(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0063791
(43) 공개일자 2021년06월02일

(51) 국제특허분류(Int. Cl.)
G05D 1/02 (2020.01) *B60W 30/14* (2006.01)
B60W 40/02 (2006.01)
(52) CPC특허분류
G05D 1/0221 (2013.01)
B60W 30/14 (2013.01)
(21) 출원번호 10-2019-0152319
(22) 출원일자 2019년11월25일
심사청구일자 2019년11월25일

(71) 출원인
한국기술교육대학교 산학협력단
충청남도 천안시 동남구 병천면 충절로 1600 (한국기술교육대학교내)
(72) 발명자
유승열
충청남도 천안시 동남구 일봉로 71, 102-401
윤범진
경상북도 봉화군 재산면 미잠길 84-104
(74) 대리인
김견수

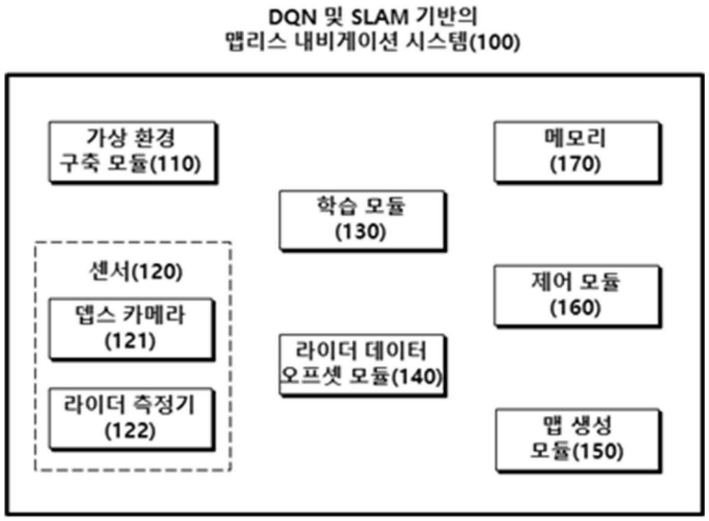
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 **장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법**

(57) 요 약

본 발명은 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법에 관한 것으로, 로봇, 차량, 드론 등의 무인 이동체의 자율 주행을 위한 DQN 기반의 맵리스 내비게이션을 구현하여 타겟의 크기를 점차 줄여가면서 학습 시간을 감소시키고, 사람의 개입 없이 영상인식과 거리측정을 통해 확인한 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 부여하여 SLAM에 적용함으로써, 무인 이동체가 여러 장애물이 존재하고 사람과 같이 협업하는 스마트 팩토리 환경에서 사람이나 장애물과의 안전거리를 고려하여 최적의 경로 주행을 수행할 수 있도록 하는 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법에 관한 것이다.

대 표 도 - 도1



(52) CPC특허분류
B60W 40/02 (2013.01)
G05D 1/0238 (2013.01)
B60W 2420/52 (2013.01)
B60W 2554/00 (2020.02)
B60W 2556/50 (2020.02)

이 발명을 지원한 국가연구개발사업
과제고유번호 20004687
부처명 산업통상자원부
과제관리(전문)기관명 한국산업기술평가관리원
연구사업명 스마트공장제조핵심기술개발사업
연구과제명 생분해성 고분자 봉합원사 및 덴탈재료 제조공정의 10%이상의 생산성 향상과 생산품질 5% 향상을 위해 CCPS 및 IT 신기술 도입을 통한 의료바이오 분야의 고도화된 대표 스마트공장 구축
기 여 율 1/1
과제수행기관명 (주)메타바이오메드
연구기간 2019.07.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

사람, 물체 또는 이들의 조합을 포함한 장애물이 위치한 소정의 공간에서, 현재 상태에서 다음 상태로의 무인 이동체에 대한 액션을 출력하기 위하여, 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리 데이터, 상기 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 학습하는 학습 모듈;

상기 학습을 수행하는 과정에서, 상기 장애물이 감지되면 상기 장애물의 종류, 속도 또는 이들의 조합을 포함한 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하는 라이더 데이터 오프셋 모듈; 및

상기 오프셋한 라이더 데이터를 입력받아 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시하는 맵 생성 모듈;을 포함하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템.

청구항 2

청구항 1에 있어서,

상기 학습 모듈은,

각 에피소드에 따른 학습을 수행할 때 각 스텝 타임마다 상기 무인 이동체에서 측정한 라이더 데이터를 참조하여 액션을 선택하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템.

청구항 3

청구항 2에 있어서,

상기 에피소드는,

상기 무인 이동체가 타겟 지점에 도착하거나, 벽, 장애물 및 사람과 충돌이 발생하거나, 또는 상기 타겟 지점에 기 설정된 시간 이내에 도착하지 못했을 경우를 하나의 에피소드로 설정하며,

상기 충돌은,

상기 무인 이동체가 벽, 장애물 및 사람의 기 설정된 범위 이내로 접근하였을 때를 충돌로 판정하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템.

청구항 4

청구항 2에 있어서,

상기 학습 모듈은,

상기 에피소드를 최초에 수행할 때 상기 타겟 지점의 반경을 원래 크기의 정수 배로 확장하여 설정하고,

상기 에피소드의 수행에 따라 상기 무인 이동체가 상기 타겟 지점에 도착할 때마다 상기 타겟 지점의 반경을 기 설정된 범위로 줄여 다음의 에피소드를 수행하며,

상기 타겟 지점의 반경이 원래 크기가 될 때까지 각 에피소드를 수행함으로써, 탐색 범위를 좁혀 학습 시간을 단축하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템.

