공한다.

이러한 이동 로봇들은 상당수가 자율 주행 시스템(AMR)을 탑재하고 있다. 목표 지점까지 로봇이 자율적으로 경로를 계획하면서 주행하는 AMR 네비게이션 시스템의 핵심 기능은 효율적인 경로 생성과 충돌 회피이다. 이를 위해 전역 경로 계획자(global planner)는 목표 지점까지의 효율적인 경로를 생성하고[1], 지역 경로 계획자는 짧은 시간 동안의 충돌 없는 지역 경로를 계획한다[2].

지역 경로 계획자는 정적 환경뿐만 아니라 동적 환경에서도 효율적이며 즉각적인 충돌 회피 경로를 생성해야 한다. 그러나 상당수의 지역 경로 계획자는 여전히 정적 장애물만을 고려하며 경로를 계획한다. 또한, 다수의 동적 환경 네비게이션 연구는 로봇이 장애물의 정확한 상태를 알고 있다고 가정하므로, 실제 로봇에 적용할 때 통신 의존성이 높아지는 한계점이 있다. 이러한 한계점을 보완하기 위해 미지의 환경에서도 로봇이 자율적으로 주변의 동적 환경을 예측하여 회피하는 시스템이 필요하다. 본 논문에서는 충돌 예측 확률에 기반한 동적 장애물 회피 알고리즘을 제안한다. 제안하는 알고리즘은 동적 환경에서의 불확실성을 고려하여 로봇의 시간 윈도우 상의 충돌 확률을 계산하고 충돌 회피 경로를 생성한다.

본 논문의 알고리즘은 로봇의 동역학 및 기구학이 잘 구현된 로봇 소프트웨어 플랫폼 ROS에서 개발하고 실험하였다. 특히 ROS의 지역 경로 계획법인 DWA[3]를 보완하여 기존 정적 장애물의 회피 기능뿐만 아니라 동적 장애물의 회피 기능을 갖도록 한다. 본 논문에서는 2D-LiDAR 센서를 사용하여 주변 장애물의 궤적을 예측하고 충돌 예측 시간을 계산한다. 그리고 충돌 예측 시간에 기초한 충돌 회피 확률을 계산하여 이를 DWA의 속도 샘플링 기반 최적화의 목적 함수에 반영한다. Gazebo 시뮬레이터에서 실험한 결과는 본 논문에서 제안하는 알고리즘이 기존의 DWA보다 동적 환경에서 더 적절한 충돌 회피 기동을 보이고, 더 효율적인 경로로 주행이 가능함을 보였다.

B. 연구 목표 및 기여 사항

- 본 논문에서 제안하는 알고리즘은 DWA를 확장하여 동적 장애물에 대한 회피 동작을 설계하는 것을 목표로 한다. 개발 환경을 고려한 본 연구의 하위 목표와 그에 따른 해결 과제는 다음과 같다.

    - ***2D-LiDAR 센싱에 따른 동적 장애물의 경로 추정:*** 2D-LiDAR 센서로 감지한 장애물들을 정적 장애물과 동적 장애물로 분리하고, 서로 다른 동적 장애물로 정확하게 구별하는 과정이 필요하다. 그리고 동적 장애물의 위치, 속도 값이 실제 값과 근사하도록 추정하는 과정이 필요하다.

    - ***동적 장애물의 궤적 예측을 통한 장애물과의 충돌 예측 시간 계산:*** 2차원 평면에서의 장애물과의 충돌 시간을 예측하며, 이는 로봇과 각 장애물의 상대 거리와 상대 속도를 계산하는 과정이 필요하다.

- ***충돌 예측 시간을 활용한 충돌 회피 확률 모델 제시:*** 각 장애물과의 충돌 예측 시간을 활용하여 로봇의 주행 방향 별 충돌 회피 확률을 계산하는 과정이 필요하다.

- ***충돌 회피 확률을 반영한 지역 경로 계획법의 구현:*** 속도 샘플을 평가하는 목적 함수에 충돌 회피 확률을 반영하여 계산할 수 있도록 구현하는 작업이 필요하다.

따라서 본 논문이 기여하는 주요사항은 기존 DWA의 확장 연구들과 달리 장애물의 동작을 예측하고 충돌 확률을 사용한 동적 환경에서의 장애물 회피 및 경로 생성 알고리즘이다.

또한, 본 논문에서는 ROS의 기본 네비게이션 스택을 사용하는 이동 로봇에 광범위하게 적용할 수 있는 지역 충돌 회피 알고리즘을 제시한다.

C. 논문 구성

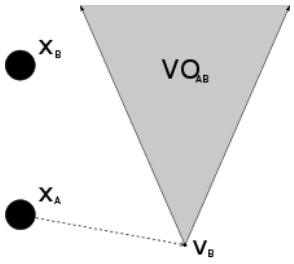
본 논문의 구성은 다음과 같다. 2장에서는 지역 충돌 회피에 관한 선행 연구를 확인한다. 3장에서는 시스템 개요, 그리고 4장에서는 연구 방법을 제시한다. 그리고 5장에서는 유의미한 실험 결과를 확인하고 분석한다. 마지막으로 6장에서는 결론 및 향후 계획에 대해 논의한다.

Ⅱ. 선행 연구

본 장에서는 로봇의 지역적 충돌 회피에 관련한 선행 연구를 알아본다. 이러한 지역적 충돌 회피는 실시간으로 다가올 짧은 시간 단계의 지역적 경로를 계획하고 최적의 속도로 로봇이 주행하게 한다.

A. 충돌 속도 집합을 활용한 연구

지역 충돌 회피 알고리즘에는 Velocity obstacles(VO)[4] 접근법이 있다. VO 접근법은 로봇과 충돌 가능성이 있는 속도의 집합을 구하고 그 속도 집합을 제외한 영역에서 택한 속도로 로봇을 주행하게 한다. [그림 1]과 같이 전방으로 움직이는 홀로노믹(holonomic)한 원형 로봇 A와 장애물 B 각각의 위치와 속도가 주어졌을 때 충돌 가능성이 생기는 A의 속도 집합을 A의 B에 대한 VO 영역으로 정의한다. 로봇은 VO 영역을 제외한 나머지 가능한 속도 집합 영역에서 로봇의 목표 지점을 향하는 속도를 선택한다.



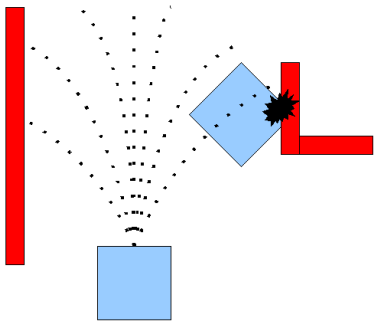
[그림 1] Velocity Obstacles [5]

이를 확장한 연구로는 진동 없는 동작(oscillation-free motion)을 생성하는 Reciprocal Velocity Obstacles (RVO)[6], ORCA[7]가 있다. 또 차동 로봇의 기구학적 한계점을 보완한 DD-ORCA[8], NH-ORCA[9]가 있다. 이 연구들은 다중 로봇 네비게이션(multi-robot navigation)에 적용되며, 모두 다음 시간 단계의 충돌 없는 동작(collision-free motion)을 위한 지역 경로를 계획한다. 그

러나 언급한 연구들은 시뮬레이션 환경상에서 모든 객체가 서로의 정확한 위치와 속도 값을 알고 있는 상태에서 실험한다. 이는 실제 로봇에 적용 시 중앙 서버와의 통신 혹은 로봇 간 통신이 필요하므로 성능 속도가 저하되며 통신 상태에 따라 정보를 유실할 위험성이 있다. 본 논문에서는 이러한 문제점을 줄이고자, 장애물들의 정확한 상태 정보를 별도로 수신하는 과정을 생략하고 센싱을 이용하여 장애물의 상태를 추정하고 계산한다.

B. 최적화 기반 속도 탐색 연구

Dynamic Window Approach(DWA)는 동역학을 고려하여 로봇의 가능한 속도 집합을 dynamic window로 제한한다. 이후 DWA는 원형 호를 사용하여 궤적 후보의 표본을 찾고 샘플 기반 최적화를 진행한다. [그림 2]의 붉은색 다각형은 장애물, 파란색 사각형은 로봇을 나타내며, 원형 호 점선은 표본을 나타낸다. 로봇은 모든 표본에서 정적 장애물과의 거리, 목표 지점과의 근접도, 현재 속도와의 유사도를 고려하는 목적 함수를 계산하고 최댓값을 갖는 최적의 경로를 찾아 주행한다. 이 방법은 최적 시간(time-optimal) 해를 구하기 위해 연속적 궤적 최적화를 진행하는 Time Elastic Band(TEB) 지역 경로 계획법[10]에 비해 더 효율적인 계산이 가능하다. DWA는 동역학을 고려한다는 점과 계산 효율성으로 인하여 ROS 네비게이션 지역 경로 계획[2]에 사용된다.



[그림 2] DWA의 로봇(파란 색 사각형), 장애물(붉은 색 다각형), 속도 표본(원형 호 점선) [2]

DWA를 확장한 연구로 [11]에서는 홀로노믹 모델의 기능을 더하며, 목표 지점까지의 얼라인먼트(alignment) 정보를 반영하는 전역 DWA를 제안한다. 그리고 [12]에서는 모델 예측 제어와 제어 랴푸노프 함수를 적용하여 로봇이 국소 최저치 없이 목표 지점으로 수렴하도록 한다. 그리고 예측 기반의 DWA 연구로, [13]에서는 정적 장애물 실험 환경에서 다가오는 두 단계의 샘플을 예측하고 최적화를 진행하여 경로를 선택한다. 이외에도 [14]에서는 장애물의 동작을 예측하고 충돌 시간을 계산하여 로봇의 경로를 생성하는 예측 기반 DWA를 제안한다. 본 논문에서는 장애물의 동작을 예측하여 확률적으로 충돌을 확인하고 이를 로봇 경로 계획에 반영하는 DWA를 제안한다. 선행 DWA 연구들과 제안하는 충돌 예측 확률 기반 DWA의 비교 사항은 [표 1]에 명시한다.

