



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

강화학습을 이용한 모바일 로봇의 국지적 동적 장애물 회피

Local Dynamic Obstacle Avoidance of Mobile Robot
using Reinforcement Learning

指導教授 양기훈

이 論文을 碩士學位 請求論文으로 提出함

2020 年 02 月

科學技術聯合大學院大學校
생산기술(로봇공학) 專攻

김 진 석



金 珍石 의 工學部
碩士學位論文 認准함

2020年 02月

委員長 배 지 훈 印

委 員 양 기 훈 印

委 員 안 범 모 印

科學技術聯合大學院大學校



사사(謝辭)

2017년 12월에 처음 연구실에 들어갔던 기억을 지금도 잊을 수가 없습니다. 그때로부터 벌써 2년이라는 시간이 흘렀다는 게 신기하게만 느껴집니다. 지금 이렇게 돌이켜보니 석사 졸업을 무사하게 준비하기까지 정말 많은 도움의 손길과 격려가 있었습니다.

하고 싶은 연구 분야를 할 수 있도록 모바일 로봇을 공부하게 해주신 저의 지도교수님이신 양기훈 박사님, 매 발표와 준비를 할 때 많은 조언을 해주신 안범모 박사님, 학문적으로 많이 막힐 때 같이 고민해주시고 커피 취미를 같이 즐겨주신 장인훈 박사님, 이동 로봇의 경로계획법에 대해 지도해주신 윤한얼 박사님 그리고 같은 그룹으로 연구의 방향과 대학원 생활의 도움을 주신 표동범 박사님, 고광은 박사님, 정우석 박사님, 강재현 박사님께 감사드립니다.

연구실 생활을 할 때 적응하는데 많은 도움을 주시고 행복한 분위기를 만들어준 지웅씨, 재우씨, 재형씨, 익수씨 그리고 현조씨 덕분에 랩실 생활을 재미있고 보람차게 보냈습니다. 종종 모여서 같이 놀기도 하고 운동도 하였고 제가 만들어온 간식이랑 커피를 맛있게 먹으면서 정말 화목한 분위기를 느끼며 지냈습니다. 또 다른 연구실이었지만 같은 연구실 사람처럼 느껴졌던 운재, 준우형, 동우형, 조금 짧은 기간이었지만 우영씨, 현지씨한테도 많은 것들을 받았습시다. 그리고 지금은 여기 계시지는 않지만 제가 연구하고 공부하는데 있어서 정말 많은 도움을 주신 규상씨, 석우씨, 경환씨께 감사드립니다.

마지막으로 저를 항상 응원해주신 가족들에게 감사드립니다.

본 논문은 주행하는 로봇 보다 빠른 동적 장애물 회피를 하기 위해 강화학습과 국지적 회피 방법을 결합한 동적 장애물 회피 알고리즘을 제안한다.

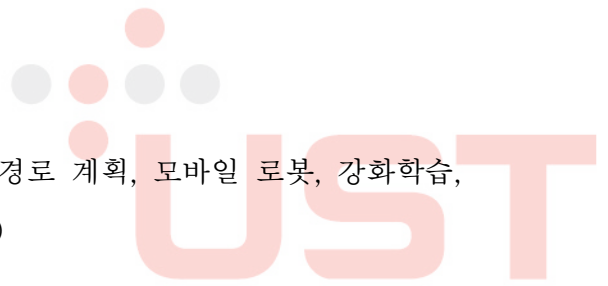
기존의 장애물 회피 방법을 통해 제어되는 로봇의 속도에 추가로 보조 제어를 한다면 동적 장애물 회피가 가능하다고 가정하였다.

거리 센서의 정보만을 가지고 동적 장애물에 대한 정보를 파악하는 알고리즘을 구성하였고 이를 통해서 학습 상태(State)를 정의하였다. 회피를 위한 행동(Action)은 경로 계획법에서 최종적으로 로봇에게 주행을 명령하는 속도 값에 영향을 주는 매개변수를 제어한다. 이 제어를 통해 움직이는 장애물을 회피 할 수 있는 속도를 구할 수 있다.

학습은 총 5000번 진행되었고 다양한 상황에 대처하기 위해서 동적 장애물과 로봇의 조우를 다르게 구성하였다. 실험은 기존 방법과 제안된 방법을 장애물의 속도를 다르게 하는 것과 움직임 패턴을 변경하여 각각 100번씩 진행하였다.

기존의 방법(DWA)만 사용한 주행과 비교하여 회피 성공률이 높은 것을 실험을 통해 확인하였다. 또한 학습한 동적 장애물의 속도가 아닌 다른 속도와 다른 움직임을 보이는 환경에서도 제안한 방법이 더 좋은 회피 능력을 보여주었다.

이를 통해서 동적 장애물 회피에 대해서 추가적인 보조 속도제어를 통해서 장애물 회피 성능을 향상 시킬 수 있다는 것을 증명하였다.



주요단어(Keyword) : 충돌 회피, 경로 계획, 모바일 로봇, 강화학습,
dynamic window approach(DWA)



ABSTRACT

This thesis proposes a dynamic obstacle avoidance algorithms that combines reinforcement learning and local avoidance for faster dynamic obstacle avoidance than a driving robot.

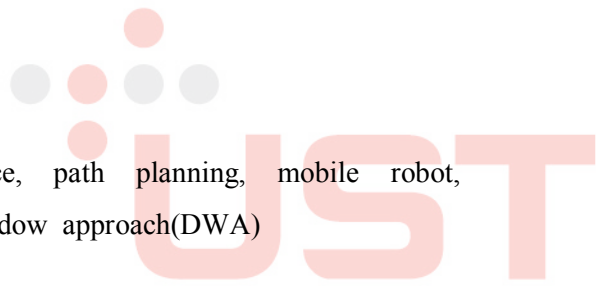
It is assumed that dynamic obstacle avoidance is possible if auxiliary control is performed in addition to the speed of the robot controlled by the existing obstacle avoidance method.

The algorithm that grasps data about dynamic obstacles using only distance sensor information is composed and learning state is defined through this. The Action for avoidance control parameters that affects the speed value that finally commands the robot to travel in path planning. This control allows to find the speed at which enable to avoid moving obstacle.

The learning was conducted 5000 times in total, and the encounters of robot and dynamic obstacle were constructed differently to cope with various situations. In the experiment, the proposed method and the Dynamic Window Approach (DWA) were performed 100 times each by changing the speed of obstacles and the movement pattern.

The experiments confirmed that the success rate of evasion using proposed method was higher than that of the DWA. In addition, the proposed method showed better avoidance in the environment where the speed is different from the speed of the dynamic obstacle.

Through this, we proved that the obstacle avoidance performance can be improved through additional auxiliary speed control for dynamic obstacle avoidance.

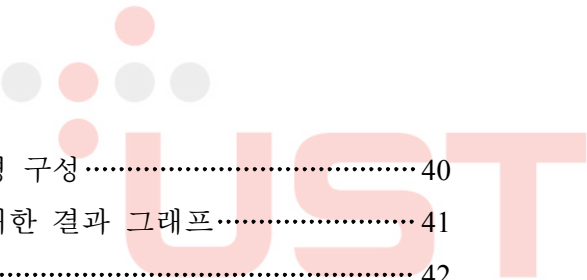


Key words : collision avoidance, path planning, mobile robot, reinforcement learning, dynamic window approach(DWA)

I. 서 론	1
1. 연구배경	1
2. 연구목표	6
3. 논문구성	7
II. 관련 배경이론	8
1. Reinforcement Learning	8
가. Markov Decision Process(MDP)	9
나. Q-learning	10
다. Deep Q Network(DQN)	11
2. Dynamic Window Approach(DWA)	13
III. 학습 에이전트 정의	17
1. Deep Q Network 설정	18
2. 전체 알고리즘 구성	26

IV. 실험 및 분석	27
1. 실험 환경 구성	27
2. Navigation 구조 설계	30
3. 학습 알고리즘 설계와 학습	33
4. 가상 환경 실험	38
가. 회전 운동 장애물 환경 실험	38
나. 직진 운동 장애물 환경 실험	40
5. 현실 세계 실험	42
가. 실험 환경 구성	42
나. 실험 및 결과	44
V. 결 론	47
참고문헌	49

<그림 1> 강화 학습의 구성	8
<그림 2> DQN Algorithm	12
<그림 3> 로봇의 환경과 목적 위치	13
<그림 4> 속도 영역과 Dynamic Window	14
<그림 5> 속도, 거리, 각도 값을 합친 그래프	15
<그림 6> 식 (6)을 적용한 그래프	15
<그림 7> 모바일 로봇의 환경	18
<그림 8> 장애물 정보의 추적	20
<그림 9> 행동에 따른 속도 값 입력	21
<그림 10> Dynamic Window에서의 로봇의 행동에 따른 속도 위치	22
<그림 11 > 정의된 보상	25
<그림 12> 전체 알고리즘 구성도	26
<그림 13> Gazebo7.0로 구현한 가상환경	28
<그림 14> TurtleBot3 burger	29
<그림 15> Sick Tim 55x 감지 영역	29
<그림 16> 가상 실험환경에 대한 지도 정보 [회색영역 : 확인되지 않은 미지영역, 흰색영역 : 로봇이 이동 가능한 자유영역, 흑색영역 : 확인되지 않은 미지 영역]	30
<그림 17> ROS Navigation Stack 모식도	31
<그림 18> Costmap에서 inflation 정의	32
<그림 19> 설계된 학습 알고리즘	33
<그림 20> 장애물과 로봇 그리고 목표 위치 구성	35
<그림 21> 에피소드 진행에 따른 장애물 회피 성공률	35
<그림 22> 에피소드 진행에 따른 누적 보상 그래프	37
<그림 23> 회전 운동하는 장애물 환경 구성	38
<그림 24> 회전하는 장애물 실험에 대한 결과 그래프	39



<그림 25> 직진 운동하는 장애물 환경 구성	40
<그림 26> 직진하는 장애물 실험에 대한 결과 그래프	41
<그림 27> 실제 주행 실험 환경	42
<그림 28> 주행에 사용된 로봇 Turtlebot3 burger	42
<그림 29> 실제 주행한 환경의 Grid map	43
<그림 30> 정적 장애물 환경	45
<그림 31> 동적 장애물 환경	45
<그림 32> 동적 장애물 환경	45

<표 1> 장애물의 개수, 거리, 속도, 각도를 인식하는 알고리즘	19
<표 2> 행동에 따른 로봇의 속도 제어 알고리즘	23
<표 3> DQN parameter	34
<표 4> 회전하는 장애물 속도에 대한 결과[성공률]	39
<표 5> 직진하는 장애물 속도에 대한 결과[성공률]	41

I. 서론

1. 연구 배경

모바일 로봇에서 자율주행기술이란, 직접 운전이나 움직임을 제어하지 않고, 사용자가 원하는 목적지까지 로봇 스스로 주변 환경을 인지하고 적절한 상황판단을 하면서 주행 할 수 있는 기술이다. 자율주행기술을 이루기 위해 필요한 대표적인 기술들로 분류하면 첫 번째로 주변 환경을 인식하는 기술, 두 번째로 자신의 위치를 파악하는 것, 그리고 세 번째는 앞서 이야기한 환경 정보와 자기 위치 정보를 바탕으로 경로 계획을 하고 장애물을 회피하는 기술이다. 이 중에서 장애물 회피 기술은 주행 안전성에 결정적인 요인을 결정짓는 중요한 분야이다. 모바일 로봇이 실제 주행은 정적인 장애물들과 동적인 장애물들이 포함된 환경에 노출된다. 장애물 회피 문제에서 비교적 회피에 용이한 정적 장애물과 달리 동적 장애물 같은 경우는 이동하는 특성을 갖고 있기 때문에 이동 방향, 속력, 장애물의 크기 등 고려할 요인이 더 크다[1,2]. 또한, 동적 장애물의 회피만을 고려하여 경로 생성한 부분에 사전에 알지 못하였던 정적 장애물이 있을 경우에는 충돌 위험에 놓이게 된다. 따라서 장애물 회피를 하기 위해서는 동적 장애물과 정적 장애물을 같이 구해야한다. 그 과정에 필요한 것이 Local Planner 이다.

Local Planner는 Path Planning에서 분류되는 Global Planner과 다르게 출발 위치부터 목적 위치까지 도달하는 최적의 경로를 생성하는 것과 조금 다르게 인근에 존재하는 장애물에 대한 회피 기술과 주행하는 로봇을 따르는 경로계획기술에 대한 것을 포함한다.

Local Planner에서 장애물 회피에 대한 기술이 중요하므로 관련된 연구가 많이 있어왔다[3]. Local Planner 알고리즘에 대표적인 연구들 중에서 전통적으로 사용되는 것들로 DWA(Dynamic Window Approach)[4,5], VFH(Vector Field Histogram)[6,7], RRT(Rapidly exploring Random Tree)[8,9], VO(Velocity Obstacle)[10,11], ARF(Artificial Potential Field)[12,13] 등이 있다.