청구항 5

청구항 1에 있어서,

상기 라이더 데이터 오프셋 모듈은,

상기 무인 이동체에서 촬영한 상기 장애물의 뎁스 데이터 및 영상인식에 따른 바운딩 박스 데이터를 토대로 상

기 장애물을 감지하여 상기 장애물의 종류를 확인하고,

상기 장애물이 감지되면 상기 무인 이동체에서 측정한 현재 상태와 이전 상태간의 라이더 데이터의 차이를 계산하고, 상기 계산한 라이더 데이터의 차이를 통해서 상기 장애물의 위치를 확인하며,

상기 확인한 장애물의 종류에 따라 상기 확인한 장애물의 위치로부터 설정해야 할 안전거리를 차등적으로 부여하고, 상기 장애물과, 상기 차등적으로 부여한 안전거리에 대한 라이더 데이터의 오프셋을 수행하는 것을 특징으로 하는 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템.

청구항 6

DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 사람, 물체 또는 이들의 조합을 포함한 장애물이 위치한 소정의 공간에서, 현재 상태에서 다음 상태로의 무인 이동체에 대한 액션을 출력하기 위하여, 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리 데이터, 상기 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 학습하는 학습 단계;

상기 학습을 수행하는 과정에서, 상기 장애물이 감지되면 상기 장애물의 종류, 속도 또는 이들의 조합을 포함한 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하는 라이더 데이터 오프셋 단계; 및

상기 오프셋한 라이더 데이터를 입력받아 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시하는 맵 생성 단계;를 포함하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법.

청구항 7

청구항 6에 있어서,

상기 학습 단계는,

상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 각 에피소드에 따른 학습을 수행할 때 각 스텝 타임마다 상기 무인 이동체에서 측정한 라이더 데이터를 참조하여 액션을 선택하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법.

청구항 8

청구항 7에 있어서,

상기 에피소드는,

상기 무인 이동체가 타겟 지점에 도착하거나, 벽, 장애물 및 사람과 충돌이 발생하거나, 또는 상기 타겟 지점에 기 설정된 시간 이내에 도착하지 못했을 경우를 하나의 에피소드로 설정하며,

상기 충돌은,

상기 무인 이동체가 벽, 장애물 및 사람의 기 설정된 범위 이내로 접근하였을 때를 충돌로 판정하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법.

청구항 9

청구항 7에 있어서,

상기 학습 단계는,

상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 상기 에피소드를 최초에 수행할 때 상기 타겟 지점의 반경을 원래 크기의 정수 배로 확장하여 설정하는 단계;

상기 에피소드의 수행에 따라 상기 무인 이동체가 상기 타겟 지점에 도착할 때마다 상기 타겟 지점의 반경을 기 설정된 범위로 줄여 다음의 에피소드를 수행하는 단계; 및

상기 타겟 지점의 반경이 원래 크기가 될 때까지 각 에피소드를 수행하는 단계;를 더 포함하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법.

청구항 10

청구항 6에 있어서,

상기 라이더 데이터 오프셋 단계는,

상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 상기 무인 이동체에서 촬영한 상기 장애물의 뎁스 데이터 및 영상인식에 따른 바운딩 박스 데이터를 토대로 상기 장애물을 감지하여 상기 장애물의 종류를 확인하는 단계;

상기 장애물이 감지되면 상기 무인 이동체에서 측정한 현재 상태와 이전 상태간의 라이더 데이터의 차이를 계산하고, 상기 계산한 라이더 데이터의 차이를 통해서 상기 장애물의 위치를 확인하는 단계; 및

상기 확인한 장애물의 종류에 따라 상기 확인한 장애물의 위치로부터 설정해야 할 안전거리를 차등적으로 부여하고, 상기 장애물과, 상기 차등적으로 부여한 안전거리에 대한 라이더 데이터의 오프셋을 수행하는 단계;를 더 포함하는 것을 특징으로 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법에 관한 것으로, 더욱 상세하게는 로봇, 차량, 드론 등의 무인 이동체의 자율 주행을 위한 DQN(Deep Q Network) 기반의 맵리스 내비게이션을 구현하여 타겟의 크기를 점차 줄여가면서 학습 시간을 감소시키고, 사람의 개입 없이 영상인식과 거리측정을 통해 확인한 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 부여하여 SLAM(Simultaneously Localization and Mapping)에 적용함으로써, 무인 이동체가 여러 장애물이 존재하고 사람과 같이 협업하는 스마트 팩토리 환경에서 사람이나 장애물과의 안전거리를 고려하여 최적의 경로 주행을 수행할 수 있도록 하는 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법에 관한 것이다.

배 경 기 술

[0002] 최근 들어 고성능 GPU 및 CPU의 개발로 컴퓨팅 속도가 급속도로 발전하면서 인공지능 기술을 이용한 제어 시스템 및 데이터 분석이 활발하게 이루어지고 있으며, 특히 4차 산업혁명에 따라 제조업 분야에서 인공지능 기술과 IoT 기술을 기존 공장 및 창고 설비에 접목시킨 스마트 팩토리(smart factory)에 대한 연구 및 개발이 급속하게 확장되고 있다.

[0003] 상기 스마트 팩토리는 다수의 IoT 센서 등을 통해 취득된 데이터를 인공지능 기반의 제어 시스템을 통해 설비의 제어 및 고장진단을 수행한다. 이러한 스마트 팩토리의 중요 요소 중 하나는 기존의 단방향 컨베이어 벨트 기반의 생산 시스템과 다른 유연한 물류이송을 위한 생산 시스템의 구축인데, 최근 지어지고 있는 공장들은 로봇을 통해 공정 설비 간에 자유롭게 재료 및 물류를 이송하여 다품종의 생산에 대응하고 있다.

[0004] 이를 위해서, 상기 로봇은 우선적으로 주행할 공간의 장애물에 대한 정보를 SLAM을 통해 맵의 형태로 취득하고 위치를 동기화한 후, 목표지점까지의 경로를 생성하여 이동하게 된다.