알고리즘	참고 문헌	특징	한계점	동적 환경 회피	예측 기반	확률 기반
DWA	[3]	· 동역학을 고려한 허용 속도 계산 · 궤적 후보의 샘플 기반 최적화	· 국소 최저치 (local minima) 생성 · 빠른 동적 장애물에 즉각적인 회피 어려움 · 동적 환경에서의 여전한 충돌 발생	O	X	X
전역 DWA	[11]	· 홀로노믹 모델로의 확장 · 최적화 목적함수에 목표 지점까지의 정렬도 반영	· 목표 지점까지의 거리를 저장하는 룩업 테이블 비용 · 정적 캡쳐를 통한 장애물 판단 · 동적 환경에서의 여전한 충돌 발생	O	X	X
모델 예측 제어 DWA	[12]	· 제어 시스템을 적용한 수렴성 네비게이션 제시 · 국소 최저치를 생성하지 않음	· 동적 장애물에 적용하지 않음	X	O	X

예측 기반 DWA	[14]	· 장애물의 동작 예측 · 목적함수에 충돌 시간 반영	· 동역학을 제외한 시뮬레이션 실험 · 동적 환경에서의 여전한 충돌 발생	O	O	X
퍼지 추론을 사용한 예측 기반 DWA	[13]	· 앞선 두 시간 단계의 속도 표본 예측 및 최적화 · 적응형 뉴로 퍼지 추론[15]을 통한 목적함수의 매개변수 수정	· 동적 환경에 적용 어려움	X	O	X
충돌 예측 확률 기반 DWA	본 논문	· 동적 환경에서 장애물의 동작 예측 · 목적함수에 충돌 예측 확률 반영	· 장애물의 형태 제약 · 동적 환경에서의 여전한 충돌 발생	O	O	O

[표 1] DWA 선행 연구와 충돌 예측 확률 기반 DWA의 특징

C. 확률 기반의 충돌 회피 연구

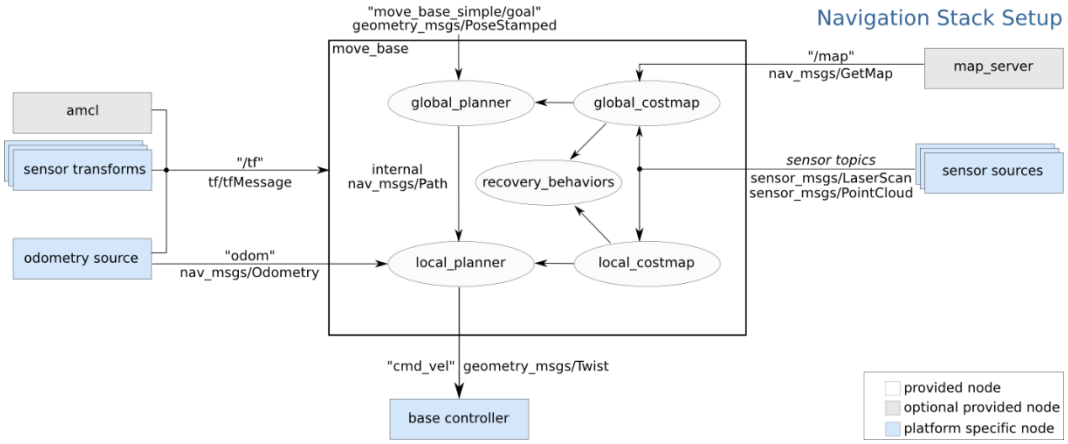
논문 [16]에서는 장애물과의 거리를 이용하여 로봇과 충돌 가능성이 있는 금지된 동작(proscriptive behavior)에 대한 확률을 계산한다. 이 연구를 확장한 [17]에서는 확률 모델에 기반하여 로봇이 원하는 명령(desired command)의 가능성을 평가하고 진행 방향을 찾는다.

불확정성에 기반한 충돌 회피 연구 역시 확률 모델을 제시한다. [18]에서는 로봇의 위치를 추정한 뒤에 속도 확률 분포를 생성하여 제약조건을 만족하는 속도 변수의 확률을 계산한다. [19]에서는 몬테카를로 위치 추정을 통한 충돌 회피 알고리즘을 제안하고, [20]에서는 로봇 위치와 센싱의 불확정성을 고려한 충돌 확률을 계산한다. 해당 연구들은 동작 및 센싱 불확실성의 가우시안 분포에 따라 이동 로봇에 주어진 동작 계획을 실행한다. 이러한 연구들은 로봇의 상태에 대한 불확실성을 고려하지만, 동적인 환경에서의 불확실성을 고려하지는 않는다. 본 논문에서는 동적 환경에서의 불확실성을 고려하여, 동적 장애물과의 충돌 확률 값에 대한 가우시안 분포를 계산한다. 이를 활용하여 로봇의 가능한 주행 방향 별 충돌 회피 확률을 계산한다.

[21], [22], [23]의 연구는 충돌 확률을 계산하는 방법을 제안한다. [21]에서는 점유 확률과 로봇의 위치 불확실성을 결합함으로써 충돌 확률을 얻어 안전한 속도를 계산한다. 충돌 예측 시간에 따라 충돌 발생 확률을 계산하는 [22], [23]에서는 각각 충돌 발생 확률을 충돌 예측 시간의 역수에 비례하는 값으로 계산하여 위험성을 예측하고, 학습한 충돌 예측 시간에 따라 충돌 확률을 계산한다. 본 논문에서는 충돌 예측 시간에 따라 로봇의 추후 이동 방향에 대한 충돌 확률을 구하고 이를 활용하여 지역 경로 계획의 비용 함수에 추가한다.

Ⅲ. 시스템 개요

A. ROS 네비게이션



[그림 3] ROS 네비게이션 [24]

본 논문의 알고리즘 구성도에 앞서 ROS와 ROS 네비게이션 시스템에 대해 간략한 설명을 한다. ROS는 로봇 동작 계획, 네비게이션, SLAM(Simultaneous Localization and Mapping) 등을 제공하는 오픈 소스 기반 로봇 소프트웨어 플랫폼이다[25]. 이 중 ROS 네비게이션은 로봇의 위치와 센서 정보를 얻어 경로를 계획하고, 속도 명령을 로봇에 전달하는 기능을 한다. ROS의 네비게이션에 대한 구성도는 [그림 3]과 같고, 네비게이션 주요 노드의 기능은 다음과 같다.

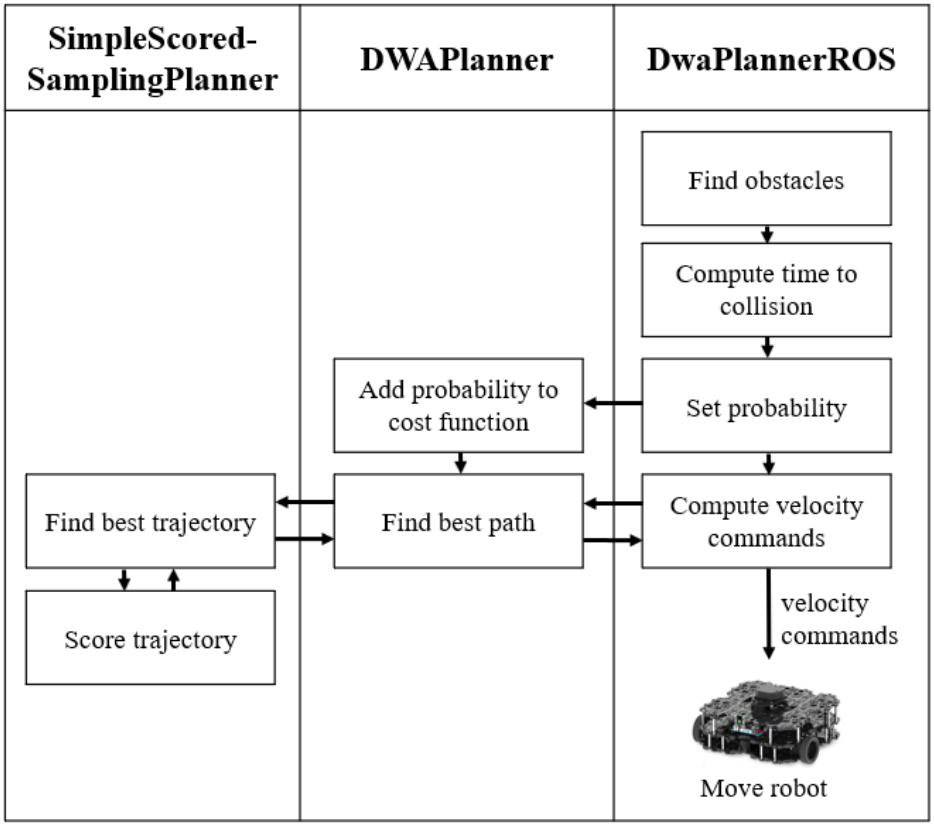
- map_server: SLAM으로 사전 제작한 지도를 네비게이션에 활용하도록 로봇 이동 노드에 전달한다. 지도에는 정적 장애물에 대한 점유 격자(occupancy grid) 정보가 주어진다.
- AMCL: 적응형 몬테카를로 위치 추정법(Adaptive Monte-Carlo Localization)을 통해 로봇의 위치를 추정한다.

- global_planner (전역 경로 계획자): A* 알고리즘을 사용하여 로봇의 현재 지점과 목표 지점까지의 전역 경로를 생성한다.
- local_planner (지역 경로 계획자): DWA 접근법을 사용하여 로봇의 지역 경로를 생성한다.

본 논문에서는 사전 제작한 지도, 로봇의 추정 위치, 전역 경로가 모두 주어질 때 제안하는 충돌 회피 알고리즘에 따라 지역 경로 계획자를 개선한다.

B. 구성도

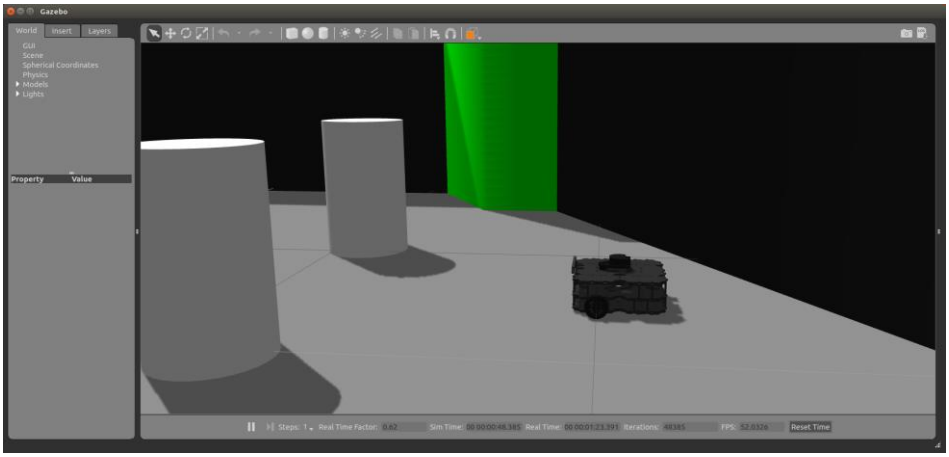
ROS 지역 경로 계획을 위한 본 논문의 충돌 회피 알고리즘 구성도는 [그림 4]와 같다. 먼저 [그림 4]의 DWAPlannerROS에서는 장애물의 궤적을 예측하고 충돌 예측 시간을 계산하여 충돌 회피 확률 모델을 생성한다. 그리고 DWAPlanner에서는 전역 경로와의 거리, 목표 지점까지의 거리, 정적 장애물에 대한 비용을 계산하는 기존 DWA 비용함수에 회피 확률 값을 한 요소로 추가한다. 그리고 SimpleScoredSamplingPlanner는 가능한 모든 경로의 비용함수 값을 계산하여 이를 최소화하는 최적의 경로를 찾아 반환한다. 마지막으로 DWAPlannerROS는 반환받은 경로를 속도 명령으로 변환하여 로봇을 주행하게 한다. 이 과정은 로봇이 목표 지점에 도달할 때까지 반복되며, 접근법의 자세한 내용은 다음 장에서 설명한다.



[그림 4] 알고리즘 구성도

C. 실험 환경

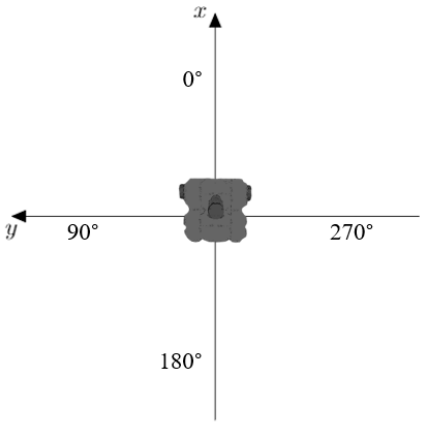
본 논문의 개발 및 성능 실험은 [그림 5]와 같이 물리 엔진이 탑재된 Gazebo 시뮬레이터에서 차동 구동형 모바일 로봇(differential drive mobile robot) 터틀봇3를 사용하여 진행한다. 터틀봇3에는 동역학, 기구학적 제약에 따른 주행 속도 제한 값이 존재한다.