[4]에서 처음 소개된 DWA는 정적 장애물에 대해서 우수한 회피 성능을 가지고 있으며 저속 장애물에 대한 회피도 가능하다. 또한 로봇의 속도 모델을 고려하여 장애물 회피하여 목적지까지 빠른 주행을 가능케 한다. 하지만 DWA는 동적 장애물에 속력이 빠르거나 동적 장애물과 수직 방향으로 만나게 되는 경우에는 장애물을 회피하는 여유 이동경로의 손실 문제로 충돌이 일으킬 수 있다. [5]에서는 DWA의 문제를 개선하였다. 로봇의 크기를 고려하여 회피를 위한 여유 이동경로의 문제를 개선시켰고 전체적인 경로 주행 동선의 효율을 상승시켰다.

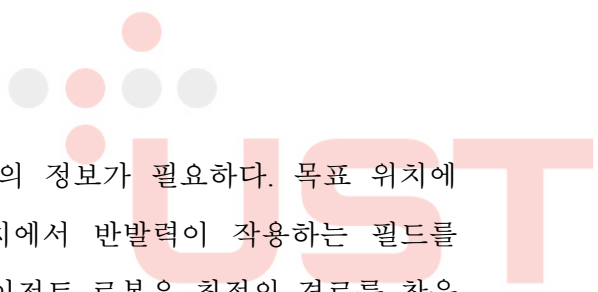
VFH[6]는 장애물과 로봇 각각에 Certainty 값을 만든다. 그리고 그 값으로 로봇이 목표점으로 가는데 장애물과 부딪치지 않고 주행할 수 있는 가능 히스토그램 영역과 불가능한 히스토그램 영역을 나누어 로봇을 가능 히스토그램 영역으로 주행하게 한다. 이 알고리즘은 움직이지 않는 정적 장애물 환경에서 높은 신뢰도의 회피 성능을 가지지만 실제 현실은 동적인 환경에서는 충돌을 일으킨다는 문제가 있다. 이러한 문제를 해결하기 위해 나온 알고리즘이 E-VFH[7]이다. E-VFH는 동적 장애물을 타원형의 모델로 가정한다. 동적 장애물의 속도와 장애물의 크기를 바탕으로 타원형의 모델 정보를 생성하여 로봇에게 전달한다. 이를 통해 장애물과의 충돌을 회피할 수 있게 한다. 하지만 E-VFH는 VFH에서 동적 장애물의 모

텔링 정보만을 수정하였기 때문에 로봇의 이동시간을 보장할 수 없다.

RRT[8]는 Voronoi region에서 착안한 빠른 공간 탐색 방법을 이용하며 계산량이 적고, 빠른 시간 안에 도착점까지의 경로를 생성해 낼 수 있다는 장점을 가지고 있다. RRT를 이용한 경로 계획을 실행하며 흔들림(jittering)을 가진 경로가 생성되는데, 이를 최적화시켜 장애물이 없는 두 지점을 직선으로 연결하는 방법을 사용한다. 하지만 이러한 방법은 실제 로봇의 동역학을 만족하지 못하며 복잡한 환경에서 새 경로를 생성하는 시간이 걸리는 단점이 존재한다. Extended RRT[9]에서 알려지지 않은 임의의 장애물이 존재하는 경우에도 빠르게 장애물을 회피하는 경로를 생성하는 방법을 고안하였다. 이 방법은 기존 RRT의 경로를 더 최적화 시켜 탐색시간을 줄이는 방법과 여러 장애물 및 로봇에 대한 노드를 공유하여 다중 장애물 회피 경로 생성에 최적화를 하였다.

VO[10] 기반 방법은 충돌을 피하기 위해 장애물의 현재 속도 정보를 사용한다. 이 방법에서 장애물들은 메인 로봇의 속도 공간 안에 표시되며, 이 정보를 사용하여 회피가 가능한 VO 영역을 계산할 수 있다. 그러나 로봇이 영역 밖의 속도를 선택하는 경우에는 로봇이 진동을 일으키는 모션을 발생시키는 문제가 있다. 이러한 한계점을 극복하기 위해서 RVO[11]가 제안되었다. 두 로봇이 서로 마주쳤을 때 각 로봇은 충돌 방지에 대한 책임을 절반씩 부여받는다. 이 방법을 사용하면 직접적인 통신 없이 장애물과 다른 로봇의 충돌을 피할 수 있다. 하지만 두 로봇 혹은 장애물과 로봇 사이의 어느 방향으로 움직일지에 대한 상호 동의 방식이 실패할 경우에는 충돌 가능성이 존재한다.

ARF[12, 13] 방법은 잠재적 필드를 생성하기 위해서 시작 위치,

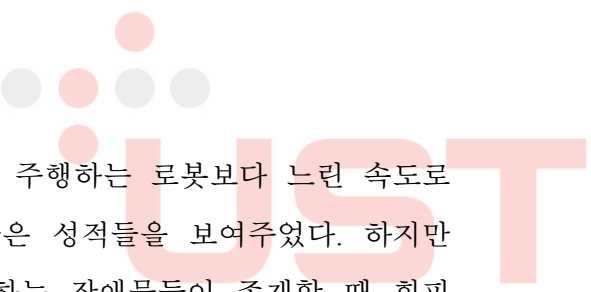


목표 위치, 인력, 반발력 등 일련의 정보가 필요하다. 목표 위치에 인력이 발생하고 장애물들의 위치에서 반발력이 작용하는 필드를 생성한다. 생성된 필드를 통해 에이전트 로봇은 최적의 경로를 찾을 수 있다. 하지만 이 방법은 로봇이 모든 장애물의 위치 및 속도와 같은 전체 환경에 대한 사전 지식을 갖고 있다는 것을 전제로 한다는 한계점을 지니고 있다. 이 한계로 인해 동적인 환경에서 필드를 지속적으로 재생성하는 문제로 실시간 주행이 어렵다.

또한 인공지능 기법을 활용한 Planner 연구도 계속 있었는데 그 중에서 강화학습을 이용한 방법이 최근 활발하게 연구되었다[14]. 한 연구에서는 동적인 환경에서 지도학습과 강화학습을 사용하여 특정 목표 지점까지 자율 주행하였다[15]. 이 논문에서 저자는 두 알고리즘을 통하여 골 위치까지 도달하며 장애물 회피와 경로 최적화 성능을 향상시켰다. 하지만 이 방법은 실제 로봇이 아닌 가상환경에서 경우의 수를 제한시켜 진행하였기 때문에 실제 환경 같은 복잡한 환경에서는 한계가 존재한다. 강화학습 기반에서 실제 주행을 시도한 방법으로 Q-learning과 Neural network 사용하여 장애물 회피를 한 알고리즘[16]이 있다. 이 알고리즘을 통해서 local minima를 회피하며 조정 가능한 속도로 모바일 로봇의 효과적인 경로 주행이 가능하며 또한 해결 방법의 하이브리드 특성으로 인해 Q-learning의 단점이 제거되기에 환경에 대한 사전 지식 없이 최적 경로 생성이 가능하다. 그러나 이 시스템을 유지하기 위해서 주변 환경의 모든 동적 매개 변수를 알아야한다는 단점이 존재한다.

위에 제안된 방법들을 분석하여 동적 환경에서 장애물 회피에 영향을 끼치는 주된 문제들은 다음과 같다.

- (1) 이동 속도가 빠른 동적 장애물 회피 기술
- (2) 로봇의 속도를 고려한 주행 제어



이전 연구들에서 로봇은 정적이고 주행하는 로봇보다 느린 속도로 움직이는 장애물들에 한해서는 좋은 성적들을 보여주었다. 하지만 더 빠르거나 비슷한 속도로 주행하는 장애물들이 존재할 때 회피 성능에 문제가 발생하는 것을 확인 할 수 있었고 로봇의 속도 같은 동역학적인 요소를 반영하지 못한 방법들은 주행 속도가 제한되는 문제가 발생하였다. 이러한 복합적인 문제들을 해결하기 위해서 본 논문에서는 기존의 전통적인 방법과 강화학습 방법을 결합하려 한다.

2. 연구 목표

본 논문에서는 속도가 빠른 동적 장애물 회피에 대한 문제를 해결하고자 한다. 이 문제를 해결하기 위한 방법으로 앞서 설명 하였던 DWA 방법에 인공지능을 활용한 보조 속도 제어 모듈을 결합하는 방법을 제안한다. DWA를 적용한 로봇은 저속 장애물에 대하여 회피가 가능하며 로봇이 주행하는 속도를 보장하는 알고리즘이다. 속도가 빠른 장애물과 직면하는 상황에서 회피 성공률을 보장하지는 못하지만 일정 부분의 회피를 위한 모션을 취한다. 이점을 착안하여 동적 장애물에 대하여 미리 파악하여 로봇의 주행의 영향을 주는 제어를 추가 한다면 회피가 가능 하다고 가정한다. 이 가정을 입증하기 위해서 본 논문에서는 주변 환경 상황에 대하여 인지 판단하여 장애물 회피를 보조하는 속도 제어 모듈을 설계하고 실험할 것이다.

모듈을 설계하기 위해서 최근 자율주행에 핵심적으로 많이 사용되고 있는 강화학습을 이용한다. 동적 장애물과 정적 장애물이 있는 가상환경에서 시뮬레이션을 진행하여 적절한 회피를 위한 로봇의 속도를 조절하여 회피를 위한 모션을 학습하고자 한다.

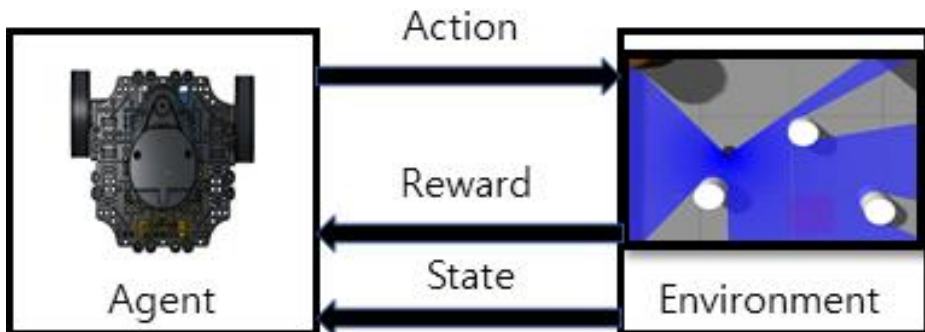
3. 논문 구성

본 논문에 구성은 다음과 같다. 2장에서는 본 연구에 핵심이 되는 강화학습과 DWA에 대하여 기술한다. 3장에서는 보조 속도 제어 모듈을 구성하기 위한 학습 에이전트를 정의한다. 4장에서는 가상환경에서 실험과 제안한 회피 알고리즘을 정략적 평가 방법을 통해 성능을 입증한다. 마지막으로 5장은 결론을 기술한다.

II. 관련 배경이론

1. Reinforcement Learning

강화학습(Reinforcement Learning)[17]은 기계 학습의 한 영역으로 행동 심리학에서 영향을 받았으며, 환경(Environment)과 에이전트 (Agent)의 각 상태(State)에 대해서 미지의 환경(Environment)에 대한 행동(Action)을 결정한다. 다른 기계 학습의 방법인 정답이 주어지는 지도학습이나 그저 주어진 데이터에 대해서 학습하는 비지도 학습과 다르게 보상 정책을 이용하여 학습을 한다. 그림 1와 같이 에이전트는 시행착오(Trial and Error)를 겪으며 긍정적인 보상(Reward)을 받도록 행동을 학습하게 된다.



<그림 1> 강화 학습의 구성

에이전트는 좋은 보상을 얻도록 새로운 행동을 하기 위한 탐험 (Explore)을 하거나 이전 좋은 보상을 얻었던 이전의 행동을 활용 (Exploit)을 할 수 있다. 탐험과 활용은 언제나 trade-off 관계를 유지하며 동시에 수행될 수 없는 구조로 되어있기 때문에 일정한 확률로 선정하거나 점차 활용을 많이 하는 형태로 수렴하는 방향으로 구성하여야한다.

가. Markov Decision Process(MDP)

강화학습의 기본인 Markov Decision Process(MDP)[18]는 확률적인 결과를 갖는 상황에서 의사결정을 모델링 하기 위한 프로세스로 강화 학습으로 문제를 해결하기 위해 문제를 정의하는 데 사용된다. MDP는 주로 다섯 가지 요소로 표현되는데 먼저 상태(State)는 에이전트(Agent)가 인식하는 자신의 상태를 의미한다. 두 번째로 행동(Action)은 한 상태에서 다른 상태로 옮겨가기 위해 행할 수 있는 행동(A)의 집합이다.

$$P_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a] \quad (1)$$

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a] \quad (2)$$

세 번째인 상태전이확률(State transition probability matrix)은 식 (1)을 통해 에이전트가 행동 a를 수행하여 현재 상태 s에서 다음 상태인 s'로 갈 확률을 정의한다. 식 (2)에서 에이전트는 보상(Reward)을 통해서 행동을 취하였을 때 보상을 받게 되는데 이때 0 ~ 1 사이의 값인 감가율(Discount factor)을 통해서 미래의 보상과 현재의 보상의 차이를 만든다. 그리고 정책(Policy)을 통해서 어떠한 상태 s에서 특정한 행동(A)을 할 확률을 구한다. 정책은 다음 식으로 정의된다.

$$\Pi(a|s) = P[A_t = a | S_t = s] \quad (3)$$

나. Q-learning

이전 MDP에서는 환경에 대한 모델의 특성인 상태 전이 및 보상 확률을 알아야 최적 정책을 구할 수 있는 한계가 있다. 이러한 환경에 대한 모델이 알려져 있지 않을 때 사용할 수 있는 대표적인 알려진 알고리즘은 큐러닝(Q-learning)[19]이다. 오프-폴리시 시간차 제어(Off-policy Temporal Difference Control)이라 불리는 큐러닝은 간단하며 시간차 방식에서 많이 사용된다. 제어 알고리즘에서 상태 s 에서 행동 a 를 수행한 결과에 대한 가치, 즉 상태-행동 가치를 사용한다. 큐러닝에서 다음 식에 값에 따라 큐값(특정 상태에 따른 어떤 행동에 따른 가치 값)을 업데이트 한다.