[0005] 하지만 SLAM을 수행하기 위해서는 초기에 장애물의 위치와 움직일 수 있는 공간을 사람이 직접 조작하여 교육해야 하며, 공장이나 창고와 같은 대규모의 공간에서 SLAM을 수행하고 맵을 만드는 것은 시간이 많이 소요되고 전담 인력이 필요하므로 효율성이 저하된다.

[0006] 또한, 공장이나 창고에서 사용되는 로봇들은 사람과 같은 공간에서 협업하는 것을 전제로 하기 때문에, 자율 주행을 수행할 때 사람 및 중요 장비 등을 인지하고 회피하거나 또는 안전거리를 가중하는 조치가 필요하다. 예를 들어 컴퓨터 혹은 고가의 장비에 부딪히거나 딱딱한 책상에 부딪힌다면 단순히 박스에 부딪히는 것과 비교하여 손해가 커질 수 있기 때문이다. 그러므로 상기 로봇은 자율 주행을 하면서 이러한 위험 요소들에 대한 위치 정보를 사용자에게 알리거나, 또는 맵 상에 표기해야 할 필요성이 발생한다.

[0007] 즉 종래의 로봇들은 SLAM을 통해 수동으로 맵을 생성한 후, 중요 장비의 위치 등 장애물의 특성을 추가적인 작업을 통해 맵 상에 표기해야 하는 문제점이 있었다.

[0008] 따라서 본 발명에서는 무인 이동체를 위한 DQN을 기반으로 하는 맵리스 내비게이션을 구현하여 학습 시간을 감

소시키기 위해서 타겟의 크기를 최초에 2배로 조정한 후 점차 줄여가는 방식을 사용하고, 장애물의 특성을 파악하여 각 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 SLAM에 적용하여 맵 상에 표시할 수 있는 방안을 제시하고자 한다.

[0009] 이를 통해 본 발명은 무인 이동체가 SLAM을 자율적으로 수행할 수 있으며, 사람과 여러 장애물이 존재하는 스마트 팩토리 환경에서 사람이나 장애물과의 충돌 없이 타겟 지점까지 최적의 경로로 주행할 수 있게 된다.

[0010] 다음으로 본 발명의 기술분야에 존재하는 선행기술에 대하여 간단하게 설명하고, 이어서 본 발명이 상기 선행기술에 비해서 차별적으로 이루고자 하는 기술적 사항에 대해서 기술하고자 한다.

[0011] 먼저 한국공개특허 제2019-0029524호(2019.03.20.)는 경로를 자율주행하도록 로봇을 훈련시키기 위한 시스템 및 방법에 관한 것으로, 초기화 위치에서 로봇의 초기 배치를 검출하고, 상기 초기화 위치로부터 출발하여 상기 로봇에게 내비게이션 가능한 경로를 데모하는 동안에, 상기 내비게이션 가능한 경로 및 주위 환경의 맵을 생성시키고, 상기 초기화 위치에서 상기 로봇의 다음 배치를 검출하고, 상기 로봇으로 하여금 상기 초기화 위치로부터 상기 내비게이션 가능한 경로의 적어도 일부를 자율적으로 내비게이션하게 하는 것을 기술적 특징으로 한다.

[0012] 즉, 상기 선행기술은 데모에 의하여 경로를 주행하도록 로봇을 훈련함으로써, 사용자로 하여금 사용자가 미리 예상하지 않았던 환경을 내비게이션하도록 로봇을 훈련하는 것을 가능하게 하는 방법에 대하여 기재하고 있다.

[0013] 또한 한국등록특허 제2008367호(2019.08.07.)는 인공지능 플래닝을 이용한 자율주행형 이동로봇과 이를 이용한 스마트 실내업무 관리 시스템 및 방법에 관한 것으로, 인공지능 플래닝기술을 이용한 자율주행 지능형 이동로봇 및 앱과, 이 로봇 및 앱을 이용하여 가정 및 사업장내 필요한 곳으로 로봇을 이동시키고, 필요한 영상촬영을 하고, 이 영상을 인식하고, 인식된 결과에 따라 필요한 조치를 취하는 스마트 실내업무 관리 시스템 및 방법에 관한 것이다. 또한 실내 측위기술을 대신하여 실내 지도를 로봇 스스로 작성하고 이 지도를 이용하여 로봇의 바퀴 회전수를 카운팅하는 등의 로봇제어기술과 인공지능 플래닝 기술을 통하여 제어하게 되는 실내업무 관리 시스템 및 방법에 관한 것이다.

[0014] 즉, 상기 선행기술은 가정이나 빌딩, 공장 내 등에서의 실내업무 관리에 있어서, 실내에 미리 세팅된 무선통신 기반의 제어 시스템이 없어도 제어대상인 객체를 찾아가는 경로를 스스로 결정하여 해당 객체에 도달할 수 있고 그 객체의 상태를 인식하고 그 결과인 객체인식정보를 제어자의 스마트폰 앱에 전달할 수 있는 시스템 및 방법을 기재하고 있다.

[0015] 이상에서 선행기술들을 검토한 결과, 상기 선행기술들은 데모에 의하여 경로를 주행하도록 로봇을 훈련하는 구성, 제어대상인 객체를 찾아가는 경로를 스스로 결정하여 해당 객체에 도달하는 구성 등을 제시하고 있고, 본 발명과 비교해 볼 때 인공지능을 통해 경로의 이동을 훈련하는 점에서 일부 유사하지만, 무인 이동체의 자율 주행을 위한 DQN 기반의 맵리스 내비게이션을 구현하여 학습 시간의 단축을 위해 점차 감소하는 타겟 사이즈를 이용하고, 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 SLAM에 적용하여 맵 상에 표시하는 기술적 특징을 제시하는 것으로서, 이와 관련된 구성에 대해서는 상기 선행기술에 아무런 기재나 그 어떠한 암시도 없기 때문에 상기 선행기술과 본 발명은 기술적 차이점이 분명한 것이다.