[그림 5] Gazebo 시뮬레이터 상의 터틀봇3

터틀봇3의 2D-LiDAR 센싱에는 LDS-01(Laser Distance Sensor, 레이저로 물체와의 거리 값을 반환하는 센서)[그림 6]을 사용한다. 이 센서는 360도 전방으로 감지하며, 근방 최대 3.5m를 감지할 수 있다. 센서의 분해능(angular resolution)은 1도이다. 따라서 센서가 측정하는 값은 로봇 기준 각 1도마다 근방 장애물과의 거리 값이 된다. LDS-01을 장착한 터틀봇3의 센싱 방향은 [그림 7]과 같다. 로봇의 전방은 0°로 시작하고 반시계방향으로 1°씩 증가하는 곳에서 물체와의 거리를 감지한다. 터틀봇3의 주행 방향 체계 역시 센싱 방향 체계와 동일하며, 이는 4장에서 이동 방향 별 충돌 회피 확률을 계산할 때 적용된다. 로봇의 위치는 주행 기록 정보인 휠 오도메트리(wheel odometry)를 통해 추정한다.

[그림 6] 터틀봇3 LDS-01 센서 [26]



[그림 7] 터틀봇3 방향 체계

D. 가정 및 제약 사항

본 논문에서는 개발 및 시뮬레이션 성능 실험 시 다음과 같은 사항을 가정하고 제한한다.

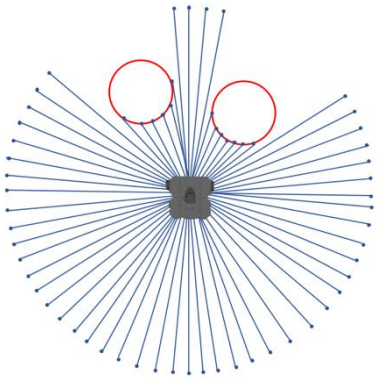
1) 장애물의 궤적을 예측할 때, 장애물이 직전 시간으로부터 등속 운동한다고 가정한다.
2) 로봇의 2D 거리 센싱에는 오차가 없다.
3) 로봇의 오도메트리 값 및 위치 정보가 정확하다.
4) 실험 환경에서 움직이는 모든 장애물은 원통형 모양으로 제한한다.

Ⅳ. 궤적 및 충돌 예측 기반 충돌 회피 알고리즘

본 장에서는 제안하는 알고리즘을 실행 순서에 따라 차례로 기술한다. A, B 절에서는 2D LiDAR 감지를 통한 장애물 지점의 위치 계산 및 동적 장애물 분리에 관해 설명한다. 그리고 C, D절에서는 분리한 동적 장애물의 궤적 예측과 충돌 예측 시간 계산에 관해 설명한다. 마지막으로 E절에서는 충돌 확률을 계산하여 이동 방향의 유효성을 확률 분포를 통해 평가하고 적절한 속도를 찾아 회피하는 방법을 다룬다.

A. 장애물 감지 및 위치 계산

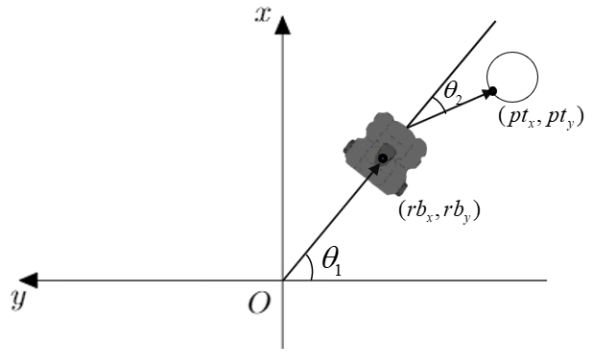
근방 장애물 감지 및 위치 계산 시, 레이저 스캔 값을 실시간으로 측정하면서 [그림 8]과 같이 유효 범위 내에서 감지된 지점 (point)들의 위치를 계산한다. 계산을 위해 센서에서 측정한 거리 값과 각도 값을 활용한다.



[그림 8] 레이저 감지

장애물 지점들의 위치는 월드 좌표계 (world frame)를 기준으로 계산한다. [그림 9]와 같이 로봇의 위치 (rb_x, rb_y)는 월드 좌표계를 기준으로 θ_1 각도에 위치

한다. 그리고 감지한 장애물 지점의 좌표 (pt_x, pt_y)는 로봇 좌표계 (robot base frame)를 기준으로 각도(θ_2), 거리($dist$)에 위치한다. 로봇 좌표계와의 상대 위치를 고려하여 월드 좌표계에서의 장애물 지점 위치를 수식 (1)과 같이 구한다.



[그림 9] 로봇 기준 장애물 지점의 위치

$$pt_x = rb_x + dist \cdot \cos(\theta_1 + \theta_2)$$
$$pt_y = rb_y + dist \cdot \sin(\theta_1 + \theta_2) \quad (1)$$

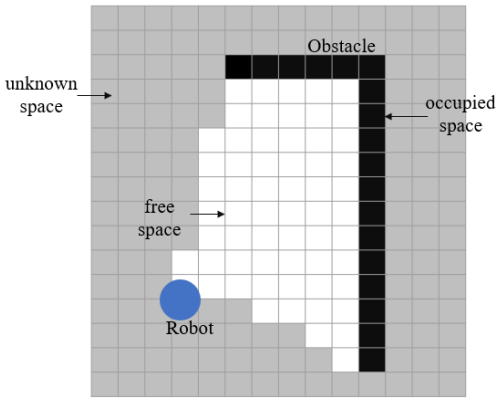
B. 동적 장애물 인식

1. 정적 그리고 동적 장애물 분리

앞선 절에서 장애물 지점들의 실시간 월드 좌표계 위치를 구하였다. 그러나 동적 장애물의 속도를 정확히 계산하기 위해서는 감지한 장애물 지점들 중 정적 지점 (static point)과 동적 지점 (dynamic point)을 분리하는 과정이 필요하다.

이를 위해 본 논문에서는 ROS에서 제공하는 2차원 점유 격자 지도

(occupancy grid map) [그림 10]를 활용한다[27]. 점유 격자 지도는 SLAM을 통해 네비게이션 지도를 만들 때 지도의 각 격자 (grid)에 점유도 값을 저장한다. 정적 장애물이 위치하여 로봇의 이동이 불가능한 지역은 점유된 공간 (occupied space)으로 간주하여 각 격자에 100을 저장한다. 이동이 가능한 지역은 자유 공간 (free space)으로 간주하며 0을 저장하고, 알 수 없는 공간은 알려지지 않은 공간 (unknown space)으로 간주하여 −1을 저장한다.



[그림 10] 점유 지도 격자

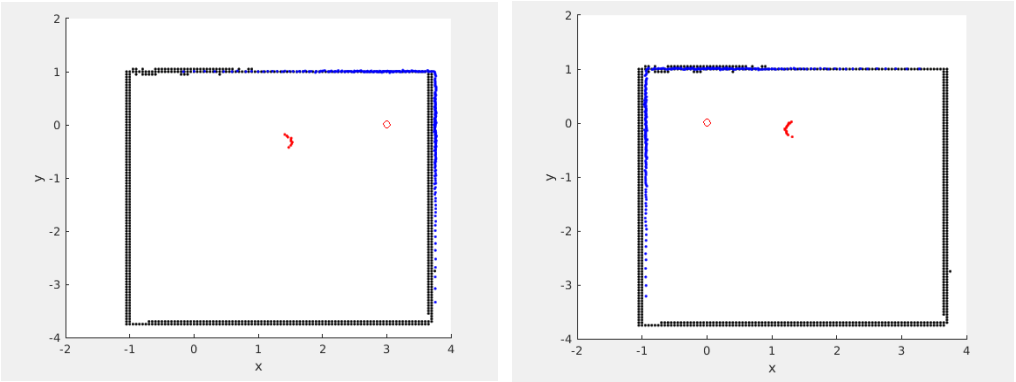
본 논문에서는 이 점유 격자 지도에 저장된 위치 정보를 활용하여 정적 지점과 동적 지점을 분리한다. 이를 위해, 먼저 정적 지도 (static map)의 크기와 규모 (scale) 정보를 이용하여 지도 내 점유 공간의 위치를 월드 좌표계로 변환한다. 정적 장애물 지점은 임의의 점유 공간 내에 위치하므로, 해당 점유 공간을 나타내는 좌표와 인접한 거리에 위치한다고 간주한다. 이에 따라, 격자 점유 값이 100인 지점들과 센서에서 감지한 지점들의 유클리드 거리 (Euclidean distance)가 특정 임계 값(0.15m) 내에 존재하는지 확인한다. 감지한 각 지점을 비교하여 임계 값 내에 존재하는 경우에는 정적 지점으로 간주하고, 존재하지 않으면 동적 지점으로 분리한다. 이 방법은 동적 지점이 정적 지도와 근접한 지점에 있을 경우, 정확한 분리가 어렵다는 한계를 보인다. 이를 설명하는 순서도는 [알고리즘 1]

과 같다.

Algorithm 1 Obstacle segmentation

Input: sensed points P
  for all points $pt \in P$ **do**
    Find point in map which its distance is less than threshold value
    for all map points $m\_pt \in Map$ **do**
      if $dist(pt, m\_pt)$ is same or less than 0.15m **then**
        pt = static point
        terminate and break
      end if
    end for
    if not found **then**
      pt = dynamic point
    end if
  end for

[알고리즘 1] 정적 장애물과 동적 장애물 분리 알고리즘



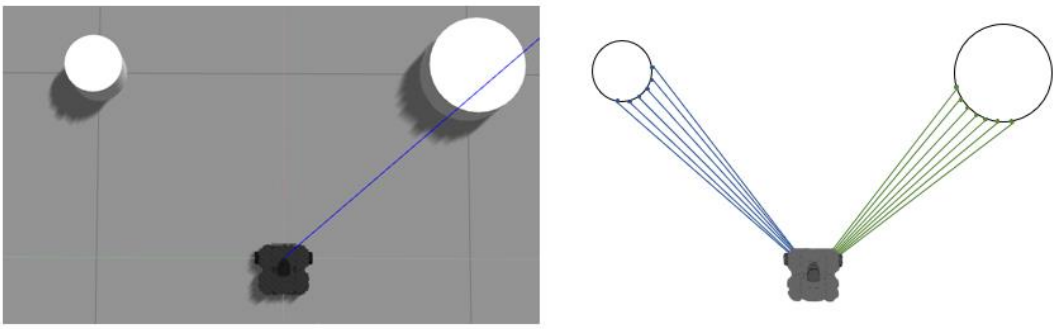
[그림 11] 정적 지점과 동적 지점의 분리 결과 그래프

[그림 11]은 로봇이 감지하는 지점들이 오류 없이 분리됨을 보인다. [그림 11]의 검은 점은 지도에서 격자 점유 값이 100인 지점들, 즉 미리 지도에 매핑된

정적 장애물을 나타내고, 붉은색 작은 원은 이동하는 로봇의 위치를 나타낸다. Algorithm 1에 의거하여 분리한 결과에 따라 로봇이 감지하는 정적 장애물은 파란색 점으로, 동적 장애물은 붉은색 점으로 표시하였다. 정적 지점을 동적 지점으로 간주하는 것이 없고, 반대의 경우도 나타나지 않는다. 이로써 제안하는 방법을 통해 센서가 감지하는 장애물 지점 중 동적 지점과 정적 지점을 적절히 분리할 수 있음을 알 수 있다.

2. 장애물 구별

서로 다른 동적 장애물은 각자 다른 속도로 움직이므로 분리해서 충돌을 예측할 필요가 있다. 따라서 동적인 지점들을 추출한 이후에는 서로 다른 동적 장애물 객체를 분리한다.