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \quad (4)$$

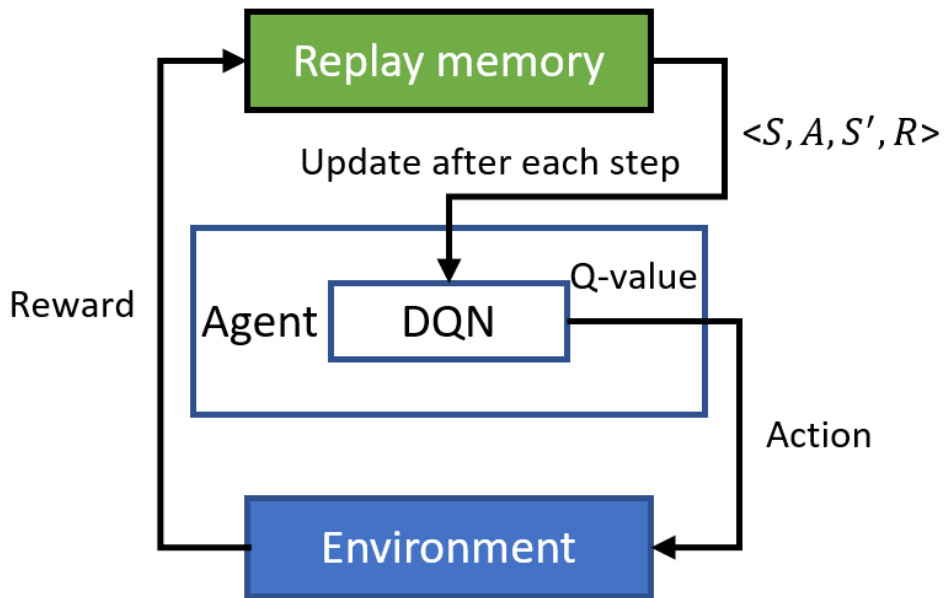
이를 통해서 상태-행동 가치($Q(s, a)$)를 구한다.

큐러닝의 장점은 매 스텝 학습이 가능하며 탐험정책을 수행 하면서도 최적의 정책을 학습 가능하다는 점이다. 행동을 선택할 때는 엡실론-그리드 정책을 사용하는데 이렇게 학습하면 수렴속도가 느려져서 학습속도가 느려지게 된다. 이 문제를 해결하기 위해서 엡실론 값을 시간에 따라 감소시키는 방법을 사용하며 큐값을 업데이트할 때는 최댓값을 가져온다.

다. Deep Q Network(DQN)

큐러닝에서 많은 상태와 각 상태마다 많은 행동을 시도하는 것은 매우 느리고 각 상태-행동 가치 값을 예측하는 것이 어렵다는 문제를 가지고 있다. 이러한 문제를 해결하기 위해서 구글의 딥 마인드사에서 Deep Q Network(DQN)[20] 알고리즘을 발표했다. DQN 알고리즘은 큐 네트워크(Q network)을 사용하여 큐 함수를 몇 가지 매개 변수 θ 로 근사($Q(s,a;\theta) \approx Q^*(s,a)$)하는 방법이다. DQN에서 신경망을 통해서 큐 함수의 값을 출력하기 때문에 최적의 값을 출력하도록 신경망인 큐 네트워크를 학습하는 것이 목표이다.

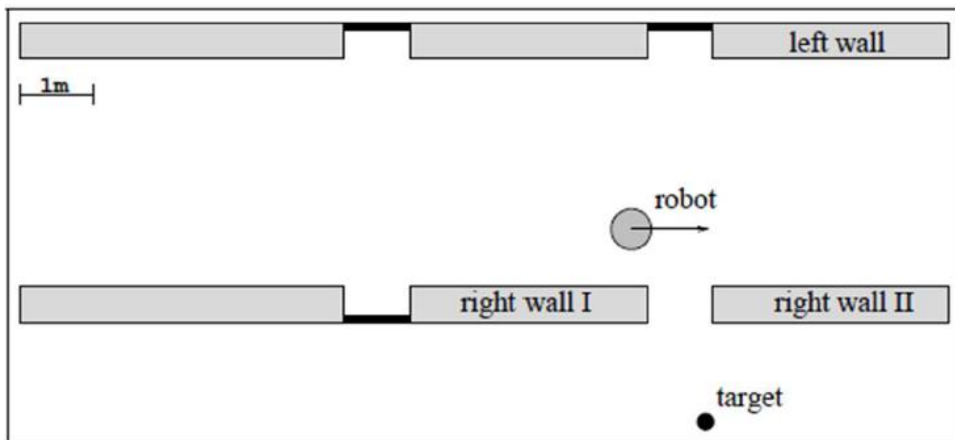
그림 2와 같이 DQN 알고리즘은 다음과 같이 구성되어있다. Convolution Neural Network(CNN)을 통해서 입력 정보를 받고 ReLu 함수를 활성화한 풀리 커넥티드 레이어(Fully Connected Layer)를 사용하여 출력 레이어의 노드수를 행동 개수와 같게 설정하여, 특정 상태에서 할 수 있는 모든 행동에 대한 큐값을 얻는다. 에이전트는 여기서 얻은 큐값 중 가장 큰 값을 가지는 행동을 취하여 보상을 받고 상태가 변경된다. 그리고 이들을 튜플 (S,A,S',R) 로 구성하여 Replay memory에 저장한다. 일정 에피소드가 진행된 후 Replay memory에서 무작위로 튜플을 추출하여 gradient descent를 계산하여 CNN을 갱신한다. 마지막으로 특정 단계마다, 실제 네트워크의 매개 변수 θ 를 타겟 네트워크의 매개변수 θ' 에 복사한다. 이 과정을 반복하며 학습을 진행한다.



<그림 2> DQN Algorithm

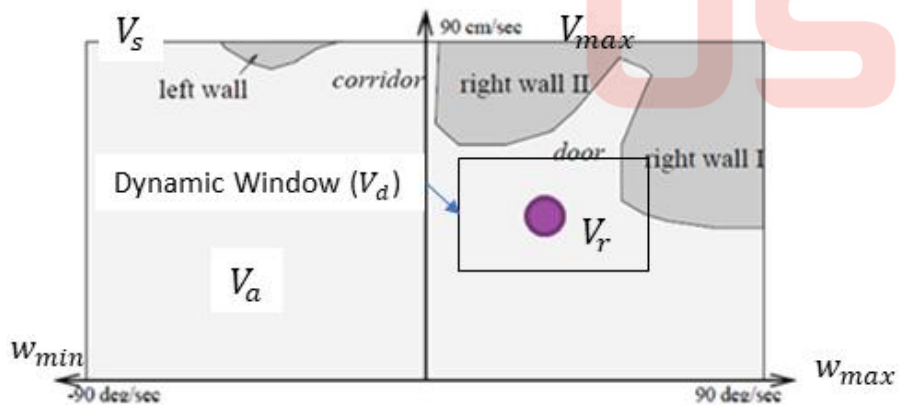
2. Dynamic Window Approach(DWA)

DWA는 로봇의 속도를 보장하며 빠른 속도로 이동하기 위해 만들어진 Local Planner이다. 장애물 회피 알고리즘 중에서 대표적인 알고리즘이며 ROS에서 제공하는 공식 패키지 중 하나인 Navigation에서 사용될 만큼 검증된 알고리즘이다. 모바일 로봇의 Rotational, Translational을 이용하여 빠르게 장애물을 회피하며 목표위치까지 도달할 수 있다.



<그림 3> 로봇의 환경과 목적 위치

로봇의 환경이 그림 3과 같이 구성되었을 때 DWA알고리즘은 로봇이 이동 가능한 속도영역을 2차원의 x 축(각속도), y 축(선속도)로 표현한다. 최종 목적지까지 이동하기 위해서 3가지 영역의 값을 이용한다.



<그림 4> 속도 영역과 Dynamic Window

먼저, V_s 속도영역(Velocity space)은 로봇의 이동 가능한 각속도와 선속도로 표현된다. 그림 4와 같이 로봇의 주변 환경을 거리 센서로 측정한 다음 속도 영역으로 표현하였다. 둘째, V_a 는 로봇 주변의 상황을 허용 가능 속도 영역으로 분류한다. 밝은 부분은 허용 가능한 속도 영역이고 회색 부분은 장애물 때문에 허용 불가능한 영역이다. 마지막으로 V_d (dynamic window)는 로봇의 다음 step 이내에 도달 할 수 있는 속도를 표현한 영역을 생성한다.

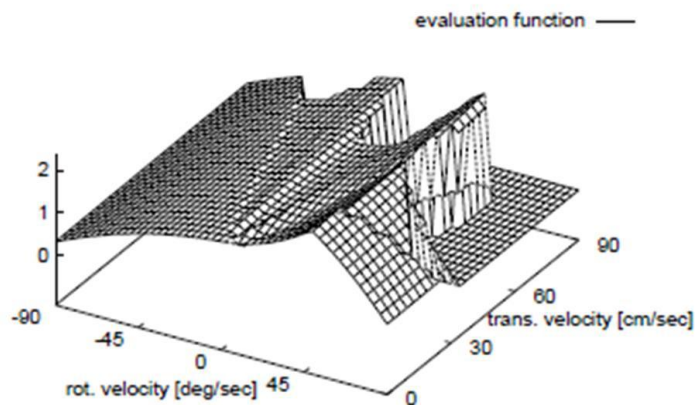
$$V_r = V_s \cap V_a \cap V_d \quad (5)$$

최종적으로 위 식 을 이용하여 로봇의 정보를 담고 있는 위치를 표현한다.

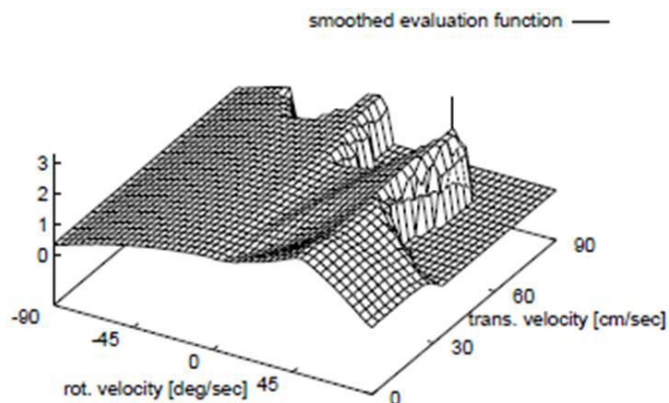
로봇의 속도 위치와 올바른 경로 안에서 로봇의 이동 시간이 적은 방향으로 향하게 하고 주행 속도를 빠른 속도로 움직이게 만드는 수식을 다음과 같이 표현한다.

$$G(v,w) = \sigma(\alpha \cdot \text{angle}(v,w) + \beta \cdot \text{dist}(v,w) + \gamma \cdot \text{velocity}(v,w)) \quad (6)$$

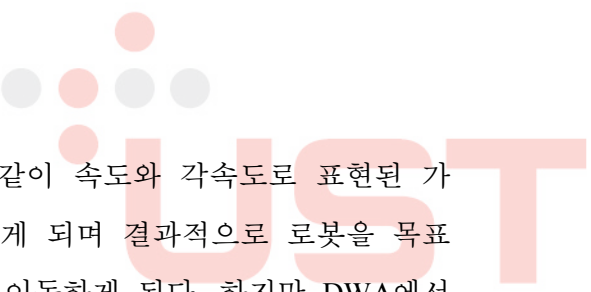
식 (6)에서 Angle함수는 로봇의 헤드와 목표점과의 방향의 각도 차이로 나타내고 Dist함수는 로봇이 이동 가능한 곡선에서 가장 가까운 장애물과의 교차되는 거리를 표현한 값으로 구성되어있다. 마지막으로 Velocity함수는 로봇의 경로에 해당하는 속도를 계산한 값이다. 이 속도, 거리, 각도의 값을 합치면 그림 5와 같이 나타낸다.