[0016] 특히 DQN 기반의 맵리스 내비게이션을 구현하여 학습 시간을 감소시키기 위해서 타겟의 크기를 최초에 2배로 조정한 후 점차 감소(decreasing target size)시키는 구성, 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 SLAM에 적용하여 맵 상에 표시하기 위해서 라이더(LiDAR) 데이터를 오프셋하는 구성은 상기 선행기술에서 전혀 제시되지 않은 본 발명만의 특징적인 기술적 구성이다.

발명의 내용

해결하려는 과제

[0017] 본 발명은 상기와 같은 문제점을 해결하기 위해 창작된 것으로서, 무인 이동체의 자율 주행을 위한 DQN 기반의 맵리스 내비게이션을 구현하여, 상기 무인 이동체의 SLAM을 자율적으로 수행할 수 있도록 하는 시스템 및 그 처리 방법을 제공하는 것을 목적으로 한다.

[0018] 또한 본 발명은 DQN 기반의 맵리스 내비게이션을 구현하여 타겟의 크기를 최초에 2배로 조정한 후 점차 줄여가면서 학습 시간을 감소시키고, 영상인식과 거리측정을 통해 장애물의 종류, 속도 등의 특성을 파악한 후 상기 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 SLAM에 적용하여 맵 상에 표시할 수 있도록 하는 시스템 및 그 처리 방법을 제공하는 것을 다른 목적으로 한다.

[0019] 또한 본 발명은 상기 무인 이동체가 여러 장애물이 존재하고 사람과 같이 협업하는 스마트 팩토리 환경에서 사람이나 장애물과의 충돌 없이 안전거리를 고려하여 타겟 지점까지 최적의 경로로 주행할 수 있도록 하는 시스템 및 그 처리 방법을 제공하는 것을 또 다른 목적으로 한다.

과제의 해결 수단

[0020] 본 발명의 일 실시예에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템은, 사람, 물체 또는 이들의 조합을 포함한 장애물이 위치한 소정의 공간에서, 현재 상태에서 다음 상태로의 무인 이동체에 대한 액션을 출력하기 위하여, 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리 데이터, 상기 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 학습하는 학습 모듈; 상기 학습을 수행하는 과정에서, 상기 장애물이 감지되면 상기 장애물의 종류, 속도 또는 이들의 조합을 포함한 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하는 라이더 데이터 오프셋 모듈; 및 상기 오프셋한 라이더 데이터를 입력받아 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시하는 맵 생성 모듈;을 포함하는 것을 특징으로 한다.

[0021] 또한 상기 학습 모듈은, 각 에피소드에 따른 학습을 수행할 때 각 스텝 타임마다 상기 무인 이동체에서 측정한 라이더 데이터를 참조하여 액션을 선택하는 것을 특징으로 한다.

[0022] 이때 상기 에피소드는, 상기 무인 이동체가 타겟 지점에 도착하거나, 벽, 장애물 및 사람과 충돌이 발생하거나, 또는 상기 타겟 지점에 기 설정된 시간 이내에 도착하지 못했을 경우를 하나의 에피소드로 설정하며, 상기 충돌은, 상기 무인 이동체가 벽, 장애물 및 사람의 기 설정된 범위 이내로 접근하였을 때를 충돌로 판정하는 것을 특징으로 한다.

[0023] 또한 상기 학습 모듈은, 상기 에피소드를 최초에 수행할 때 상기 타겟 지점의 반경을 원래 크기의 정수 배로 확장하여 설정하고, 상기 에피소드의 수행에 따라 상기 무인 이동체가 상기 타겟 지점에 도착할 때마다 상기 타겟 지점의 반경을 기 설정된 범위로 줄여 다음의 에피소드를 수행하며, 상기 타겟 지점의 반경이 원래 크기가 될 때까지 각 에피소드를 수행함으로써, 탐색 범위를 좁혀 학습 시간을 단축하는 것을 특징으로 한다.

[0024] 또한 상기 라이더 데이터 오프셋 모듈은, 상기 무인 이동체에서 촬영한 상기 장애물의 뎁스 데이터 및 영상인식에 따른 바운딩 박스 데이터를 토대로 상기 장애물을 감지하여 상기 장애물의 종류를 확인하고, 상기 장애물이 감지되면 상기 무인 이동체에서 측정한 현재 상태와 이전 상태간의 라이더 데이터의 차이를 계산하고, 상기 계산한 라이더 데이터의 차이를 통해서 상기 장애물의 위치를 확인하며, 상기 확인한 장애물의 종류에 따라 상기 확인한 장애물의 위치로부터 설정해야 할 안전거리를 차등적으로 부여하고, 상기 장애물과, 상기 차등적으로 부여한 안전거리에 대한 라이더 데이터의 오프셋을 수행하는 것을 특징으로 한다.

[0025] 아울러, 본 발명의 일 실시예에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법은, DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 사람, 물체 또는 이들의 조합을 포함한 장애물이 위치한 소정의 공간에서, 현재 상태에서 다음 상태로의 무인 이동체에 대한 액션을 출력하기 위하여, 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리 데이터, 상기 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 학습하는 학습 단계; 상기 학습을 수행하는 과정에서, 상기 장애물이 감지되면 상기 장애물의 종류, 속도 또는 이들의 조합을 포함한 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하는 라이더 데이터 오프셋 단계; 및 상기 오프셋한 라이더 데이터를 입력받아 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시하는 맵 생성 단계;를 포함하는 것을 특징으로 한다.

[0026] 또한 상기 학습 단계는, 상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 각 에피소드에 따른 학습을 수행할 때 각 스텝 타임마다 상기 무인 이동체에서 측정한 라이더 데이터를 참조하여 액션을 선택하는 것을 특징으로 한다.