(1)Gazebo 시뮬레이터의 두 동적 장애물　　(2) 두 동적 장애물의 분리

[그림 12] 동적 장애물 구별

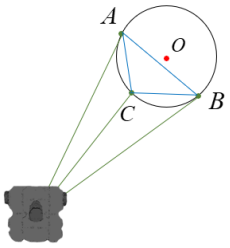
[그림 12]의 (1)과 같은 환경에서 로봇이 두 동적 장애물을 나누어 감지하도록 한다. 이를 위해 동적 지점들을 감지하는 센서의 감지 각도를 활용하고, 각도 인덱스의 연속성을 확인함으로써 과정을 수행한다. 터틀봇3은 센서의 센싱 범위 360도를 분해능 값으로 나눈 총 360개의 인덱스를 확인하여 [그림 12]의 (2)와

같이 연속된 인덱스를 갖는 첫 번째 장애물과 두 번째 장애물을 분리한다. [그림 12]의 (2)의 연속된 파란색 점들처럼 인덱스가 연속한 경우에는 연속된 지점들을 같은 장애물로 간주한다. 그리고 [그림 12]의 (2)의 파란색 점과 초록색 점과 같이 인덱스가 연속하지 않으면 서로 다른 장애물로 간주하고 동적 장애물 객체를 분리한다. 이는 로봇의 정확한 2D 센싱을 가정하고 진행하므로 센싱 오류가 존재하는 실제 시스템에서는 정확한 계산이 어려울 수 있다. 또한, 장애물이 가려지거나 여러 장애물이 인접하는 경우 각도 인덱스의 연속성을 이용하여 장애물을 구별하는 것이 어려울 수 있다.

C. 장애물 궤적 예측

1. 장애물 위치 추정

장애물의 궤적을 예측하기에 앞서 장애물의 위치를 추정한다. 이를 위해 앞선 절에서 분리한 각 동적 장애물 지점들의 위치를 활용한다. 본 절에서는 장애물의 형태를 원형으로 가정하므로 감지하는 지점 중 세 지점을 선택해 삼각형을 구성하고 그 외심을 구하여 장애물의 중심 위치를 추정한다.



[그림 13] 장애물의 위치 추정

[그림 13]에서 동일 장애물을 감지하는 연속된 인덱스 중 가장 첫 인덱스가 감

지하는 지점을 A 지점으로, 가장 마지막 인덱스가 감지하는 지점을 B 지점으로 설정한다. 그리고 해당 장애물과의 레이저 감지 거리가 가장 짧은 지점을 C 지점으로 설정한다. 삼각형 ABC 의 외심인 O 지점을 장애물의 중심 위치로 추정한다.

이 방법을 사용하여 장애물의 위치를 추정할 경우, 추정 값과 실제 시뮬레이션 환경의 정확한 장애물 위치 값은 유클리드 거리로 0.04 ~ 0.08m의 근소한 오차를 보인다.

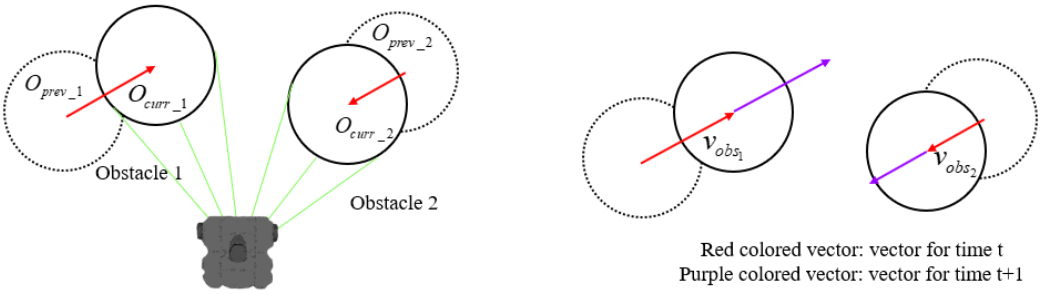
2. 장애물 이동 경로 계산 및 궤적 예측

본 항에서는 장애물의 위치 정보를 이용하여 장애물의 이동 경로를 계산하고, 시간 범위(time horizon) 내의 궤적을 예측한다. 예측을 위해서는 이전 시간 단계와 현재 시간 단계에서의 장애물의 이동 경로를 확인한다.

먼저 장애물의 궤적을 연속적으로 확인하기 위해서 각 장애물에 고유의 인덱스 i (1≤ i ≤장애물 개수)를 부여한다. 각 장애물의 인덱스에는 장애물의 위치를 저장하여 동일한 인덱스를 갖는 장애물의 이동 경로를 확인한다. 그리고 주어지는 장애물의 위치 정보로 이전 단계와 현재 단계에서 감지하는 동일한 장애물의 인덱스를 연결하기 위해, 현재 측정한 장애물들의 위치 정보와 이전에 측정한 장애물들의 위치 값을 비교한다. 현재 단계에서 측정한 장애물의 위치(O_{temp})와 위치 차가 가장 작은 이전 단계 장애물의 위치(O_{prev_i})를 찾아 동일한 인덱스 i를 부여하고, i번째 장애물의 현 위치(O_{curr_i})에는 측정한 장애물의 위치(O_{temp}) 값을 설정한다. 이 방법은 위치 차가 가장 작은 이전 단계의 장애물이 두 개 이상 측정

될 경우 해당 장애물의 시간에 따른 궤적을 정확히 측정하지 못하므로, 이러한 경우는 발생하지 않는다고 가정한다.

이어지는 시간 단계에서 동일한 인덱스의 장애물을 찾은 뒤에는 [그림 14]의 (1)과 같이 이전 단계와 현재 단계의 위치 차로 각 장애물의 이동 경로를 계산한다.



(1) 장애물의 이동 경로 계산 (2) 장애물의 궤적 예측

[그림 14] 장애물의 궤적 예측

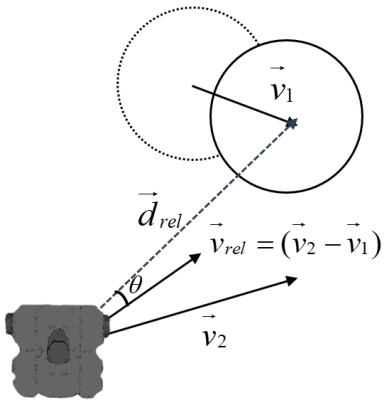
[그림 14]의 (2)와 같이 두 장애물이 각각 시간 t에서 v_{obs_1}, v_{obs_2}로 이동했을 경우, t+1에서도 시간 t와 각각 동일한 벡터로 이동한다고 가정하므로 시간 t+1에서의 이동 벡터를 각각 v_{obs_1}, v_{obs_2}로 예측한다. 본 연구에서는 장애물이 등속 운동한다고 가정하므로, 장애물의 동작 변화가 급격하게 일어나는 경우 장애물의 궤적을 정확히 예측하는 것은 어렵다.

D. 충돌 예측 시간 계산

본 절에서는 로봇과 근방 장애물들의 충돌 예측 시간을 계산한다. 이동하는 로봇과 이동하는 장애물 간의 충돌 예측 시간을 2차원 평면상에서 구하기 위해

[28]에서 제안하는 방식을 사용하여 계산하였다.

충돌 예측 시간(TTC)은 수식 (2)와 같이 시간 t+1에서의 로봇과 장애물의 거리($\left|\overrightarrow{d_{rel}}\right|$)를 로봇과 장애물의 상대 속도 크기($\left|\vec{v}_{rel}\right|\cos\theta$)로 나누어 구한다. [그림 15]와 같이 로봇의 상대 속도를 구할 때는 로봇의 이동 벡터($\vec{v}_2$)와 앞선 절에서 예측한 장애물의 이동 벡터($\vec{v}_1$)를 이용한다. 이 방법으로 로봇이 감지하는 장애물마다 충돌 예측 시간을 구한다.



[그림 15] 충돌 예측 시간 계산

$$TTC = \frac{\left|\overrightarrow{d_{rel}}\right|}{\left|\overrightarrow{v_{rel}}\right|\cos\theta} \qquad (2)$$

E. 충돌 확률 계산 및 회피 알고리즘

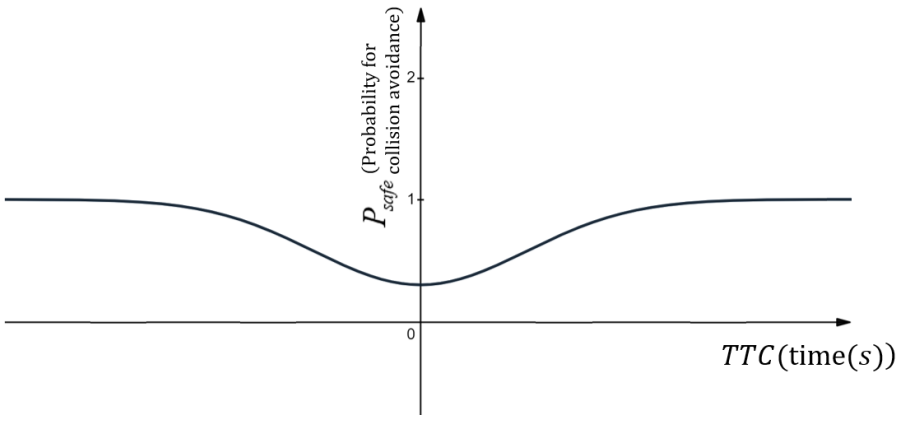
1. 충돌 확률 계산

본 항에서는 앞선 절에서 구한 충돌 예측 시간 값에 기반하여 각 장애물과의 충

돌 확률 (P_{coll})을 계산한다. 충돌 확률 계산은 [13]에서 제안하는 계산법을 활용한다. 이 계산 결과에 따르면 충돌 예측 시간이 길수록 충돌 확률이 더 낮게 계산되고, 예측 시간이 짧을수록 충돌 확률이 높게 계산된다.

충돌 확률 값을 구한 뒤에는 각 장애물에 대한 충돌 회피 확률(P_{safe})을 계산한다. 이 값은 충돌 예측 시간이 길수록 더 높게 계산되며, 예측 시간이 짧을수록 더 낮게 계산된다. 이 충돌 회피 확률 값은 해당 장애물과 가장 짧은 직선거리를 갖는 로봇의 방향과 함께 저장되어 이후의 각 주행 방향 별 충돌 회피 확률 계산에 활용된다. 수식 (3)에 따른 P_{safe} 값의 그래프는 [그림 16]과 같다. 그래프의 x축은 충돌 예측 시간을 나타내고, y축은 충돌 회피 확률을 나타낸다.

$$\begin{cases} P_{coll} = \alpha e^{-\beta(TTC)^2} \\ P_{safe} = 1 - P_{coll} \end{cases} \quad (3)$$



[그림 16] 충돌 예측 시간에 따른 충돌 회피 확률

이후에는 가능한 주행 방향 별 충돌 회피 확률(P_{safe_dir})을 로봇 전방으로 계산한다. P_{safe_dir} 값은 충돌 위험이 높은 방향으로의 주행을 피하고, 적절한 진행 방향을 찾게 한다. 이 계산은 각 장애물의 P_{safe} 값과, 로봇-장애물 간 방향을 활용한다.

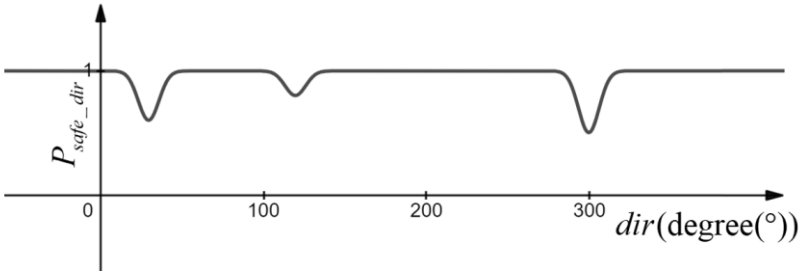
P_{safe_dir} 값의 계산은 하단의 [알고리즘 2]를 참조하여 설명한다. 장애물 방향으로 최단 거리의 로봇 주행 방향($dir_{obs^{i-th}}$, $1 \leq i \leq$장애물 개수)뿐만 아니라, 그 인접한 방향(dir) 역시 주행 시 충돌의 위험성이 있다고 판단하여, 주변의 충돌 위험성이 존재하는 방향에는 근사한 P_{safe} 값을 설정한다. 그리고 한 방향에 여러 장애물의 영향으로 서로 다른 P_{safe} 값이 주어지는 경우에는 가장 작은 P_{safe} 값을 P_{safe_dir} 값으로 택한다.