<그림 5> 속도, 거리, 각도 값을 합친 그래프



<그림 6> 식 (6)을 적용한 그래프



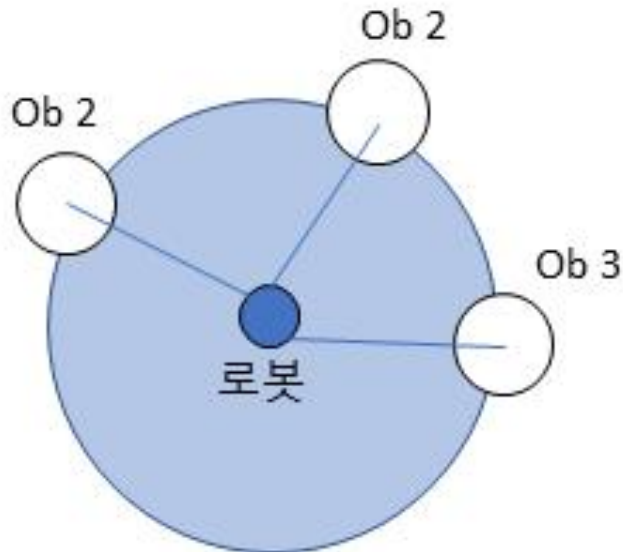
DWA는 최종적으로 그림 6과 같이 속도와 각속도로 표현된 가중치가 제일 높은 쪽으로 이동 하게 되며 결과적으로 로봇을 목표점까지 장애물을 회피하며 빠르게 이동하게 된다. 하지만 DWA에서 한계점은 주변 환경을 정적인 환경에서 측정하는 방식으로 접근한다는 문제이다. 동적 장애물이 이동함에 따라서 속도 공간에 함수값들의 신뢰도와 로봇이 이동 가능한 속도영역이 감소하게 된다. 이러한 하락세는 동적 장애물의 속도와 비례하기 때문에 속도가 빨라질수록 회피능력이 감소하게 된다.

III. 학습 에이전트 정의

본 연구의 핵심은 강화학습 기반의 DQN 알고리즘을 탑재한 보조 속도 제어 모듈을 설계하는 것이다. 연구의 목표는 모바일 로봇의 속도를 보장하면서 동적 장애물 회피 성능을 향상시키는 것이다. 본 장에서는 보조 모듈의 전반적인 구조와 각 세부 기능 및 핵심 알고리즘에 대해 기술 한다.

1. DQN 에이전트 설정

모바일 로봇의 시스템을 DQN 에이전트로 상태(state), 행동(action), 보상(reward)을 정의 하였다.



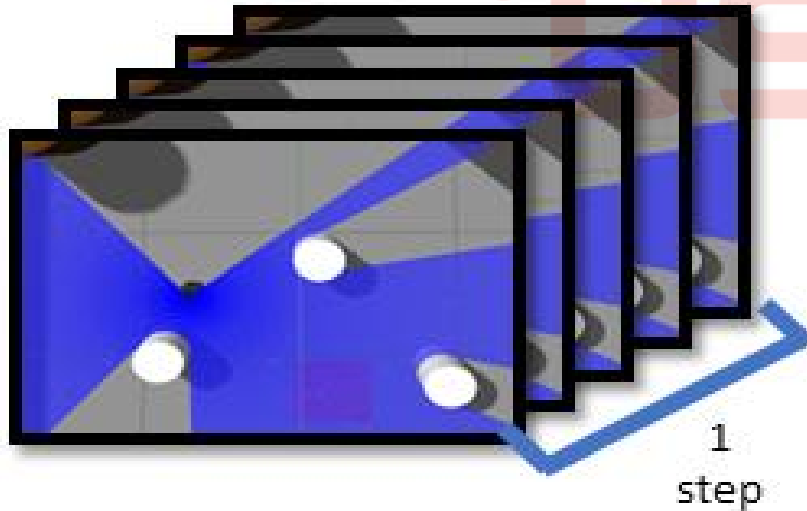
<그림 7> 모바일 로봇의 환경

먼저, 상태(State)를 정의하기 위해서 모바일 로봇의 동적 환경을 근접한 장애물의 속도, 위치 정보, 현재 로봇의 위치부터 목표 위치까지의 거리, 로봇의 현재 직선속도와 회전속도로 지정하였다. Laser Range Finder(LRF) 기반 센서를 사용한 장애물 검출 알고리즘을 만들어 장애물의 위치, 속도를 계산하였다. 장애물 검출 알고리즘의 구성은 일정 거리 이내의 접근한 장애물이 있게 되면 로봇의 헤딩 방향을 기준으로 장애물이 양 끝단의 각도정보를 구해서 장애물의 중심부터 로봇까지의 거리를 구한다. 여기서 그림 7과 같이 최대 3개까지의 장애물과 로봇 사이의 거리, 로봇의 헤딩을 기준으로 한 장애물과의 각도 정보를 구할 수 있다.

Algorithm Obstacle Detection Algorithm

1. Initialize all obstacle data[velocity1~3, distance1~3, angle1~3]
 2. Get scan_data using LRF
 3. **While** True **do**
 4. **For** i = 1, size(scan_data) **do**
 5. **If** scan_data[i] < safe_distance
 6. Get Obstacle Left degree[1~3]
 7. **Else**
 8. Get Obstacle Right degree[1~3]
 9. **End for**
 10. Get obstacle degree = (Left degree + Right degree) / 2 [1~3]
 11. Get obstacle distance = scan_data[obstacle degree] [1~3]
 12. Check time_d = current_time - old_time
 13. Get obstacle speed = (obstacle dist_old - obstacle dist) / time_d [1~3]
 14. **End while**
-

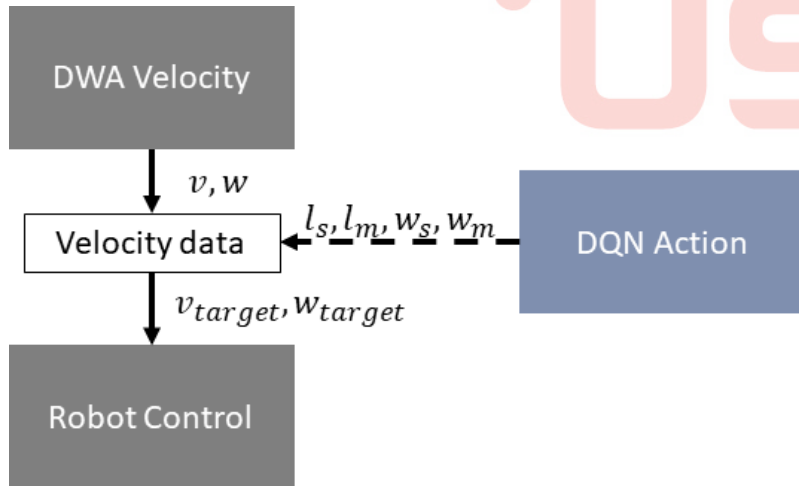
<표 1> 장애물의 개수, 거리, 속도, 각도를 인식하는 알고리즘



<그림 8> 장애물 정보의 추적

장애물의 속도를 구하기 위해서 이동평균필터를 이용하여 장애물과 로봇 사이의 거리의 정보를 안정화 시키고 이전 거리 값과 현재 거리 값을 비교하여 속도 정보를 구한다. 현재 로봇의 위치와 목적 위치 사이의 거리를 알기 위해서 ROS Navigation Stack의 포함되어 있는 Adaptive Monte Carlo Localization(AMCL)[21]을 사용하여 현재 맵을 기준으로 로봇의 위치 정보를 추정하여 구한다.

상태(State)의 구성을 정리하면 근접한 3개의 장애물의 속도, 위치정보(9개)와 현재 위치부터 목표 위치까지의 거리 (1개) 그리고 로봇의 현재 속도: 회전속도, 직선속도 (2개)로 총 12개로 구성되어있다. 상태의 정보는 그림 8과 같이 리플레이 메모리에 저장된다.

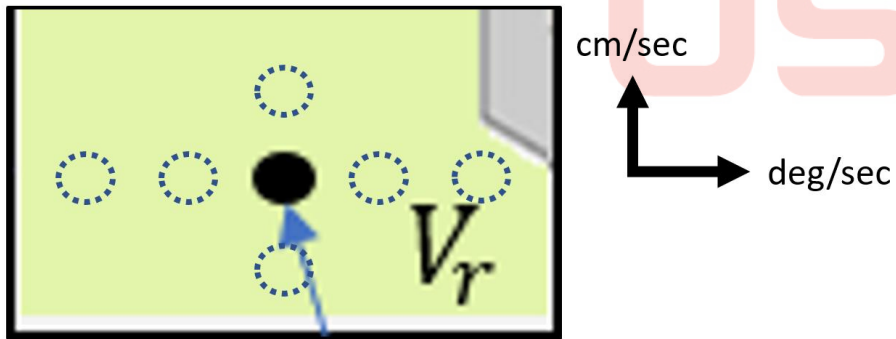


<그림 9> 행동에 따른 속도 값 입력

행동(Action)은 4가지의 직선속도와 회전속도의 증분 및 가중치 매개변수(l_s, l_m, w_s, w_m)를 조절하여 로봇의 속도에 영향을 주는 방식으로 구성되어있다. 로봇의 움직임을 구성하는 요소는 직선 속도와 회전속도로 구성되어있는데 DWA알고리즘에서 얻은 최적의 속도 값이 적용되어 움직이게 된다. 그림 9와 같이 DWA에서 나온 속도 명령 값에 식 (7)와 같이 업데이트 시키게 된다.

$$Velocity_{robot}(v_{target}, w_{target}) = (v_{dwa} * l_m + l_s, w_{dwa} * w_m + w_s) \quad (7)$$

위 식을 통한 속도 값은 로봇에게 제공되어 목표 속도를 지정하게 된다.



<그림 10> Dynamic Window에서의 로봇의 행동에 따른 속도 위치

행동(Action)의 구성을 정리하면 회전 속도 값에 일정 속도를 증감하는 것(4개), 기존 속도와 동일하게 주행(1개) 그리고 직진 속도를 증감(2개)하는 것으로 총 7개로 되어있다. 이 선택에 따라서 Dynamic Window 영역에서 선택되는 로봇의 속도 위치는 그림 10과 같이 구현된다.

Algorithm Velocity Control Algorithm

1. Initialize velocity parameters $[l_s, l_m, w_s, w_m]$ and output velocity
 2. **While** True **do**
 3. Get current DWA velocity [linear vel, angular vel]
 4. Get current velocity parameters = $[l_s, l_m, w_s, w_m]$
 5. Check data time period between DWA velocity and velocity parameters
 6. **If** time period is match
 7. Get robot_velocity = $(vel_{linear} \times l_m + l_s, vel_{angular} \times w_m + w_s)$
 8. **Else**
 9. Get robot_velocity = $(vel_{linear}, vel_{angular})$
 10. Publish robot_velocity
 11. **End while**
-

<표 2> 행동에 따른 로봇의 속도 제어 알고리즘

보상(Reward)을 통해서 큐 함수의 행동에 따른 상태의 가치 값이 바뀌게 된다. 행동을 한번 할 때 마다 보상을 주어지어야 하기 때문에 다음과 같이 보상을 정의하였다. 빠르게 목적 위치까지 이동하는 좋은 학습 모델을 설계하기 위해서 시간에 대한 부정적인 보상 r_t 정하고, 장애물을 회피하기 위해서 장애물과 거리가 가까워지는 만큼 부정적인 보상 r_{ob} 을 설정하였다. 모든 상태-행동 가치의 방향성은 좋은 보상을 받는 방향으로 향하기 때문에 목표 위치에 가까워질수록 긍정적인 보상 r_d 을 구현하였다. 그리고 장애물과 충돌할 때 큰 부정적인 보상을 목표위치에 도달하였을 때 큰 긍정적인 보상을 설계하였다. 다음 수식을 통해서 행동에 따른 보상에 대하여 정의한다. 여기서 l_{range} 의 값은 가장 가까운 장애물 거리이며 l_{safe} 의 값은 장애물 검출 가능 영역이고 $l_{collision}$ 은 장애물 충돌 거리이다.