[0027] 이때 상기 에피소드는, 상기 무인 이동체가 타겟 지점에 도착하거나, 벽, 장애물 및 사람과 충돌이 발생하거나, 또는 상기 타겟 지점에 기 설정된 시간 이내에 도착하지 못했을 경우를 하나의 에피소드로 설정하며, 상기 충돌은, 상기 무인 이동체가 벽, 장애물 및 사람의 기 설정된 범위 이내로 접근하였을 때를 충돌로 판정하는 것을 특징으로 한다.

[0028] 또한 상기 학습 단계는, 상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 상기 에피소드를 최초에 수행할 때 상기 타겟 지점의 반경을 원래 크기의 정수 배로 확장하여 설정하는 단계; 상기 에피소드의 수행에 따라 상기 무인 이동체가 상기 타겟 지점에 도착할 때마다 상기 타겟 지점의 반경을 기 설정된 범위로 줄여 다음의

에피소드를 수행하는 단계; 및 상기 타겟 지점의 반경이 원래 크기가 될 때까지 각 에피소드를 수행하는 단계;를 더 포함하는 것을 특징으로 한다.

[0029] 또한 상기 라이더 데이터 오프셋 단계는, 상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에서, 상기 무인 이동체에서 촬영한 상기 장애물의 뎁스 데이터 및 영상인식에 따른 바운딩 박스 데이터를 토대로 상기 장애물을 감지하여 상기 장애물의 종류를 확인하는 단계; 상기 장애물이 감지되면 상기 무인 이동체에서 측정한 현재 상태와 이전 상태간의 라이더 데이터의 차이를 계산하고, 상기 계산한 라이더 데이터의 차이를 통해서 상기 장애물의 위치를 확인하는 단계; 및 상기 확인한 장애물의 종류에 따라 상기 확인한 장애물의 위치로부터 설정해야 할 안전거리를 차등적으로 부여하고, 상기 장애물과, 상기 차등적으로 부여한 안전거리에 대한 라이더 데이터의 오프셋을 수행하는 단계;를 더 포함하는 것을 특징으로 한다.

발명의 효과

[0030] 이상에서와 같이 본 발명의 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법에 따르면, 무인 이동체를 위한 DQN 기반의 맵리스 내비게이션을 구현하여 타겟의 크기를 최초에 2배로 조정한 후 점차 줄여 학습 시간을 감소시키고, 장애물의 종류, 속도 등의 특성을 파악하여 각 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 SLAM에 적용하여 맵 상에 표시함으로써, 상기 무인 이동체가 SLAM을 자율적으로 수행할 수 있으며, 사람과 여러 장애물이 존재하는 스마트 팩토리 환경에서 사람이나 장애물과의 충돌 없이 최단 경로로 타겟 지점에 도달할 수 있는 효과가 있다.

도면의 간단한 설명

[0031] 도 1은 본 발명의 일 실시예에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템의 구성을 나타낸 도면이다.

도 2는 본 발명이 적용된 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리과정을 상세하게 설명하기 위한 도면이다.

도 3은 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션의 실험 환경을 설명하기 위한 도면이다.

도 4는 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션에 적용된 타겟 사이즈를 설명하기 위한 도면이다.

도 5는 본 발명에 적용된 DQN 기반 학습을 수행할 때 리워드 타입 및 값을 설명하기 위한 도면이다.

도 6은 본 발명에 적용된 DQN의 구조를 설명하기 위한 도면이다.

도 7은 본 발명에 적용된 무인 이동체의 모든 출력 액션을 설명하기 위한 도면이다.

도 8과 도 9는 본 발명에 적용된 타겟 크기를 줄이는 방식과 일반적인 타겟 크기로 DQN 기반 학습을 수행할 때의 결과를 설명하기 위한 도면이다.

도 10은 본 발명에 적용된 사람이 서있거나 걸어 다니는 환경에서 DQN 기반 학습을 수행할 때의 결과를 설명하기 위한 도면이다.

도 11은 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 적용된 뎁스 카메라의 시야각을 설명하기 위한 도면이다.

도 12는 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 적용된 속도에 따른 라이더 오프셋의 차이를 설명하기 위한 도면이다.

도 13은 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 적용된 라이더 오프셋 알고리즘을 설명하기 위한 도면이다.

도 14는 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의한 영상 인식과 라이더 데이터의 오프셋을 설명하기 위한 도면이다.

도 15는 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의해 서있는 사람과 물체의 결과를 나타낸 도면이다.

도 16은 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의해 움직이는 사람과 물체의 결과를 나타낸 도면이다.

도 17은 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의해 무인 이동체가 움직이고 있는 공간에 대해 자율적인 SLAM을 수행한 결과를 나타낸 도면이다.

도 18은 본 발명의 일 실시예에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법의 동작과정을 상세하게 나타낸 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0032] 이하, 첨부한 도면을 참조하여 본 발명의 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템 및 그 처리 방법에 대한 바람직한 실시 예를 상세히 설명한다. 각 도면에 제시된 동일한 참조부호는 동일한 부재를 나타낸다. 또한 본 발명의 실시 예들에 대해서 특정한 구조적 내지 기능적 설명들은 단지 본 발명에 따른 실시 예를 설명하기 위한 목적으로 예시된 것으로, 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는 것이 바람직하다.

[0033] 도 1은 본 발명의 일 실시예에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템의 구성을 나타낸 도면이다.

[0034] 도 1에 도시된 바와 같이, 본 발명의 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템(100)은 스마트 팩토리에서 자율 주행을 수행하기 위한 로봇, 차량, 드론 등의 무인 이동체에 적용되는 구성으로서, 가상 환경 구축 모듈(110), 센서(120), 학습 모듈(130), 라이더 데이터 오프셋 모듈(140), 맵 생성 모듈(150), 제어 모듈(160), 메모리(170) 등을 포함하여 구성된다.