Algorithm 2 Compute P_{safe_dir}

Input: possible directions D
  for all directions $dir \in D$ **do**
    min = MAX_VAL
    for all obstacles obs_i $(1 \leq i \leq$ obstacle number$) \in O$ **do**
      Find robot direction $dir_{obs^{i-th}}$ nearest to obs_i and P_{safe} for obs_i
      compute $P_{temp} = (1 - (\alpha e^{(-\frac{(dir - dir_{obs^{i-th}})^2}{\beta})})(1 - P_{safe})$
      if P_{temp} is less than min **then**
        $min = P_{temp}$
      end if
    end for
    Add min to P_{safe_dir}
  end for

[알고리즘 2] 가능한 주행 방향 별 충돌 회피 확률

결과는 [그림 17]의 그래프와 같다. 그래프의 x축은 로봇의 주행 가능한 전 방향을 나타내고, 그래프의 y축은 해당 방향의 충돌 회피 확률을 나타낸다. [그림 17]은 3개의 장애물이 근방에 존재하는 상황에서 로봇이 주행할 각 방향의 충돌 회피 확률을 나타낸다. 1에 근접할수록 충돌의 위험이 없음을 의미하고, 0에 근접할수록 주행 시 충돌의 위험이 커짐을 의미한다.



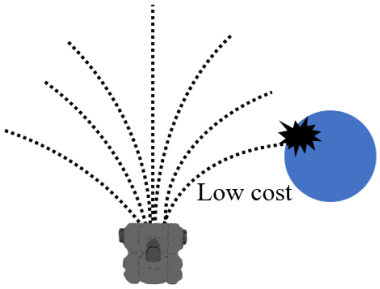
[그림 17] 로봇의 주행 방향 별 충돌 회피 확률

2. 충돌 회피 확률에 기반한 비용함수 계산

본 항에서는 로봇의 주행 방향 별 충돌 회피 확률을 DWA 비용함수의 계산에 반영한다. 먼저 기존의 DWA 비용함수는 전역 경로와의 거리(d_{global_path}), 목표 지점까지의 거리(d_{goal}), 정적 장애물에 대한 양의 비용($cost_{known\_obs}$)을 고려하여 속도 샘플을 평가한다. 이에 덧붙여, 수식 (4)와 같이, 충돌 회피 확률을 목적함수의 한 요소로 추가하여 속도 샘플을 평가한다. 이때, 수식 (4)는 실제 항목들의 차원을 스칼라 값으로 표현한 의사 거리 값(pseudo metric)이며, 매개변수의 규모(δ)는 실험을 통해 적절한 값을 찾는다. 수식 (4)의 비용함수는 비용 값을 최대화하는 속도를 최적의 속도로 계산한다. 따라서 수식 (4)의 $\delta \cdot P_{safe\_dir}$ 요소는

[그림 18]과 같이 충돌 회피 확률이 낮은 각속도의 조합에서는 낮은 비용 값을, 충돌 회피 확률이 높은 각속도의 조합에서는 높은 비용 값을 갖는다. 이로써 로봇이 주행 속도의 유효성을 충돌 확률 값을 통해 평가하고 적절한 진행 방향을 찾아 주행하게 한다.

$$\text{cost} = \alpha \cdot d_{global\_path} + \beta \cdot d_{goal} + \chi \cdot cost_{known\_obs} + \delta \cdot P_{safe\_dir} \qquad (4)$$



[그림 18] 속도 샘플의 충돌 회피 확률 비용

로봇은 목표 지점에 도달할 때까지 본 장에서 제안하는 알고리즘의 계산을 반복하여 진행한다. 제안하는 알고리즘을 적용한 로봇의 지역 경로 계획법이 동적 장애물에 적절한 회피 동작을 생성함을 5장의 실험 결과에서 논의한다.

Ⅴ. 실험 결과 및 해석

A. 실험

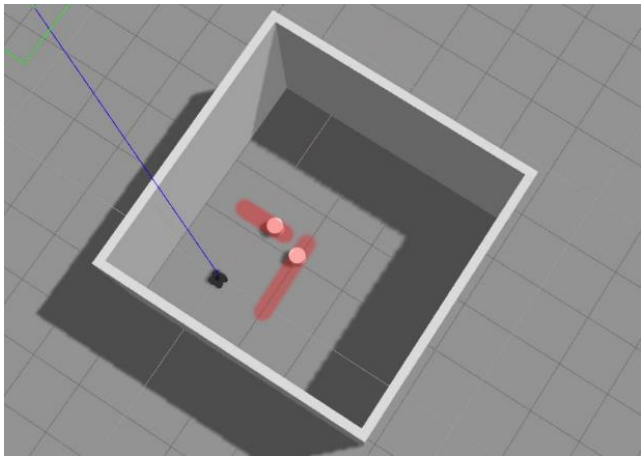
본 논문에서 제안하는 알고리즘의 실험은 Intel Core i7 CPU, 16GB RAM, NVIDIA GeForce GTX 745 GPU를 사용하여 Ubuntu 16.04 LTS 64-bit 운영체제의 ROS Kinetic 플랫폼에서 진행한다. 실험 환경은 물리 엔진을 제공하는 Gazebo 시뮬레이터에서 정적 장애물과 동적 장애물이 주어지도록 설정한다.

본 장에서는 실험을 위해 터틀봇3의 네비게이션을 실행한다. 전역 경로 계획 및 위치 추정에 필요한 사전 지도는 정적 장애물만을 사용하여 제작되었다. 지역 경로 계획은 본 논문에서 제안하는 알고리즘으로 대체하고, 전역 경로 계획, 지도 매핑, 위치 추정 등 네비게이션에 필요한 다른 요소들은 ROS에서 제공하는 기본 패키지를 사용한다.

성능 비교를 위해 동일한 환경에서 기존의 DWA 알고리즘과 본 논문에서 제안하는 알고리즘의 충돌 회피 동작, 전체 경로, 그리고 소요 시간을 비교한다.

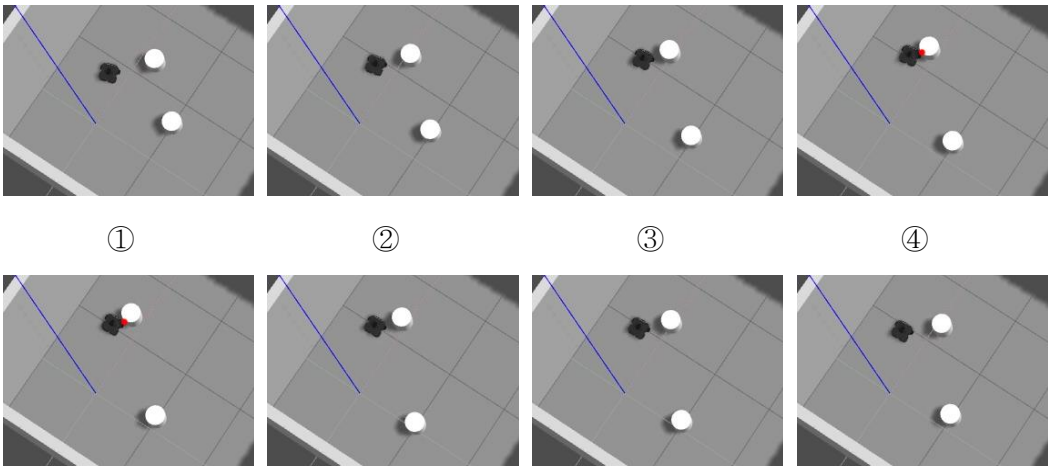
1. 2개의 장애물

2개의 동적 장애물과 정적 장애물(벽)이 주어진 환경에서 로봇의 지역 경로 계획을 확인한다. 서로 다른 장애물은 각자 주어진 속도로 이동하며, [그림 19]의 붉은 영역은 동적 장애물의 이동 범위를 나타낸다.



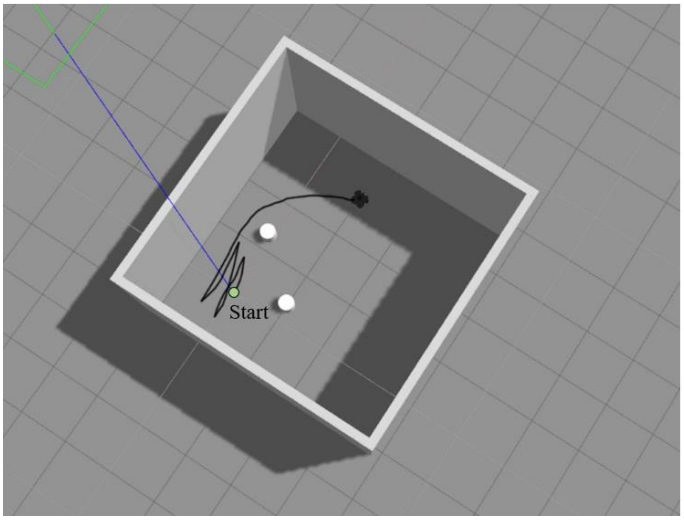
[그림 19] 동적 장애물이 2개 주어진 실험 환경

본 절에서는 해당 시뮬레이션 환경에서 기존의 DWA 알고리즘과 본 논문에서 제안하는 알고리즘의 충돌 회피 동작을 차례로 확인하며 비교한다. 기존 DWA의 충돌 회피에 대한 실험 결과는 [그림 20], [그림 21]에서 확인하며, 제안하는 알고리즘의 실험 결과는 [그림 22]에서 확인한다.



[그림 20] 2개 장애물 실험 시 DWA의 충돌 상황

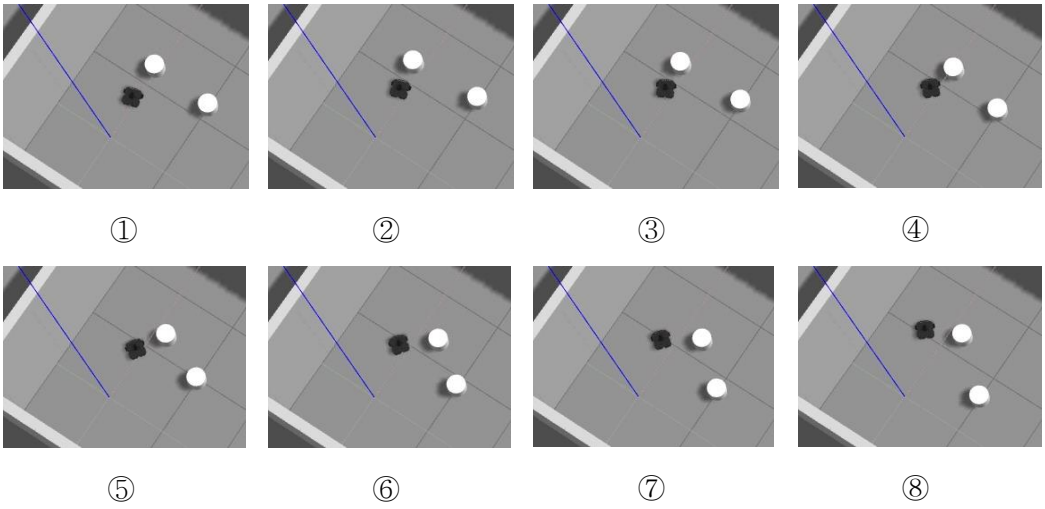
[그림 20]의 ①부터 ⑧까지의 과정은 동적 장애물에 근접할 때의 로봇 동작 및 충돌 상황을 나타낸다. ④, ⑤ 과정의 붉은 점은 장애물과의 충돌 지점을 나타낸다. [그림 20]은 기존의 DWA는 동적 장애물에 적절한 충돌 회피 경로를 생성하는 것이 어렵다는 것을 확인할 수 있다.