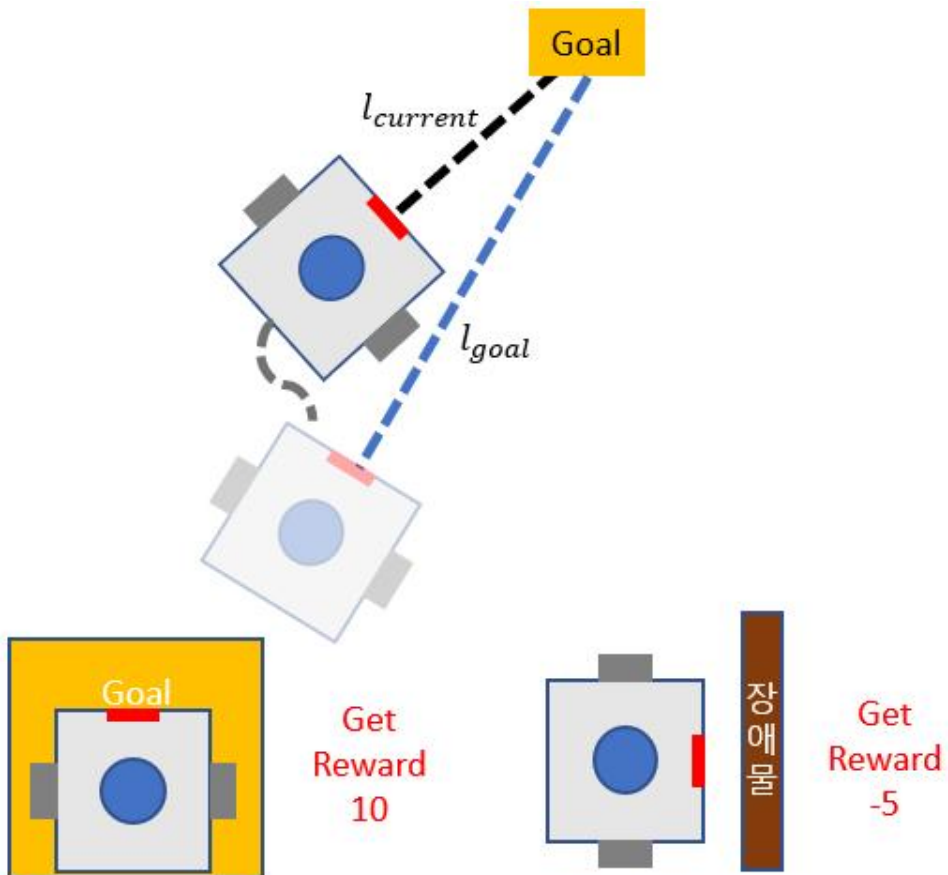
$$r_t = -0.2 \quad (8)$$

$$r_{ob} = \frac{(l_{range} - l_{safe})}{(l_{safe} - l_{collision})} * 0.4 \quad (9)$$

$$r_d = (1 - \frac{l_{current}}{l_{goal}}) \quad (10)$$

한 스텝에서 구할 수 있는 보상은 다음과 같이 정의된다.

$$reward = \begin{cases} if & obstacle\ collision & -5 \\ elseif & goal\ arrival & 10 \\ else & & r_t + r_{ob} + r_d \end{cases} \quad (11)$$



<그림 11 > 정의된 보상

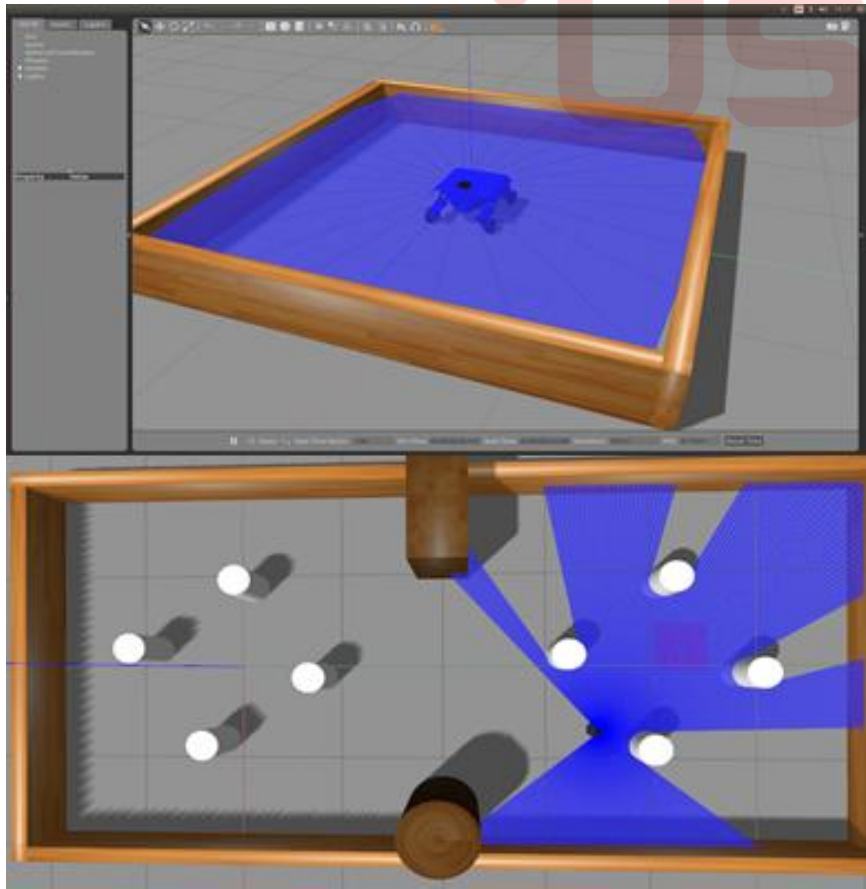
IV. 실험 및 분석

1. 실험 환경 구성

본 논문에 제안한 동적 장애물을 회피하기 위한 알고리즘 실험의 환경 구성은 다음과 같다. 가상환경에서 알고리즘의 성능 실험에서 센서 정보를 발생하고, 실제 로봇을 대체하는 시뮬레이션 로봇을 구현하였다.

가상환경은 물리엔진이 적용이 되어있는 Open Source Robotics Foundation의 Gazebo 7.0라는 시뮬레이션을 통해 구축하였다. Gazebo는 마찰력, 중력 및 기타 힘에 대하여 모델링이 가능하고 시간 가속을 통해서 실제 물리적인 환경에서 보다 빠르게 시나리오를 검증하고 실험을 할 수 있다. 또한 실 환경에서 같이 센서로 측정할 때 생기는 노이즈를 구현할 수 있기 때문에 시뮬레이션 환경과 실제 환경에 알고리즘이 구현되는 것이 비슷해진다는 장점을 가지고 있다. 이를 통해서 움직이는 동적인 환경과 센서를 구현하여 실험을 진행하였다.

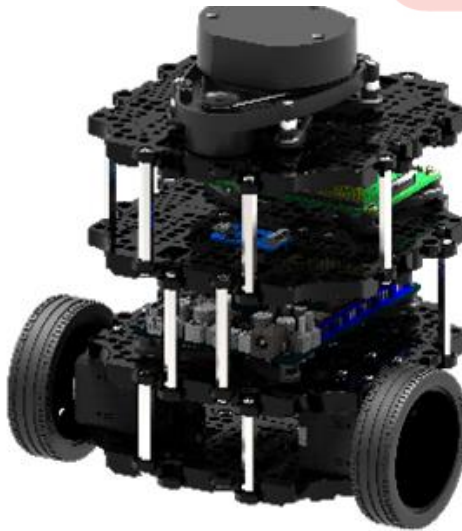
가상환경에서 구현된 장애물의 형태는 그림 13과 같이 원기둥 형태로 반지름이 0.15m, 높이가 0.5m의 크기를 가지고 있다. 전체 주행 공간의 넓이는 가로 8m, 세로 4m의 크기이다. 정적 장애물의 대한 조건을 충족하기 위해서 벽의 높이를 0.5m로 정하였고 네모박스와 원기둥모양의 장애물을 추가하였다.



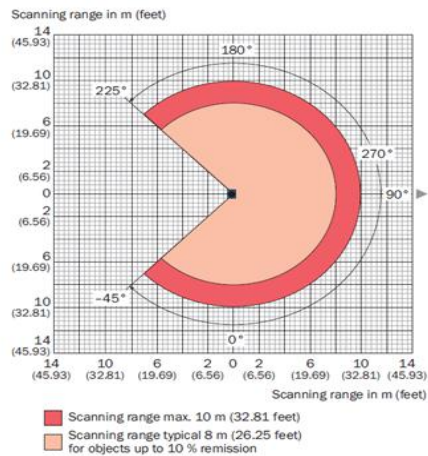
<그림 13> Gazebo7.0로 구현한 가상환경

시뮬레이터 상의 로봇 모델은 TurtleBot3 Burger로서 ROS 표준형 플랫폼 로봇이다. 해당 모델의 로봇은 2017년 로보티즈 사에 다이내믹셀을 이용하여 개발되었다. 로봇의 형태는 Differential 형태의 모바일 로봇이며, 최대 직선 속도 0.22m/s 와 최대 회전 속도 2.83rad/s 로 주행이 가능하다. 또한 IMU를 장착한 OpenCR1.0을 사용하여 모터를 제어하고 Raspberry Pi3에 ROS를 탑재하여 토픽을 통한 제어가 가능하다. 이 실험에서 Turtlebot3에 장착된 거리 센서의 최소 감지 거리가 로봇의 크기보다 커서 실제로 장애물과 충돌하는 거리를 감지하지 못하는 문제 때문에 장애물에 대한 정밀한 정보를 얻기

위해서 기존의 장착된 센서대신 SICK TIM55x 모델의 LRF을 모델링 하여 장착하였다.



<그림 14> TurtleBot3 burger

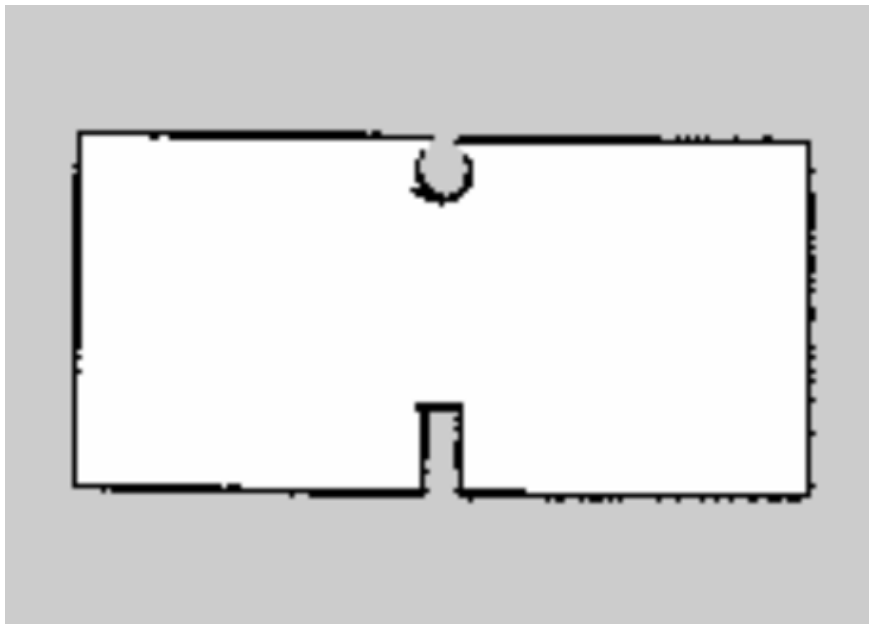


<그림 15> Sick Tim 55x 감지 영역

로봇에 장착되는 거리센서의 성능은 그림 15와 같이 270°를 1°단위로 감지하며 거리 범위는 0.05 ~ 10m이다. 또한 주기는 15 hz이다.

2. Navigation 구조 설계

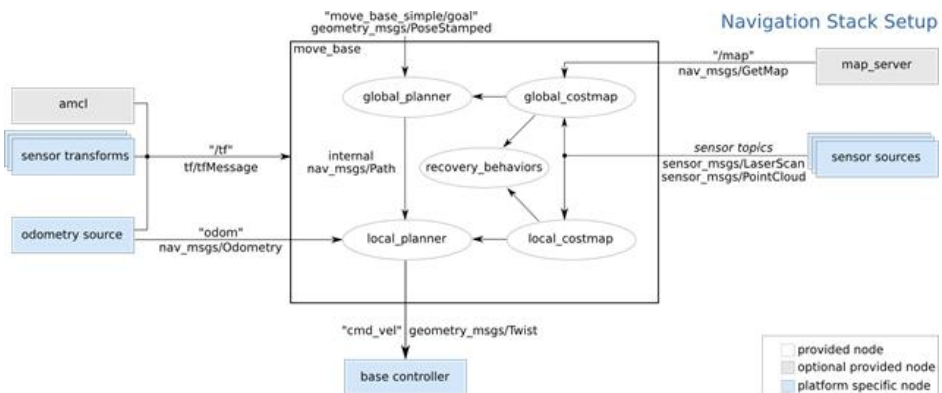
Navigation을 구동하기 위해서 필수적으로 요구되는 것은 해당 환경에 대한 사전 지식인 지도정도가 있어야한다. 이 지도 정보를 토대로 로봇이 현재 어느 위치에 있는지 장애물이 어디에 있는지를 알 수 있다. 실험환경에 대한 지도 정보를 얻기 위해서 대표적인 SLAM 알고리즘인 Gmapping[22]을 사용하여 다음 그림 16과 같은 지도 정보를 획득하였다.