[0035] 또한 상기 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템(100)은 도면에 도시하지는 않았지만, 각 구성 부분에 동작전원을 공급하는 전원부, 각종 기능에 대한 데이터 입력을 위한 입력부, 각종 동작프로그램의 업데이트를 관리하는 업데이트 관리부 등을 추가로 포함할 수 있다.

[0036] 상기 가상 환경 구축 모듈(110)은 실제 무인 이동체의 환경에서 학습을 수행할 경우 안전성 및 효율성의 문제가 발생되기 때문에, 예를 들어 로봇의 프로그래밍에 사용되는 ROS(Robot Operating System)와 물리엔진 기반의 가상 시뮬레이션 툴(예: GAZEBO)을 사용하여 시뮬레이션을 구성하는 부분이다.

[0037] 즉, 상기 가상 환경 구축 모듈(110)은 멈춰있거나 이동중인 적어도 하나 이상의 사람, 적어도 하나 이상의 물체 또는 이들의 조합을 포함한 장애물의 실제 공간에 대한 가상 공간을 생성하고, 상기 생성한 가상 공간 상에 영상인식 및 거리측정을 위한 센서(120)가 구비된 무인 이동체의 스타트 지점과 타겟 지점을 설정하는 것이다.(도 3 참조)

[0038] 상기 센서(120)는 뎁스 카메라(121)와 라이더 측정기(122)를 포함하고, 영상 촬영 정보 및 거리측정 정보를 상기 학습 모듈(130)로 출력한다. 이때 상기 거리측정 정보는 상기 라이더 데이터 오프셋 모듈(140)로 출력한다.

[0039] 상기 뎁스 카메라(121)의 경우 시야각은 60도, 프레임 사이즈는 640x480으로 설정되며, 상기 라이더 측정기(122)의 경우 샘플 수는 60개, 해상도(resolution)는 6도로 설정한다. 이때 상기 해상도를 6도로 설정한 이유는 저 성능의 라이더에서도 대응이 가능하도록 하여 범용성을 높이기 위해서이다.

[0040] 또한 상기 라이더(LiDAR) 측정기(122)는 레이저로 대상물을 조사하여 반사되는 빛을 분석함으로써 거리를 측정하는 원격 감지 장치이다.

[0041] 상기 학습 모듈(130)은 상기 무인 이동체의 주변 환경에 따른 학습을 위해서 도 6과 같이 구성한 DQN을 이용하여, 상기 가상 환경 구축 모듈(110)에 구축한 서있거나 또는 움직이는 사람, 적어도 하나 이상의 물체(예: 표면이 딱딱하거나 소프트한 것으로 구분되는 장비, 고가 또는 저가의 장비 등) 또는 이들의 조합을 포함한 가상의 공간에서, 무인 이동체의 현재 상태에서 다음 상태로의 액션값을 출력하기 위하여 학습을 수행한다. 즉 각 에피소드에 따른 학습을 수행할 때, 각 스텝 타임마다 상기 무인 이동체에 구비된 라이더 측정기(122)를 통해 주변을 스캔한 데이터들을 모으고 쌓아서 만든 스택(즉 라이더에 의한 거리 데이터)을 참조하여 액션을 선택하는 것이다.

[0042] 보다 구체적으로 설명하면, 상기 학습 모듈(130)은 현재 상태에서 다음 상태로의 무인 이동체에 대한 액션을 출력하기 위하여, 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리 데이터, 상기 무인 이동체에 구비된 상기 라이더 측정기(122)에서 측정한 소정 각도별 라이더 데이터를 학습하고, 상기 학습한 결과를 상기 제어 모듈(160)로 출력하는 것이다.

[0043] 예를 들어, 상기 학습 모듈(130)은 특정 에피소드를 최초에 수행할 때 상기 타겟 지점의 반경을 원래 크기의 정수 배(바람직하게는 2배)로 확장하여 설정하고, 상기 에피소드의 수행에 따라 상기 무인 이동체가 상기 타겟 지점에 도착할 때마다 상기 타겟 지점의 반경을 기 설정된 범위(바람직하게는 1%)로 줄여 다음의 에피소드를 수행하며, 상기 타겟 지점의 반경이 원래 크기가 될 때까지 각 에피소드를 수행함으로써, 탐색 범위를 좁혀 학습 시간을 단축할 수 있도록 한다.

[0044] 이처럼, 초기 타겟 지점의 크기를 정수 배로 키우고 크기를 줄여나가면서 학습을 수행하게 되면, 상기 무인 이동체의 탐색 범위가 좁혀지고, 이에 따라 DQN을 수행하면서 타겟 지점이 어려운 위치에 존재할 때 타겟 지점을 찾지 못하고 시간이 과도하게 오래 걸리는 상황을 방지할 수 있게 된다.

[0045] 이때 상기 에피소드는 상기 무인 이동체가 타겟 지점에 도착하거나, 벽, 장애물 및 사람과 충돌이 발생하거나, 또는 상기 타겟 지점에 기 설정된 시간(예: 200초) 이내에 도착하지 못했을 경우를 하나의 에피소드로 설정한다. 또한 상기 충돌은 상기 무인 이동체가 벽, 장애물 및 사람의 기 설정된 범위 이내로 접근하였을 때를 충돌로 판정한다.

[0046] 상기 라이더 데이터 오프셋 모듈(140)은 상기 학습 모듈(130)을 통해 학습을 수행하는 과정에서, 상기 장애물이 감지되면 영상인식 및 거리측정 정보를 토대로 상기 장애물의 종류, 속도 또는 이들의 조합을 포함한 특성을 확인하고, 상기 확인한 장애물의 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하고, 상기 오프셋한 라이더 데이터를 상기 맵 생성 모듈(150)로 출력한다.