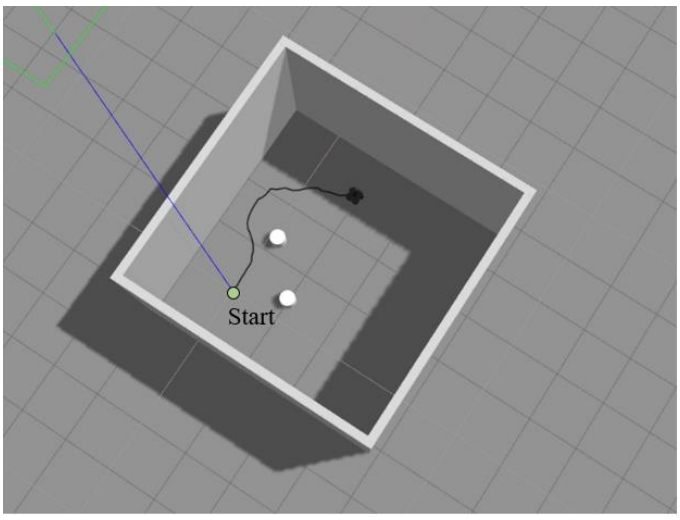
[그림 21] 2개 장애물 실험 시 DWA의 주행 경로

또한, DWA는 충돌 회피 경로를 생성하는 경우에도 이동 거리나 시간의 효율성이 문제가 될 수 있다. [그림 21]은 장애물과 충돌하지 않은 DWA의 주행 실험 결과와 그 경로를 보여준다. 검은 실선은 로봇이 목표 지점에 도달할 때까지 DWA로 계획한 주행 경로를 나타낸다. 해당 경로는 로봇이 동적 장애물과 근접할 경우 후방 주행을 하는 것을 확인할 수 있다. 이는 궤적 후보 중 DWA의 비용함수가 양의 값을 갖는 가능한 지역 경로가 없다고 판단할 때, 충돌을 피하며 전역 경로를 다시 계획하는 과정에서 발생하는 현상이다. 하지만 이 현상이 반복됨에 따라 주행 경로가 상당히 길어지는 것을 확인할 수 있고, 이는 목표 지점까지의 주행 시간을 증가시켜 효율적인 경로 계획을 어렵게 한다.

이에 반해, 본 논문에서 제안하는 새로운 알고리즘은 유연한 충돌 회피 동작을 생성한다. 제안하는 알고리즘의 충돌 회피를 실험한 결과는 [그림 22]와 같다.



(1) 제안하는 알고리즘의 충돌 회피 동작



(2) 제안하는 알고리즘의 주행 경로

[그림 22] 2개 장애물 실험 시 제안하는 알고리즘의 충돌 회피 실험

[그림 22]의 (1)에서는 근방의 동적 장애물을 감지할 경우 로봇이 적절한 충돌 회피 경로를 생성한다. ②, ③ 단계에서 장애물의 움직임을 예측하고 (장애물은 오른쪽으로 이동), 로봇은 이후 ③, ④, ⑤ 단계에서 왼쪽으로 이동하여 충돌 상황을 피한다.

[그림 22]의 (2)의 경로를 확인하면 로봇이 시작 지점부터 목표 지점까지 충돌 없는 지역 경로를 생성하고 주행한다. 전역 경로 재계획(re-planning)이 필요하지 않으므로, 제안하는 방법은 기존의 알고리즘보다 주행 경로 길이와 시간을 단축하여 더 거리-효율적이고, 시간-효율적인 주행을 가능하게 한다.

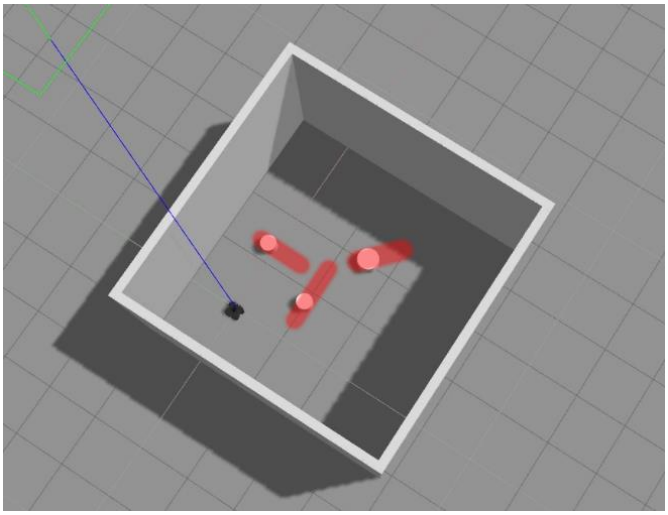
동일한 환경에서 네비게이션 실험을 진행한 결과에 대한 요약[표 2]은 다음과 같다. [표 2]의 충돌 회피 성공률은 전체 실험 횟수에 대한 충돌 없는 경로 생성 횟수의 비율로 정의한다. 본 논문에서 제안하는 알고리즘은 충돌 회피 성공률(= 충돌 없는 경로 생성 횟수 / 전체 실험 횟수)이 80%로 DWA의 40%보다 2배 높음을 보인다. 그리고 평균 주행 시간이 기존 DWA 대비 2배 이상 단축됨을 보인다.

	Success rate for collision avoidance	Max time	Mean time
Original DWA	40%	48.0s	33.0s
Ours	80%	14.0s	13.5s

[표 2] 2개의 장애물 실험 시 DWA와 제안하는 알고리즘의 성능 비교

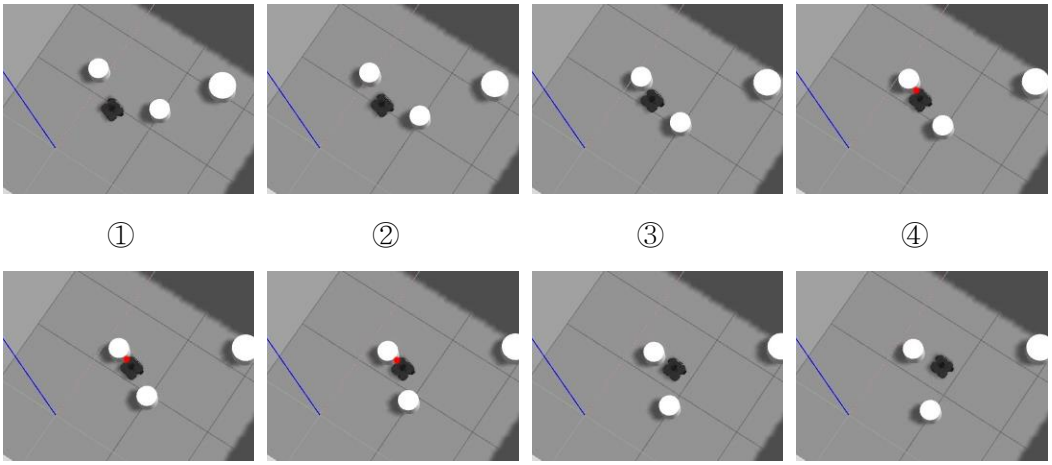
2. 3개의 장애물

3개의 동적 장애물과 벽(정적 장애물)이 주어진 상황에서 로봇의 지역 충돌 회피를 확인한다. 장애물의 이동 범위는 [그림 23]의 붉은 영역이다.



[그림 23] 동적 장애물이 3개 주어진 실험 환경

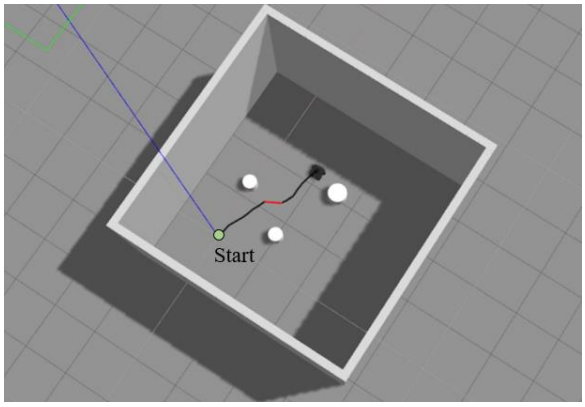
본 항 역시 동일 시뮬레이션 환경에서 기존의 알고리즘과 제안하는 알고리즘의 충돌 회피 동작을 차례로 확인하며 비교한다. 기존 DWA의 충돌 회피를 실험한 결과는 [그림 24], [그림 25]에서 확인하며, 제안하는 알고리즘의 실험 결과는 [그림 26], [그림 27]에서 확인한다.



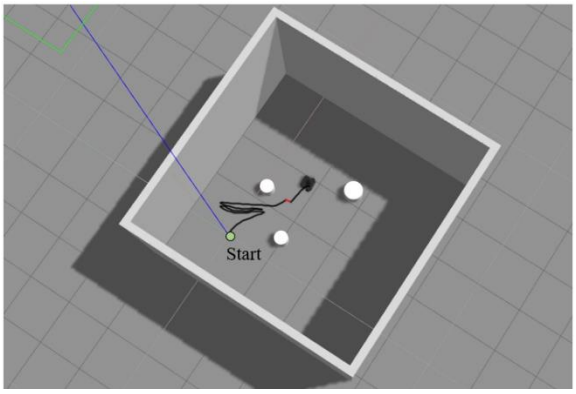
[그림 24] 3개 장애물 실험 시 DWA의 충돌 상황

[그림 24]의 ①부터 ⑧까지의 과정은 동적 장애물에 근접할 때의 로봇 동작 및 충돌 상황을 나타낸다. ④, ⑤, ⑥ 과정의 붉은 점은 장애물과의 충돌 지점을 나타낸다.

기존 DWA의 로봇 주행 경로는 [그림 25]와 같다. 경로의 붉은 선은 로봇이 장애물에 충돌하면서 밀린 경로를 나타낸다. [그림 25]의 (1)은 충돌이 발생하지만, 전역 경로 재계획을 실행하지 않는다. 하지만 [그림 25]의 (2)에서는 전역 경로 재계획을 실행하여 주행 경로가 상당히 길어진 것을 확인할 수 있다.