<그림 16> 가상 실험환경에 대한 지도 정보

[회색영역 : 확인되지 않은 미지영역, 흰색영역 : 로봇이 이동 가능한 자유영역, 흑색영역 : 확인되지 않은 미지 영역]

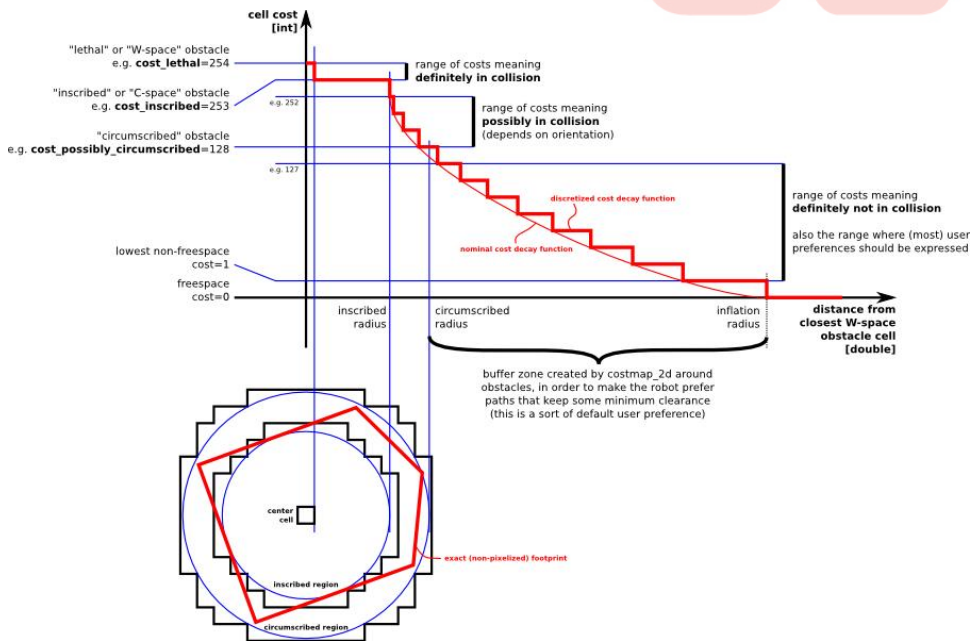
DWA과 포함되어 있는 navigation을 사용하기 위해서 ROS Navigation Package를 사용한다. 그림 17에서 Move base는 DWA가 적용된 Local Planner에게 받은 벡터 정보를 바탕으로 로봇의 구동계의 직접 명령을 내린다. 그리고 현재 위치에 대한 정보는 센서 기반의 AMCL과 로봇의 엔코더를 기반에 데드로커닝 그리고 IMU 센서를 통해 추정해낸다. Global Planner에서는 A*와 Dijkstra's 알고리즘을 결합한 경로계획[23]을 사용하며 로봇이 최종적으로 시작 지점에서 목표 지점까지 이동하는 최적의 경로를 생성하여 경로를 계획한다. 각 Node들을 ROS Core을 통한 토픽으로 모든 데이터를 주고받는 구조로 되어있다.



<그림 17> ROS Navigation Stack 모식도

Navigation Stack을 사용할 때 각 알고리즘에 부여된 매개변수를 조절하여 장애물 회피 주행의 성능과 목표위치까지 빠르게 도달하는 성능을 올릴 수 있다[24]. 대표적으로 Costmap[25]의 특성에서 inflation_radius에 값을 조절하여 장애물에 접근 못하는 영역을 늘려 장애물에 반응성을 크게 유도하였다. 또한 dwa_local_planner에 특성에서 전 방향 시뮬레이션 궤적시간과 DWA의 비용함수 $G(v, w)$ 의 가중치를 변경하여 기존 회피 성능과 비교 하였을 때 장애물 회피의

성공 확률을 20% 향상시켰다.

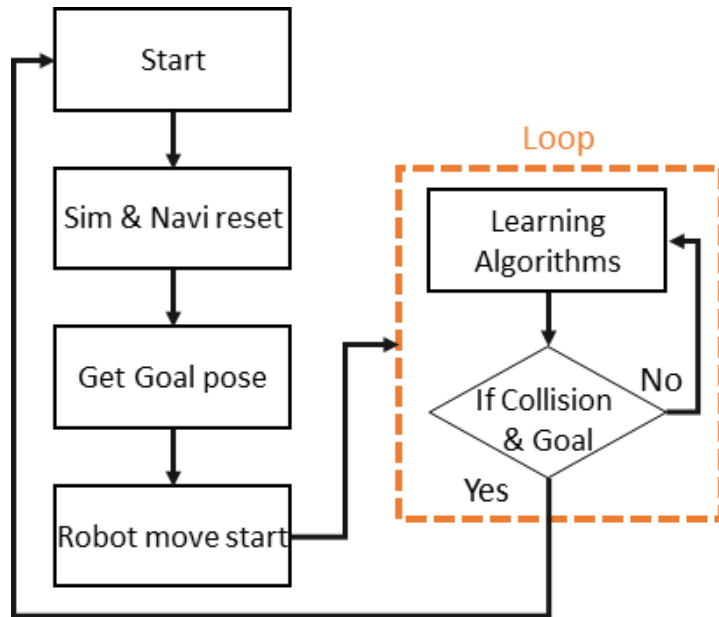


<그림 18> Costmap에서 inflation 정의

inflation은 거리에 따라 감소하는 점유 된 셀에서 비용 값을 전파하는 프로세스이다. 여기서 로봇이 장애물의 특정 영역을 회피하기 위해 장애물과 상관없이 해당 영역의 cost value를 설정할 수 있다. 이 값이 바로 inflation_radius 값이고 위 그림에서 128이지만, 실제 값은 사용자의 설정 값을 따른다.

3. 학습 알고리즘 설계와 학습

동적 장애물의 회피를 학습하기 위해서 다음 그림 19과 같이 학습 알고리즘을 설계하였다.



<그림 19> 설계된 학습 알고리즘

처음 학습을 시작과 함께 Simulation과 Navigation 알고리즘을 Reset 하여 모든 상태 조건을 초기화 시켜준다. 그 다음 목표 위치에 정보를 받아서 모바일 로봇의 주행을 시작한다. 로봇의 주행 명령이 시작되면 Learning algorithms loop안에서 상황(state)와 행동(action) 그리고 보상(reward) 순으로 반복하여 학습한다. 반복되는 상황 중 로봇이 장애물이나 벽에 충돌하거나 목적위치에 도달하게 되면 현 에피소드는 종료하게 되고 다음 에피소드를 시작하게 된다.

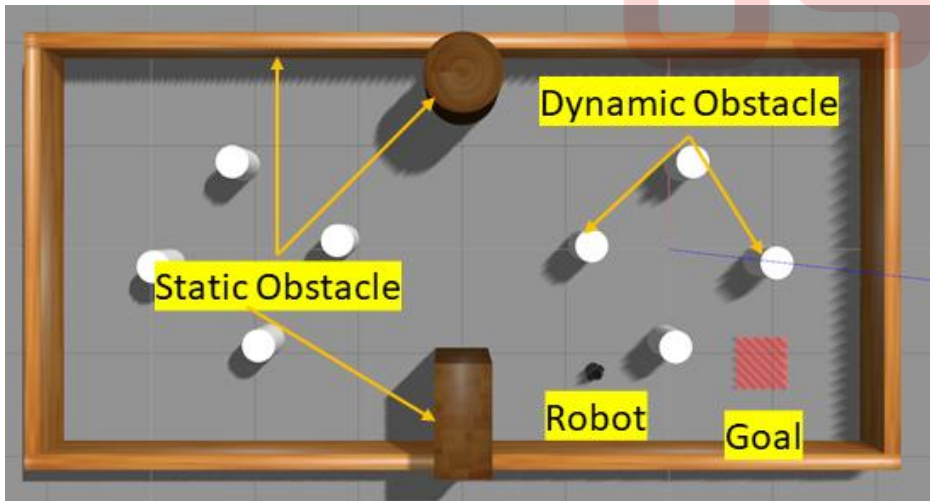
학습을 하기 위해서 학습 횟수, 한 에피소드 당 최대 몇 번까지

Step을 허용할지, 감쇠율의 크기를 설정하는 등 강화학습의 대한 세 부적인 세팅을 다음 표 3와 같이 구성하였다.

Hyper parameter	Default	Description
episode step	6000	The time step of one episode.
target update	2000	Update rate of target network.
discount factor	0.99	Represents how much future events lose their value according to how far away.
learning rate	0.00025	Learning speed. If the value is too large, learning does not work well, and if it is too small, learning time is long.
epsilon	1.0	The probability of choosing a random action.
epsilon decay	0.996	Reduction rate of epsilon. When one episode ends, the epsilon reduce.
epsilon min	0.05	The minimum of epsilon.
batch size	64	Size of a group of training samples.
train start	64	Start training if the replay memory size is greater than 64.
memory	1000000	The size of replay memory.

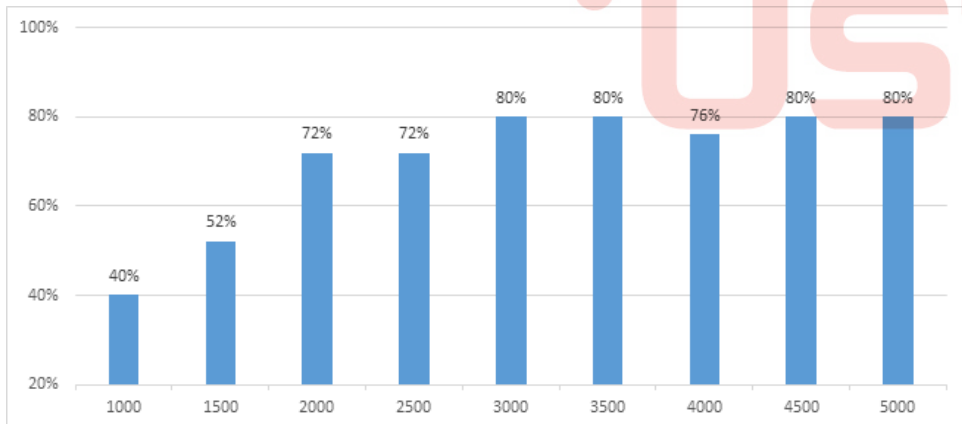
<표 3> DQN parameter

동적 장애물의 회피 학습을 하기 위해 다음 그림 20과 같은 가 상환경을 구축하였다.



<그림 20> 장애물과 로봇 그리고 목표 위치 구성

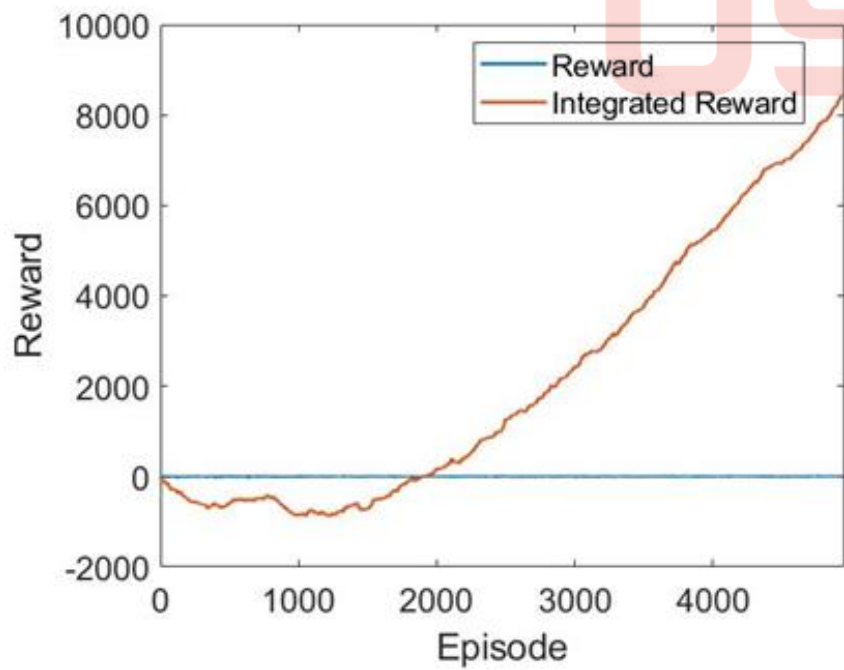
위 그림과 같이 로봇은 주어진 환경 정보를 센서 데이터를 통해 목표 위치까지 주행을 한다. 장애물의 구성은 동적 장애물과 정적 장애물로 구성되어 있는데 동적 장애물은 0.85m 반경으로 4개의 장애물이 중점 위치를 기준으로 하여 8rad/s로 회전한다. 초기 장애물과 로봇이 매번 다른 상황에서 만나도록 장애물의 처음 움직이는 시간을 다양하게 구성하였다. 장애물과 로봇의 충돌조건은 센서 정보가 0.13m이내로 접근하면 충돌로 간주한다. 목표 위치에 도달하는 조건은 현재 로봇의 위치와 목표 위치 사이에 거리가 0.2m 이내가 되면 도달하였다고 판단한다.



<그림 21> 에피소드 진행에 따른 장애물 회피 성공률

학습의 에피소드는 5000번을 수행하였다. 에피소드의 진행에 따라서 장애물 회피 실험을 25씩 진행하여 그림 21이라는 결과를 얻었다. 이 실험 결과로 학습이 진행될수록 성능이 상승하고 3000번 이후로는 80% 성공률에 수렴하는 결과를 얻었다. 최종 에피소드까지 따른 전체 보상은 다음 그림 22와 같다.