[0047] 즉 상기 라이더 데이터 오프셋 모듈(140)은 사람이나 여러 물체를 포함한 장애물을 검출하고, 상기 검출한 장애물의 종류에 따라 안전거리의 가중치를 변경하여 라이더 데이터를 오프셋하고, 이를 토대로 상기 맵 생성 모듈(150)에서 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시할 수 있도록 함으로써, 상기 학습 모듈(130)에서 상기 무인 이동체가 안전하게 최적의 경로로 이동할 수 있도록 학습하는 데 도움을 주는 것이다.

[0048] 이를 구체적으로 설명하면, 상기 라이더 데이터 오프셋 모듈(140)은 상기 무인 이동체의 뎁스 카메라(121)에서 촬영한 상기 장애물의 뎁스 데이터 및 영상인식에 따른 바운딩 박스 데이터를 토대로 상기 장애물을 감지하여 상기 장애물의 종류를 확인한다. 이때 본 발명에서는 상기 영상인식을 위한 알고리즘으로 You Only Look Once(YOLO) 네트워크를 사용한다.

[0049] 그리고 상기 라이더 데이터 오프셋 모듈(140)은 상기 장애물이 감지되면 상기 무인 이동체의 라이더 측정기(122)에서 측정한 현재 상태와 이전 상태간의 라이더 데이터의 차이를 계산한 다음, 상기 계산한 라이더 데이터의 차이를 통해서 상기 장애물의 위치를 확인한다.

[0050] 그리고 상기 라이더 데이터 오프셋 모듈(140)은 상기 확인한 장애물의 종류에 따라 상기 확인한 장애물의 위치로부터 설정해야 할 안전거리를 차등적으로 부여하고, 상기 장애물과, 상기 차등적으로 부여한 안전거리에 대한 라이더 데이터의 오프셋을 수행한다.

[0051] 이때 상기 오프셋된 라이더 데이터는 ROS용의 데이터 메시지 폼으로 변환되어 상기 맵 생성 모듈(150)로 출력된다.

[0052] 상기 맵 생성 모듈(150)은 상기 라이더 데이터 오프셋 모듈(140)로부터 오프셋한 라이더 데이터를 입력받아 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시한다.

[0053] 즉 상기 라이더 데이터 오프셋 모듈(140)로부터 제공받은, 장애물이 감지된 지점을 기준으로 장애물의 위치로부터 설정해야 할 경계면의 거리만큼 오프셋을 수행한 정보를 SLAM에 적용하여 장애물에 대한 경계면이 맵 상에 표시되도록 하는 것이다.

[0054] 이때 상기 장애물에 대한 경계면이 업데이트된 맵은 상기 학습 모듈(130)에서 다음 번 에피소드를 수행할 때 상기 무인 이동체의 현재 상태에서 다음 상태로의 액션을 출력하기 위한 학습에 이용된다.

[0055] 상기 제어 모듈(160)은 상기 가상 환경 구축 모듈(110), 센서(120), 학습 모듈(130), 라이더 데이터 오프셋 모듈(140) 및 맵 생성 모듈(150)의 동작을 총괄적으로 제어하는 부분이다.

[0056] 상기 메모리(170)는 상기 무인 이동체에서 사용되는 각종 동작프로그램을 저장하고 있다.

[0057] 또한 상기 메모리(170)는 상기 학습 모듈(130)에서 수행한 학습 모델을 저장하고 있으며, 상기 맵 생성 모듈(150)을 통해 구현된 SLAM 정보를 저장한다.

[0058] 이와 같이 구성된 본 발명의 DQN 및 SLAM 기반의 맵리스 내비게이션 처리과정을 도 2를 참조하여 상세하게 설명하면 다음과 같다.

[0059] 도 2는 본 발명이 적용된 DQN 및 SLAM 기반의 맵리스 내비게이션 처리과정을 상세하게 설명하기 위한 도면이다.

[0060] 도 2에 도시된 바와 같이, 상기 학습 모듈(130)은 각 에피소드별로 사람, 물체 또는 이들의 조합을 포함한 장애물이 포함된 소정의 공간에서 타겟 지점까지의 각도 및 거리 정보, 소정 각도별 라이더 데이터를 입력으로 하여 학습을 수행하고, 학습 결과에 따라 상기 무인 이동체의 현재 상태에서 다음 상태로의 액션값을 출력한다.

[0061] 또한 상기 라이더 데이터 오프셋 모듈(140)은 상기 학습 모듈(130)에 의한 학습이 진행되는 과정에서, 장애물이 감지되면 상기 뎁스 카메라(121)로 촬영한 로우 뎁스 데이터(depth_raw)와, 로우 이미지 데이터(image_raw)로부터 영상인식을 수행한 바운딩 박스(bounding box) 데이터와, 라이더 측정기(122)에서 측정한 소정 각도별 라이더 데이터를 입력으로 하여 상기 감지된 장애물의 종류, 속도 등의 특성을 확인한다.

[0062] 또한 상기 라이더 데이터 오프셋 모듈(140)은 상기 확인한 장애물의 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하고, 상기 오프셋한 데이터를 ROS로 변환하여 상기 맵 생성 모듈(150)로 출력한다.

[0063] 이때 상기 사전에 설정한 안전거리는 움직이는 사람, 서 있는 사람, 고가 장비, 저가 장비의 순서로 점차 줄어들도록 설정된다.

[0064] 이에 따라 상기 맵 생성 모듈(150)은 상기 오프셋 정보를 SLAM에 적용하여 장애물에 대한 경계면을 맵 상에 표시한다.

[0065] 이와 같이 상기 무인 이동체의 액션 출력을 위한 학습 및 장애물의 특성에 따른 안전거리 가중치의 차등 적용을 토대로 최종적인 학습 모델과 각종 장애물 특성이 고려된 맵이 완성된다.