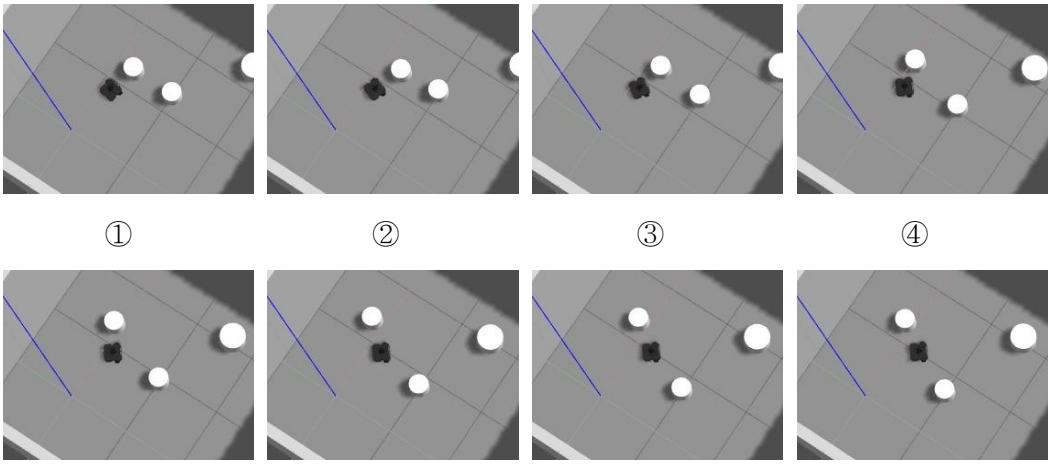
(1) 전역 경로 재계획을 실행하지 않은 경로



(2) 전역 경로 재계획을 실행하는 경로

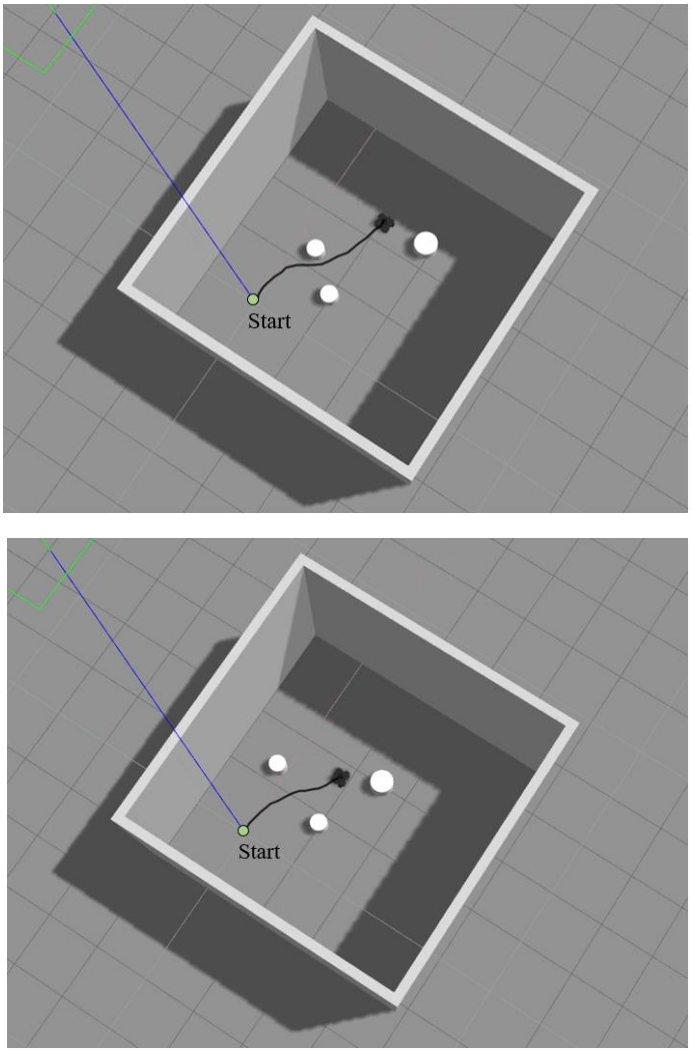
[그림 25] 3개 장애물 실험 시 DWA의 주행 경로

[그림 26], [그림 27]은 각각 본 논문에서 제안하는 알고리즘의 충돌 회피 경로 생성과 주행 경로를 나타낸다. [그림 26]에서는 근방의 동적 장애물을 감지할 경우 로봇이 적절한 충돌 회피 경로를 생성한다. ①, ②, ③ 단계에서 장애물의 움직임을 예측하고 (장애물은 왼쪽으로 이동), 로봇은 이후 ③, ④, ⑤ 단계에서 오른쪽으로 주행하여 충돌 상황을 피한다.



[그림 26] 3개 장애물 실험 시 제안하는 알고리즘의 충돌 회피

[그림 27]은 동일한 환경에서 서로 다른 목표 지점을 설정하였을 때, 제안하는 알고리즘이 [그림 25]와 비교하여 적절한 충돌 회피 경로를 생성하는 동시에 효율적인 경로로 주행함을 보인다.



[그림 27] 3개 장애물 실험 시 제안하는 알고리즘의 두 가지 경로

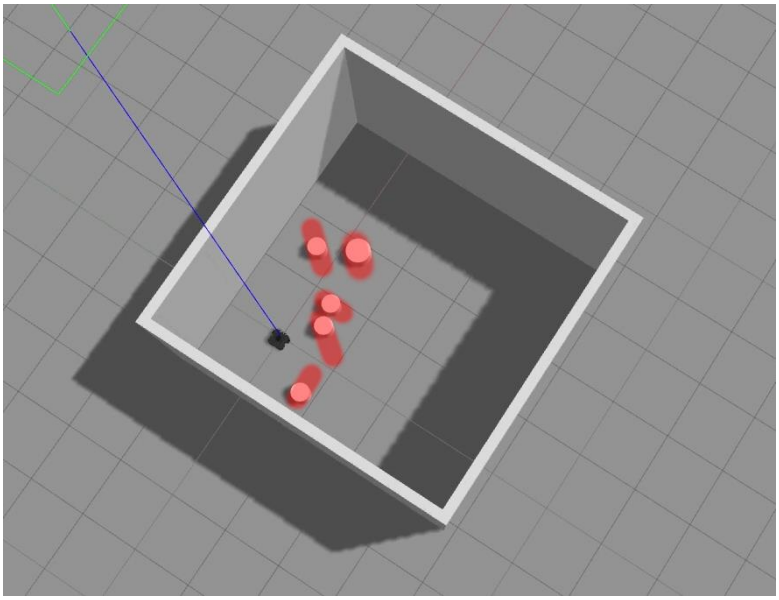
동일한 환경에서 네비게이션 실험을 진행한 결과에 대한 요약[표 3]은 다음과 같다. 본 논문에서 제안하는 알고리즘은 충돌 회피 성공률이 80%로 DWA의 40% 보다 높다. 그리고 평균 주행 시간이 기존 알고리즘 대비 2배 단축된다.

	Success rate for collision avoidance	Max time	Mean time
Original DWA	40%	45.0s	18.6s
Ours	80%	11.0s	9.4s

[표 3] 3개의 장애물 실험 시 DWA와 제안하는 알고리즘의 성능 비교

3. 5개의 장애물

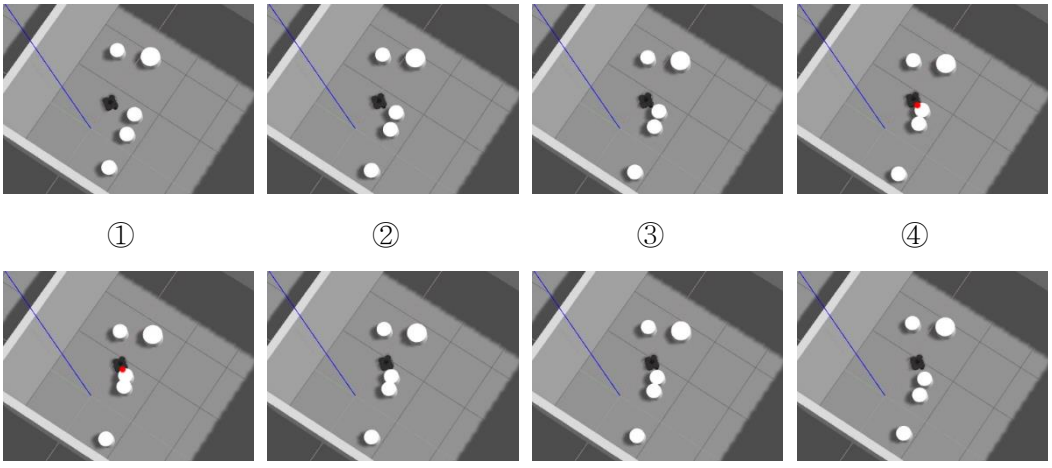
5개의 동적 장애물과 벽(정적 장애물)이 주어진 상황에서 네비게이션 실험을 진행한다. 장애물의 이동 범위는 [그림 28]의 붉은 영역이다.



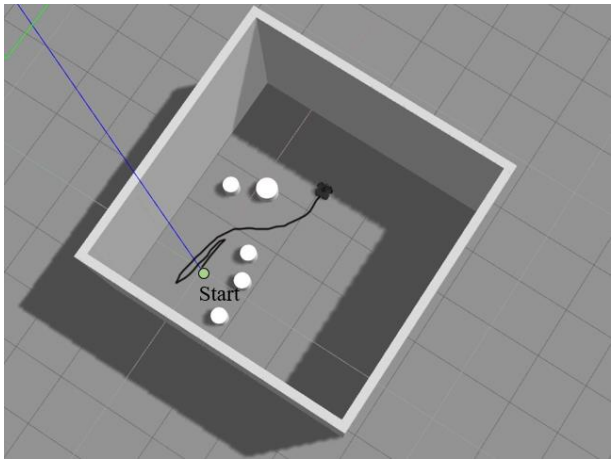
[그림 28] 동적 장애물이 5개 주어진 실험 환경

본 항 역시 동일 환경에서 DWA와 제안하는 알고리즘의 충돌 회피 동작을 차례로 확인하며 비교한다. DWA의 충돌 회피를 실험한 결과는 [그림 29], [그림 30]에서 확인하며, 제안하는 알고리즘의 실험 결과는 [그림 31]에서 확인한다.

[그림 29]의 ①부터 ⑧까지의 과정은 동적 장애물에 근접할 때의 로봇 동작 및 충돌 상황을 나타낸다. ④, ⑤ 과정의 붉은 점은 장애물과의 충돌 지점을 나타낸다. [그림 30]의 경로 역시 이전 항의 DWA 실험과 같이, 전역 경로 재계획을 실행하여 주행 경로가 길어진 것을 확인할 수 있다.