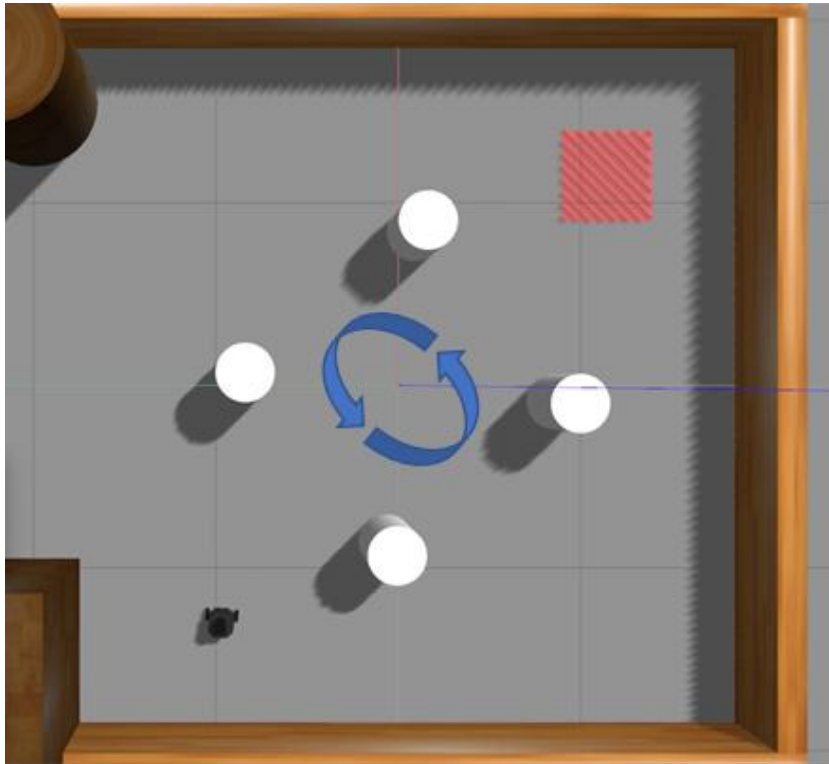
학습에 일반화를 막기 위해서 장애물과 만나는 조건을 매번 다르게 설정과 또한 장애물 충돌을 제외한 부정적인 보상으로 인해 그래프에 편차가 많이 발생하였다. 하지만 에피소드의 흐름에 따라서 전체적인 보상이 커지는 현상이 발생하므로 학습이 되고 있다는 것을 확인하였다. 다음 기존 방법과 학습한 방법을 비교하는 실험을 통해 학습결과를 검증하고자 한다. 실험방법은 3가지로 구성되며 학습한 동일한 환경, 장애물이 직진 운동을 하는 경우 마지막으로 실제 환경에서 이 알고리즘이 구동이 가능한지를 테스트하려고 한다.



<그림 22> 에피소드 진행에 따른 누적 보상 그래프

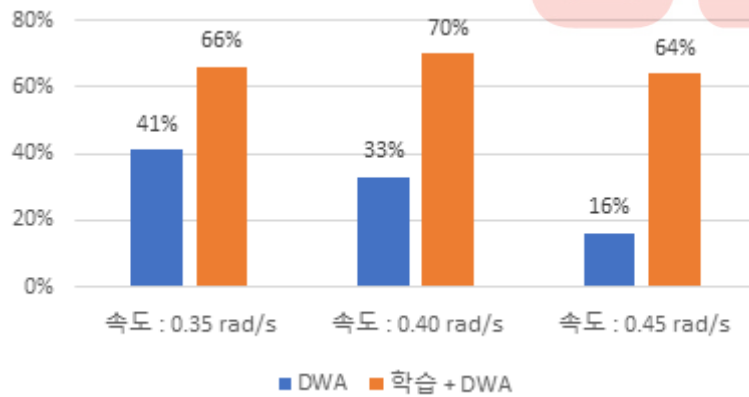
4. 가상 환경 실험

가. 회전 운동 장애물 환경 실험



<그림 23> 회전 운동하는 장애물 환경 구성

실험의 환경 조건은 그림 23과 같이 장애물이 회전하는 환경에서 주행을 하는 것이다. 장애물과 만나는 지점을 다르게 하기 위해서 로봇이 출발하는 시간을 다르게 적용하였다. 본 실험은 동적 장애물의 속도를 다르게 하여 학습된 속도를 제외한 다른 두 가지 속도에서도 학습된 알고리즘에 성능이 유효한지 확인하였다. 표본의 신뢰도를 올리기 위해서 매 실험은 100회 수행하여 진행하였다.



<그림 24> 회전하는 장애물 실험에 대한 결과 그래프

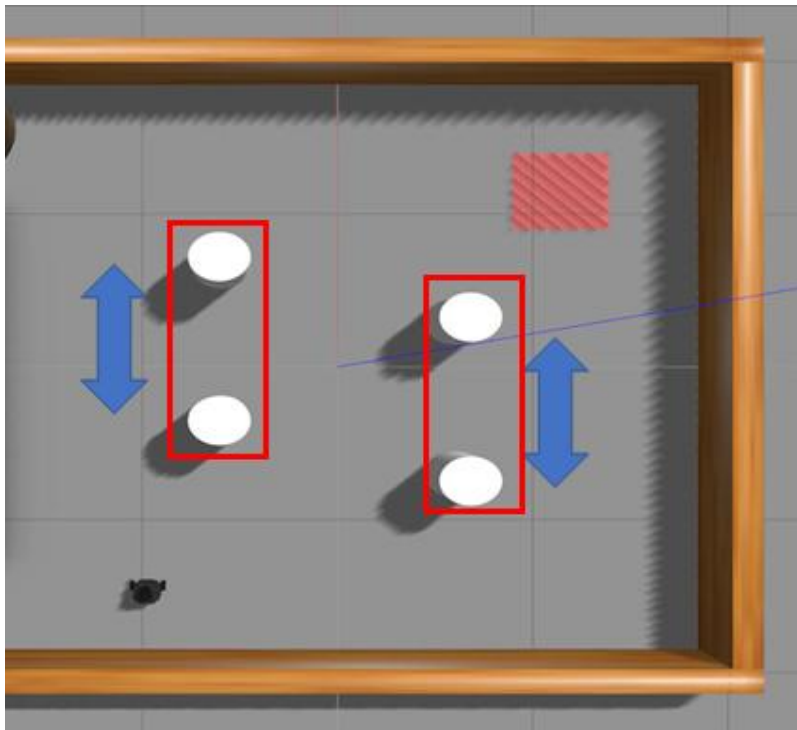
(성공률)	속도: 0.35 rad/s	속도: 0.40 rad/s	속도: 0.45 rad/s
DWA	41%	33%	16%
학습 + DWA	66%	70%	64%

<표 4> 회전하는 장애물 속도에 대한 결과[성공률]

표 4과 그림 24의 실험결과를 보면 회전하는 장애물들을 여러 속도에 대하여 실험한 결과 동적 장애물 회피 학습이 된 것을 확인할 수가 있었다. 또한 학습된 속도가 아닌 다른 속도에서도 학습된 모듈을 사용한 알고리즘의 장애물 회피에서도 기존 방식보다 높은 성공률을 가진 것을 확인하였다. 앞서 설명하였던 것처럼 기존의 방식으로 주행을 하였을 때 동적 장애물의 속도가 빨라질수록 회피 성능이 점차 하락하지만 제안된 방식에서는 실험한 환경 내에서 전부 64%이상의 성공률을 보장하였다.

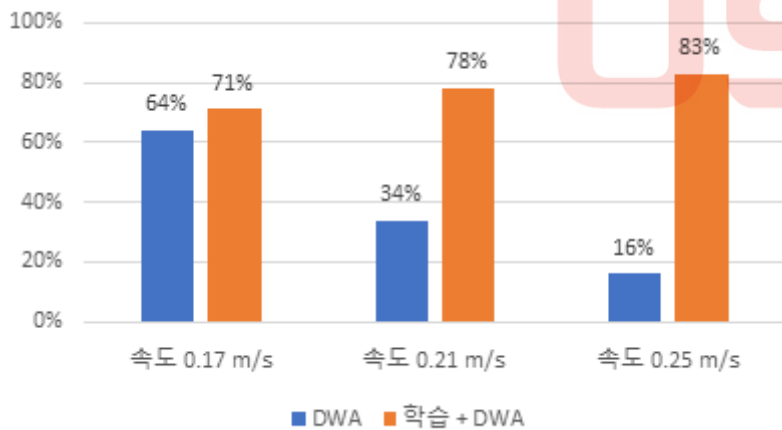
나. 직진 운동 장애물 환경 실험

다른 환경에서 제안된 장애물 회피 알고리즘이 유효한지 판단하기 위해서 다음 그림 25와 같은 조건의 환경을 구성하였다.



<그림 25> 직진 운동하는 장애물 환경 구성

이 환경 장애물은 상하로 움직이며 빨간색 상자를 기준으로 한 쌍으로 움직인다. 오른쪽 장애물의 속도를 약간의 편차(-0.01m/s)를 두고 올라가고 내려가는 조건을 바꾸어 장애물과 로봇이 만나는 조건을 다양하게 구성하였다. 각 장애물이 움직이는 거리는 상하로 각각 0.4m 만큼 움직이고 최대 위치만큼 올라가고 내려가면 반대 방향으로 변경되어서 다시 움직이도록 구성되었다.



<그림 26> 직진하는 장애물 실험에 대한 결과 그래프

(성공률)	속도 0.17 m/s	속도 0.21 m/s	속도 0.25 m/s
DWA	64%	34%	16%
학습 + DWA	71%	78%	83%

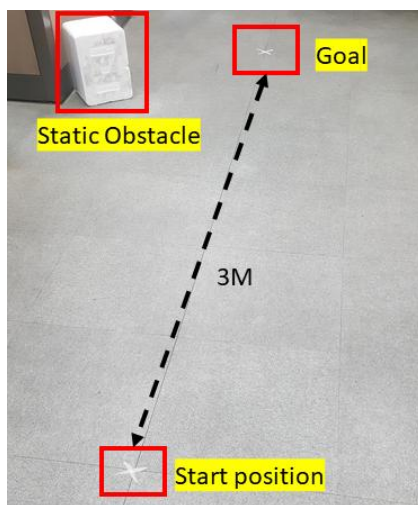
<표 5> 직진하는 장애물 속도에 대한 결과[성공률]

이 실험은 대한 결과는 표 4와 그림 26에서 보여주는 것 같이 빠른 장애물에 대해서도 유효한 결과를 확인 할 수 있었다. 학습된 환경과 동적 장애물에 대한 조건이 많이 변경되었고 더욱 복잡한 환경이었음에도 기존 방법보다 더 큰 안전성을 보인 것을 확인하였다. 이 실험 또한 DWA만 사용한 경우 장애물에 속도에 따라서 동적 장애물의 대한 회피 성능이 비례하여 감소하지만 속도 제어 보조 모듈을 장착한 DWA 알고리즘인 경우에 오히려 속도가 빠른 장애물에 대해서 성능이 상승하는 것을 확인하였다.

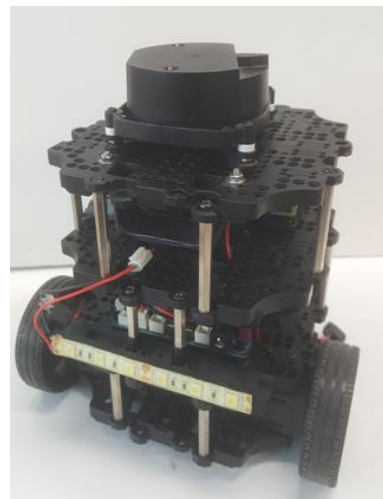
5. 현실 세계 실험

가. 실험 환경 구성

이 논문에서 제안된 방법이 실제로 로봇을 구동하여 주행을 하였을 때도 작동이 가능한지 확인하였다. 가상환경에서 구동되는 알고리즘들이 현실에서도 적용할 수 있도록 구성하였기 때문에 추가적인 작업 없이도 바로 적용할 수 있다. 실험은 정적 장애물만 있는 환경에서 주행과 동적 장애물이 존재하는 환경 두 가지를 진행하였다. 실험을 하기 위해서 사용한 로봇은 가상환경에서 구현한 로봇과 동일한 Robotis사에 Turtlebot3 burger 모델을 사용하였다. 정적 장애물과 동적 장애물은 상자를 가지고 구현 하였다. 실험 환경 구성은 그림 27에서 보여주는 것 같이 배치되었다.

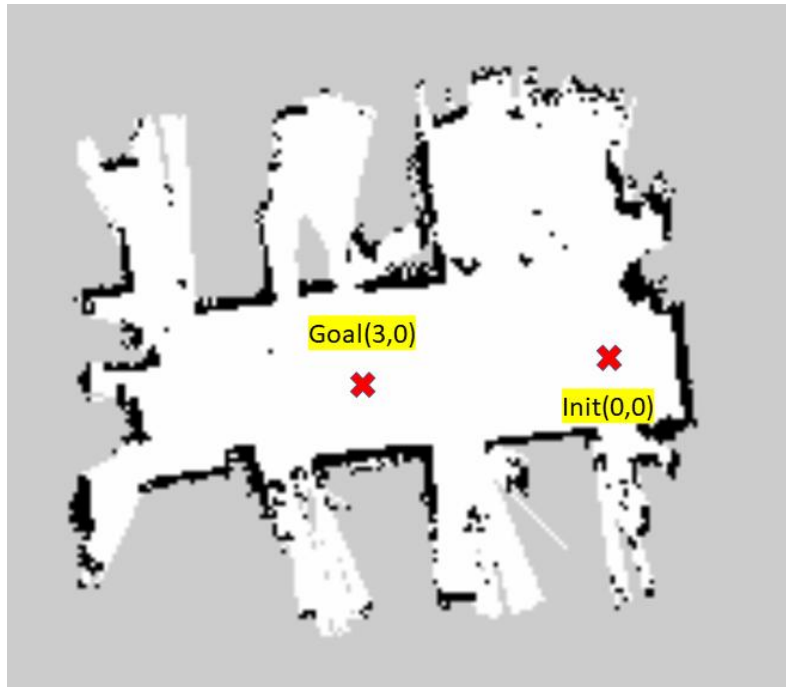


<그림 27> 실제 주행 실험
환경



<그림 28> 주행에 사용된
로봇 Turtlebot3 burger

또한 지도 정보의 로봇의 위치정보를 획득하기 위해서 SLAM을 이용하여 주변 환경을 관측하여 지도 정보를 구성하였고 목표 위치 (3.0, 0)와 시작 위치(0, 0)를 정하였다.

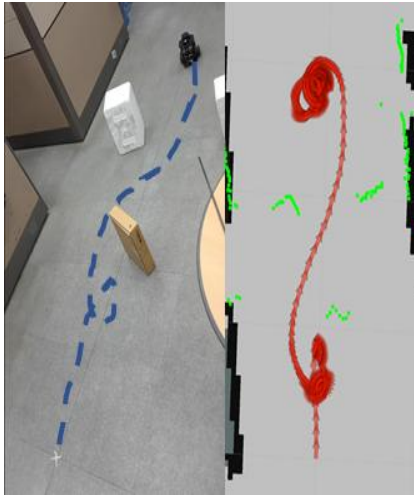


<그림 29> 실제 주행한 환경의 Grid map

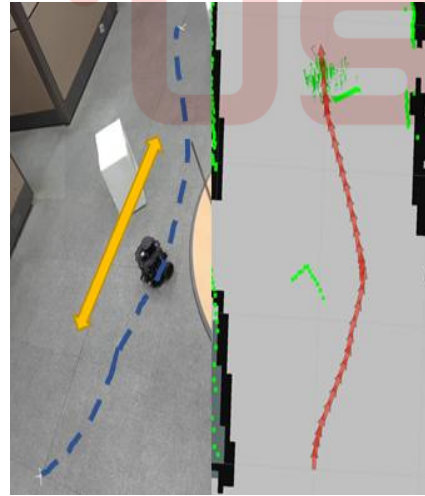
나. 실험 및 결과

실험은 3가지 유형의 실험을 진행하였다. 그림 30 ~ 32와 같이 정적 장애물만 존재하는 상황, y축 방향으로 움직이는 장애물 상황 그리고 x축 방향으로 움직이는 장애물 상황으로 구성하였다.

먼저 1번째 실험인 정적 장애물만이 존재하는 환경에서 로봇은 사전에 알려주지 않았던 장애물들을 성공적으로 회피하였으며 목표 위치까지 도달한 것을 확인 할 수 있다. 이를 기존의 알고리즘에서 구현되었던 기능이 제안된 방법에서도 유효한 것을 확인하였다. 2번째 실험은 로봇의 주행방향과 동일한 각도로 다가오는 동적 장애물에 대한 회피 성능을 체크하였다. 기존의 방법으로 이러한 환경에 처하게 되면 속도영역에 손실 문제로 장애물 회피를 하지 못하였지만 제안된 방법을 적용한 로봇은 이 실험에서 사전에 장애물에 대한 회피를 진행하여 성공적으로 회피하였다. 마지막 실험은 동적 장애물이 센서로 미리 감지 못하는 벽 뒤에서 갑자기 나타나는 구성이었다. 이 실험에서 주행 중인 로봇은 숨어있던 장애물에 대해 반응하여 안정적으로 동적 장애물 회피를 수행하였다.

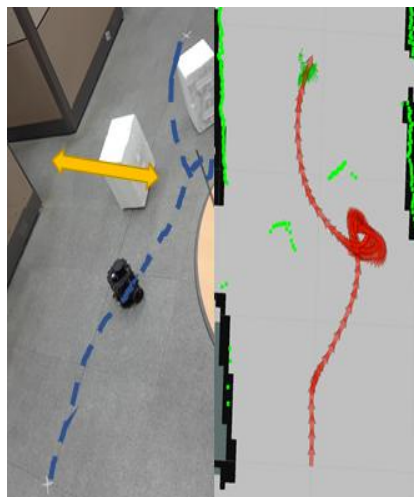


<그림 30> 정적 장애물 환경



<그림 31> 동적 장애물 환경

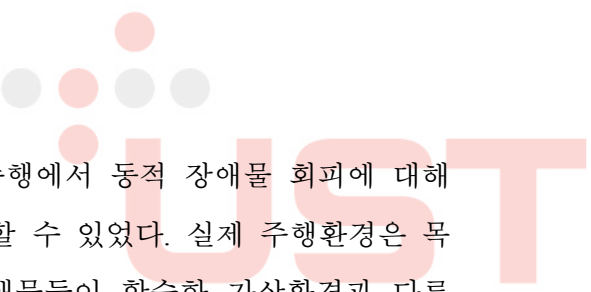
(x축)



<그림 32> 동적 장애물 환경

(y축)

[좌 : 실제 주행 영상, 파란 점선 : 로봇의 실제 주행 궤적, 노란 화살표 : 이동 장애물의 진행방향
우 : Rviz 상의 로봇의 주행, 빨간 궤적 : 로봇의 측정된 주행 궤적, 초록 포인트 : 센서의 감지된 거리 정보]



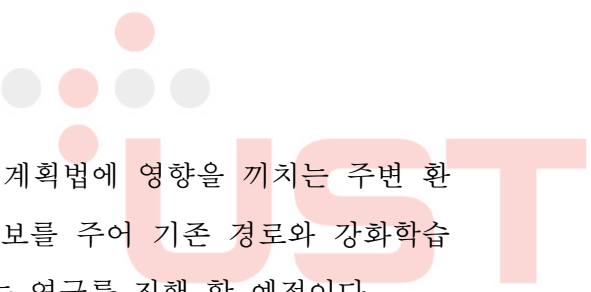
3가지 실험결과를 통해 실제 주행에서 동적 장애물 회피에 대해서 유효한 주행을 하는 것을 확인할 수 있었다. 실제 주행환경은 목표위치와 출발 위치에 구성과 장애물들이 학습한 가상환경과 다른 구조에 환경이었다. 이러한 어려운 조건들에서도 위 그림의 경로들은 장애물들과 충돌 없이 목표 위치까지 도달하는 것을 보여주었다.

V. 결 론

본 논문에서는 동적 환경에서 강화학습을 이용하여 동적 장애물을 탐지하고 회피하는 알고리즘을 제안하였다. 대표적인 DWA 알고리즘과 심층강화학습 DQN을 적용하여 이동 속도가 빠른 동적 장애물 회피를 할 수 있는 보조 속도 제어 모듈을 설계하였다. 보조 속도 제어 모듈은 주변 상태에 대하여 어떤 속도 증감 및 가중치를 배분할 것인지를 보상을 상승시키는 방향으로 학습한다. 학습이 점차 진행되면서 빠른 동적 장애물에 대한 장애물 회피 성능이 상승하는 것을 확인 하였고 이를 기존 DWA만 사용하는 방법과 비교하여 실험을 통해 회피성능을 검증하였다. 로봇은 회전하는 동적 장애물과 로봇이 다른 각도로 만나는 환경에서 5000 에피소드 학습을 진행하였다. 학습이 진행될수록 로봇의 회피 성공률이 높아지는 결과를 얻었다. 최종 학습된 방법을 기존과 비교하였을 때 장애물 회피 성공률이 2.12배 높은 것이 확인되었다. 또한 학습된 환경이 아닌 다른 장애물 환경과 다른 속도에서도 회피성능이 기존과 비교하여 회피 성공률이 평균 15%이상 상승 되었다. 이를 통해서 동적 장애물 회피에 대해서 추가적인 보조 속도제어를 통해서 장애물 회피 성능을 향상 시킬 수 있다는 것을 증명하였다.

하지만 본 논문의 한계점이 있다. 회피 성능이 상승하였지만 이동시간이 5 ~ 10%정도 증가하는 문제가 발생하였다. 이 문제는 로봇이 장애물에 대한 회피 주행을 하면서 회피를 위한 추가 모션이 발생하여 일어나는 문제이다. 또한 설계한 행동에 회피 성능에 영향을 끼치는 한계점이다. 기본적으로 본 논문에서 제안하는 방법은 기존의 DWA 경로에 영향을 주어 회피 주행하는 방식이기에 제어 한계를 벗어난 속도 모델이 요구될 때 충돌하는 경우가 발생한다.

따라서 향후 연구과제로 동적 장애물에 대한 강화학습으로 영향을 주는 값들이 로봇의 움직임으로 바로 대입시켜 속도에 제어의



한계를 넓힐 계획이다. 또한 경로 계획법에 영향을 끼치는 주변 환경 정보에 동적 장애물에 대한 정보를 주어 기존 경로와 강화학습이 제안하는 경로가 일치할 수 있는 연구를 진행 할 예정이다.

참 고 문 헌

- [1] Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza (2011), *‘Introduction to autonomous mobile robots’*, MIT press.
- [2] Wong, Cuebong, et al (2017), *‘Adaptive and intelligent navigation of autonomous planetary rovers—A survey’*, NASA/ESA Conference on Adaptive Hardware and Systems (AHS). IEEE.
- [3] Pandey, Anish, S. Pandey, and D. R. Parhi (2017), *‘Mobile robot navigation and obstacle avoidance techniques: A review’* Int Rob Auto J 2.3 : 00022.
- [4] D. Fox and W. Burgard and S. Thrun (1997), *‘The Dynamic Window Approach to Collision Avoidance’*, IEEE Robotics & Automation Magazine, 4(1), 23-33.
- [5] Li, Xiuyun, et al (2017), *‘Obstacle avoidance for mobile robot based on improved dynamic window approach’*, Turkish Journal of Electrical Engineering & Computer Sciences 25.2, 666-676.
- [6] J. Borenstein and Y. Koren (1991), *‘The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots’*, IEEE Transactions on Robotics and Automation, 7(3), 278-288.
- [7] 오세권, 김주민, and 김대원 (2012), ‘동적 환경에서 LRF 센서를 이용한 이동 장애물 탐지 알고리즘 개발’, 대한전기학회 학술대회 논문집, 1385-1386.
- [8] LaValle, Steven M (1998), *‘Rapidly-exploring random trees: A new tool for path planning’*.
- [9] Connell, Devin, and Hung Manh La(2018), *‘Extended rapidly exploring random tree-based dynamic path planning and replanning for mobile robots’*, International Journal of Advanced Robotic Systems 15.3, 1729881418773874.

- [10] P. Fiorini and Z. Shiller (1998), '*Motion planning in dynamic environments using velocity obstacles*', Int. J. Robot. Res, vol. 17, no. 7, 760–772.
- [11] Van den Berg, J., Lin, M., & Manocha, D. (2008), '*Reciprocal velocity obstacles for real-time multi-agent navigation*', In 2008 IEEE International Conference on Robotics and Automation, 1928-1935, IEEE.
- [12] Montiel, O., Orozco-Rosas, U., & Sepúlveda, R (2015), '*Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles*', Expert Systems with Applications, 42(12), 5177-5191.
- [13] Rostami, S. M. H., Sangaiah, A. K., Wang, J., & Liu, X. (2019), '*Obstacle avoidance of mobile robots using modified artificial potential field algorithm*', EURASIP Journal on Wireless Communications and Networking, 70.
- [14] Xin, J., Zhao, H., Liu, D., & Li, M. (2017), '*Application of deep reinforcement learning in mobile robot path planning*', In 2017 Chinese Automation Congress (CAC), 7112-7116, IEEE.
- [15] A. Khare, R. Motwani, S. Akash, J. Patil and R. Kala (2018), '*Learning the Goal Seeking Behaviour for Mobile Robots*', 2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Singapore, 56-60.
- [16] M. Duguleana and G. Mogan (2016), '*Neural networks based reinforcement learning for mobile robots obstacle avoidance*', Expert Syst. Appl., vol. 62, 104–115.
- [17] Sutton, Richard S., and Andrew G. Barto (1998). '*Reinforcement learning: An introduction. Vol. 1. No. 1.*' Cambridge: MIT press.
- [18] Thie, Paul R (1983), '*Markov decision processes*'. Comap, Incorporated.
- [19] Watkins, Christopher JCH, and Peter Dayan (1992), '*Q-learning*', Machine learning 8.3-4, 279-292.

- [20] Mnih, Volodymyr, et al (2013), '*Playing atari with deep reinforcement learning*', arXiv preprint arXiv:1312.5602.
- [21] Dellaert, Frank, et al (1999), '*Monte carlo localization for mobile robots*' ICRA. Vol. 2.
- [22] Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard (2005) '*Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling*', Proceedings of the 2005 IEEE international conference on robotics and automation. IEEE.
- [23] Brock, O., & Khatib, O. (1999). '*High-speed navigation using the global dynamic window approach*' In Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C) (Vol. 1, 341-346). IEEE.
- [24] Kaiyu Zheng. (2016), '*ROS Navigation Tuning Guide*'
- [25] Eltan M, David V. L, Dave H (2018), '*costmap_2d*', Retrieved August 5, 2018, from http://wiki.ros.org/costmap_2d