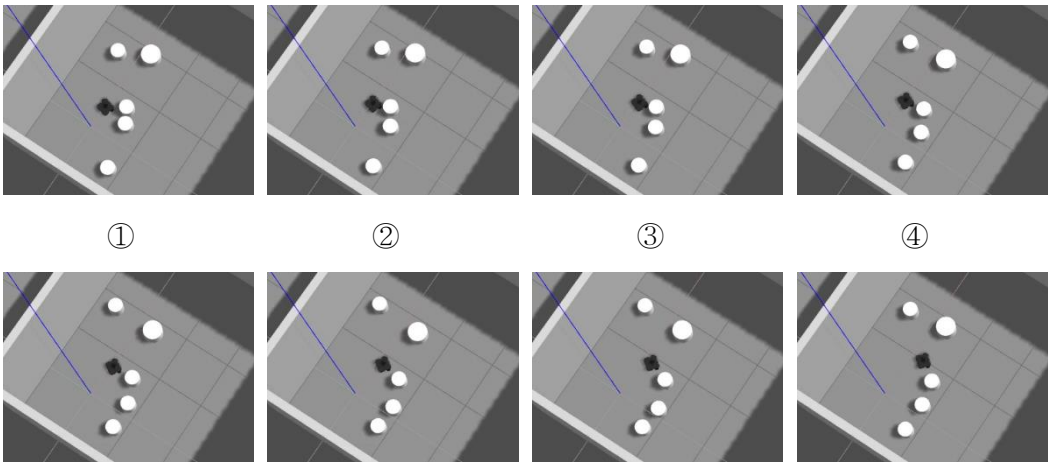


[그림 29] 5개 장애물 실험 시 DWA의 충돌 상황

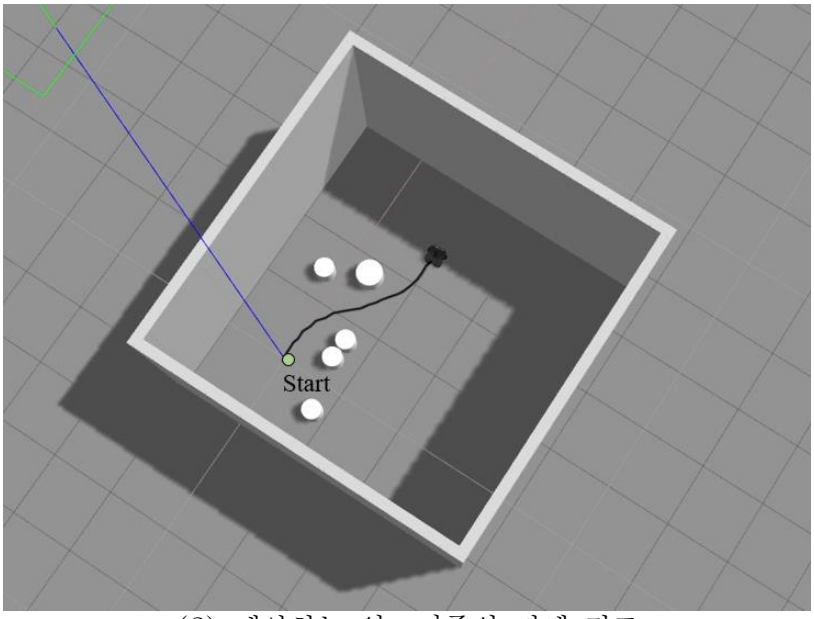


[그림 30] 5개 장애물 실험 시 DWA의 주행 경로

[그림 31]의 (1), (2) 항목은 각각 본 논문에서 제안하는 알고리즘의 충돌 회피 경로 생성과 주행 경로를 나타낸다. [그림 31]의 (1)에서는 근방의 동적 장애물을 감지할 경우 로봇이 적절한 충돌 회피 경로를 생성한다. ①, ②, ③ 단계에서 장애물의 움직임을 예측하고 (장애물은 왼쪽으로 이동), 로봇은 이후 ③, ④, ⑤ 단계에서 오른쪽으로 주행하여 충돌 상황을 피한다. [그림 31]의 (2)는 제안하는 알고리즘이 [그림 30]과 비교하여 효율적인 경로로 주행함을 보인다.



(1) 제안하는 알고리즘의 충돌 회피 동작



(2) 제안하는 알고리즘의 전체 경로

[그림 31] 5개 장애물 실험 시 제안하는 알고리즘의 충돌 회피 실험

동일한 환경에서 네비게이션 실험을 진행한 결과에 대한 요약[표 4]은 다음과 같다. 본 논문에서 제안하는 알고리즘은 충돌 회피 성공률이 40%로 DWA의 20% 보다 높다. 그리고 평균 주행 시간이 DWA 대비 약 2.6배 단축된다.

	Success rate for collision avoidance	Max time	Mean time
Original DWA	20%	87.0s	33.8s
Ours	40%	17.0s	12.8s

[표 4] 5개 장애물 실험 시 DWA와 제안하는 알고리즘의 성능 비교

B. 결과 해석

앞선 절에서의 실험 결과는 본 논문에서 제안하는 알고리즘이 동적 장애물이 등장하는 다양한 시나리오에서 기존의 DWA보다 충돌 회피에 더 높은 성공률을 보임을 알 수 있다. 그리고 장애물이 가까이 있어도 적절한 지역 경로를 찾고 전역 경로 재계획을 진행하지 않음으로써 목표 지점까지 더 효율적인 경로와 시간으로 이동함을 보인다. 이로써 충돌 예측 확률을 반영한 DWA가 동적 환경에서 충돌 회피를 위한 경로를 계획하고 반응형 동작을 생성함을 보였다.

하지만 제안하는 알고리즘 역시 동적 장애물과의 충돌을 회피하지 못하는 경우가 존재한다. 로봇의 동역학을 고려할 때, 급변하는 속도를 생성하기가 어렵다는 점은 하나의 충돌의 원인이 된다. 그리고 속도를 택하는 비용 함수에서 장애물과의 충돌 회피 확률 이외의 다른 요소들의 비용 값이 클 때 급히 장애물을 회피하는 속도를 최적으로 계산하지 못한다는 점 역시 충돌의 원인이 된다. 이 한계점은 충돌 예측 시간을 활용하여, 충돌이 임박한 경우 회피 확률을 우선으로 판단하는 방법을 통해 개선할 수 있을 것으로 기대한다.

Ⅵ. 결론 및 향후 계획

본 논문에서는 동적 환경에서의 충돌 예측 확률을 이용한 지역 충돌 회피 알고리즘을 제안하였다. 제안한 알고리즘으로 기존 지역 경로 계획자 DWA를 개선하여 충돌 예측 확률 기반의 DWA를 제시하였다. 이를 위해 근방 동적 장애물의 궤적을 예측하여 충돌 예측 시간을 계산하였고, 충돌 예측 시간에 기반하여 충돌 회피 확률 모델을 제시하였다. 충돌 회피 확률은 속도 샘플을 최적화하는 목적 함수에 반영하여, 로봇이 충돌 가능성이 높은 속도는 최대한 택하지 않도록 비용 함수를 계산하였다.

실험은 Gazebo 시뮬레이터 상에서 2개, 3개, 5개의 동적 장애물을 배치하여 네비게이션을 실행하는 것으로 진행하였다. 실험 결과는 기존 DWA에 제안하는 알고리즘으로 생성한 충돌 확률 모델을 적용했을 때, 적용하지 않은 DWA보다 동적 환경에서 좀 더 유연한 충돌 회피 경로와 효율적인 주행 경로를 계획함을 보였다.

하지만 제안하는 알고리즘은 가정 및 제약 사항에 따른 한계점을 보인다. 먼저, 장애물의 등속운동을 가정하지만, 장애물에 급격한 동작 변화가 있을 때는 정확한 예측이 어려운 경우가 존재한다. 또한, 로봇의 정확한 센싱이나, 장애물의 형태에 대한 가정 역시 실제 로봇 적용 시 정확한 계산을 어렵게 한다.

향후 연구를 통하여 본 논문에서 가정한 제약 사항들을 더 일반적인 상황에 적용할 수 있도록 시스템을 개선하고자 한다. 먼저 다양한 형태의 장애물의 위치를 추정할 수 있도록 각 장애물에 원형 반경을 설정할 예정이다. 그리고 실제 로봇에서 발생하는 로봇 센서의 측정 오차와 휠 오도메트리의 오차를 보정하여, 보다 정확한 장애물 감지 및 계산을 할 수 있도록 개선할 계획이다. 마지막으로 더 많은

수의 장애물과 충돌을 회피하도록 확장하여, 향후 연구를 통해 군중 환경이나 분산형 다중 로봇 환경에서도 로봇의 효율적인 충돌 회피 경로 생성을 기대한다.

참 고 문 헌

[1] ROS global planner. http://wiki.ros.org/global_planner

[2] ROS base local planner. http://wiki.ros.org/base_local_planner

[3] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23-33. 1997.

[4] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research (IJRR),* 17:760-772, 1998.

[5] Velocity Obstacles. https://en.wikipedia.org/wiki/Velocity_obstacle

[6] J. van den Berg, M. C. Lin, and D. Manocha, Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation. *IEEE International Conference on Robotics and Automation (ICRA),* 2008.

[7] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. *Robotics Research*, 3–19, 2011.

[8] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. *IEEE/RSJ International Conference on Intelligent Robots Systems (IROS)*. pp. 4584–4589, 2010.

[9] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots. *International Symposium on Distributed Autonomous Robotic Systems (DARS),* Nov. 2010.

[10] C. Rösmann, F. Hoffmann, and T. Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and*

Autonomous Systems (RAS). 88, 142–153. 2017.

[11] O. Brock and O. Khatib. High-speed Navigation using the Global Dynamic Window Approach. *IEEE International Conference on Robotics and Automation (ICRA),* 1999.

[12] P. Ogren and N. E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, April 2005.

[13] D. Teso-Fz-Betoño. E. Zulueta. U. Fernandez-Gamiz. A. Saenz-Aguirre. and R. Martinez. Predictive Dynamic Window Approach Development with Artificial Neural Fuzzy Inference Improvement. *Electronics*, 8, 935, 2019.

[14] M. Missura and M. Bennewitz. Predictive Collision Avoidance for the Dynamic Window Approach. *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[15] Adaptive neuro fuzzy inference system. https://en.wikipedia.org/wiki/Adaptive_neuro_fuzzy_inference_system

[16] C. Koike, C. Pradalier, P. Bessiere, and E. Mazer. Proscriptive bayesian programming application for collision avoidance. *IEEE/RSJ International Conference on Intelligent Robots Systems (IROS)*. pp. 394-399. 2003.

[17] G. Alenya, A. Nègre, and J. Crowley. Time To Contact for Obstacle Avoidance. *European Conference on Mobile Robotics*. Sep 2009.

[18] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna and D. Manocha. PRVO: Probabilistic Reciprocal Velocity Obstacle for multi robot navigation under uncertainty. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 1089-1096. 2017.

[19] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen. Collision avoidance under bounded localization uncertainty. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 1192-1198, 2012.

[20] S. Patil, J. van den Berg and R. Alterovitz. Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. *IEEE International Conference on Robotics and Automation (ICRA),* pp. 3238-3244, 2012.

[21] J. Lunenburg, R. Molengraft, and M. Steinbuch. A representation method based on the probability of collision for safe robot navigation in domestic environments. *Autonomous Robots* 42: 601. 2018.

[22] J. Eggert. Predictive risk estimation for intelligent ADAS functions. *International IEEE Conference on Intelligent Transportation Systems (ITSC).* pp. 711-718, 2014.

[23] J. Ward, G. Agamennoni, S. Worrall and E. Nebot. Vehicle collision probability calculation for general traffic scenarios under uncertainty. *IEEE Intelligent Vehicles Symposium Proceedings*, pp. 986-992, 2014.

[24] ROS Navigation. 2018.
http://wiki.ros.org/navigation/Tutorials/RobotSetup

[25] Robot Operating System. 2019. https://www.ros.org/about-ros/

[26] Turtlebot3. 2019.
http://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_01/

[27] Occupancy grid mapping.
https://en.wikipedia.org/wiki/Occupancy_grid_mapping

[28] S. Jeong, J. Gim, C. Ahn. V2V Based Vehicle Detection and Collision Avoidance Algorithm. *Transactions of Korean Society of Automotive Engineers*. Vol. 26, No. 6, 773-782. 2018.

ABSTRACT

Local Collision Avoidance Algorithm in Dynamic Environment using Collision Probability

Kim, Jee-Seon

Major in Computer Science and Engineering

The Graduate School of Ewha Womans University

The key functions of the autonomous mobile robot (AMR) navigation are creating an efficient path and avoiding collision. To avoid collision, local planners generate collision-free path for a short period time and repeat this process until reaching goal position. Local planners should also make an immediate, and efficient collision-free path in dynamic environments. However, many local planners still plan their path considering only static obstacles. Besides, local planners still assume that the robot has exact information about obstacles' velocity and position, which makes limitations when applied to real robot.

Therefore, in this dissertation, a dynamic obstacle avoidance algorithm based on the collision probability is proposed. The proposed algorithm complements DWA (Dynamic Window Approach), which is mainly used as a local planner in ROS (Robot Operating System), to have avoidance function for dynamic environments. The proposed approach firstly predicts the trajectory of nearby obstacles by using a 2D-LiDAR sensor, then calculates time-to-collision for each obstacle. The collision avoidance probability based on time-to-collision is then calculated and added in the cost function of DWA to find a path with the highest value. This method avoids robot from driving at a speed which is more likely to cause a collision.

We tested the performance of the proposed algorithm by using Gazebo simulator with dynamic environments and executing navigation for Turtlebot3. We compared collision-free movement, path, and total time of DWA algorithm and proposed algorithm. The experimental results show that the proposed algorithm creates flexible collision-free path in a dynamic environment than the existing DWA algorithm and drives in a distance-efficient and time-efficient manner.