[0066] 다음에는, 이와 같이 구성된 본 발명에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션에 대하여 도 3 내지 도 10을 참조하여 보다 상세하게 설명한다.

[0067] 도 3은 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션의 실험 환경을 설명하기 위한 도면으로서, 본 발명에서는 우선 도 3의 (a)에 나타낸 바와 같이 뎁스 카메라(121)와 라이더 측정기(122)가 구비된 실제 세계(real world)의 무인 이동체를 가상의 공간 상에 구현한다.

[0068] 또한 도 3의 (b)에 나타낸 바와 같이 소정 크기(예: 6m x 6m)의 직사각형 가상 공간 내부에 고정된 원통형의 장애물 2개를 배치한다. 본 발명의 실험에서는 상기 장애물의 크기는 반지름 0.3m로 설정하고, 약 2m 간격으로 설치하며, 타겟 지점(goal position)의 좌표는 장애물 주변의 미리 지정된 8개의 좌표에서 랜덤하게 설정한다.

[0069] 또한 서 있는 사람이나 움직이는 사람 이외에, 다양한 장애물을 가상환경에서 구현하는 것이 쉽지 않기 때문에, 본 발명에서는 도 3의 (c)에 나타낸 바와 같이 고가의 장애물은 모니터, 저가의 장애물은 컵으로 대체하여 이미지화한 후 배치한다.

[0070] 상기 도 3에서 구현한 가상 환경을 토대로, 상기 학습 모듈(130)에 적용되는 DQN을 설명하면 다음과 같다.

[0071] 강화학습은 학습 및 제어의 주체인 에이전트의 최적화 제어(Optimal Control) 문제를 위하여 Bellman이 Markov Decision Process(MDP)를 기반으로 만든 Bellman Equation을 기초로 한다. 먼저 상기 MDP는 처음 시간 t = 0에서 어떠한 상태 S_0로부터 시작하여 현재의 상태 S_t로 도착할 확률이 바로 이전의 상태 S_{t-1}에서 현재의 상태 S_t까지 올 확률과 같다면 이는 Markov하다고 표현하며, 아래의 수학식 1로 표현할 수 있다

[0072] [수학식 1]

[0073]
$$P\{R_t = r, S_t = s' \mid S_0, A_0, R_1 \cdots\cdots S_{t-1}, A_{t-1}\}$$
$$= P\{R_t = r, S_t = s' \mid S_{t-1}, A_{t-1}\}$$

[0074] 여기서 액션 A_t는 에이전트가 취하는 액션을 나타내며, 리워드 R_t는 에이전트가 액션을 취한 후의 환경, 즉 주변 환경을 고려하였을 때 얼마나 의도한대로 움직였는지를 수치적으로 나타낸다. 강화학습은 일반적으로 상태 S_t가 이전의 상태 S_{t-1}과 Markov한 관계를 가진다고 가정하고 학습을 수행하기 때문에 한 에피소드가 끝난 후 받은 리워드는 이전의 상태들과 관련이 있다고 할 수 있으며, 위의 관계를 이용하여 DQN의 학습을 위한 가치 함수(Value Function)를 유도한다. 에이전트는 각 스텝마다 액션을 취했을 때 외란 등에 의해 상태 S_t에 확률적으로 도달하게 되며 이 상태에 따른 리워드를 즉시 얻게 된다.

[0075] 한 에피소드가 끝났을 때 매 타임 스텝에서 받은 리워드의 총 합이 높을수록 학습이 잘 되었다고 할 수 있으며, 에이전트의 상태에 따라 얻을 수 있는 리워드 총합의 기댓값은 State Value Function이라 표현한다.

[0076] State Value Function은 수학식 2와 같이 나타낼 수 있다.

[0077] [수학식 2]

[0078] $$v(s) = E[G_t \mid S_t = s]$$

[0079] 기댓값으로 나타내는 이유는 에이전트가 상태 S_t에 도달하는 것은 확률에 따르기 때문이며, G_t는 환경으로부터 얻은 리워드의 총합을 나타내고 수학식 3과 같이 나타낼 수 있다.

[0080] [수학식 3]

[0081] $$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \text{ (단, } 0 < \gamma < 1 \text{)}$$

[0082] 이때 위의 식에서 γ는 디스카운트 팩터(Discount Factor)를 나타내며 리워드 시간에 따른 차이를 부여한다. 단순히 각 시간마다 얻은 리워드를 더할 경우 시간이 무한대로 감에 따라 리워드의 크기를 다르게 받더라도 반드시 무한대로 수렴하기 때문에 해당 리워드를 받은 때가 시작하자마자 받은 것인지, 미래에 받은 것인지 비교가 불가능하다. DQN은 MDP를 기반으로 하여 미래의 리워드를 확률적으로 얻을 수 있기 때문에 미래의 리워드의 경우 실제로 받을 수 있을지 확실한 것이 아니다. 따라서 Discounting Factor를 사용해 시간에 따른 리워드를 차등 지급함으로써 미래에 받을 것이라 예상되는 리워드에 대한 영향력을 감소시킨다. Discounting Factor는 0에서 1사이로 책정되는데 지나치게 낮을 경우 미래의 리워드를 아예 반영하지 못하므로 본 발명에서는 0.9로 설정한다.

[0083] 수학식 2의 State Value Function은 상태와 리워드의 관계만 고려되어 있으나, 에이전트의 액션을 결정하는 정책 또한 반드시 고려되어야 한다. 이는 정책 π가 시간 t에서 에이전트가 상태 S_t를 가질 때 액션 A_t를 취할 확률을 나타내기 때문이며, 이를 수식으로 나타내면 아래의 수학식 4와 같이 표현할 수 있다.

[0084] [수학식 4]

[0085] $$\pi(a \mid s) = P(A_t = a \mid S_t = s)$$

[0086] 에이전트가 정책 $\pi(a \mid s)$를 가질 때 이를 고려한 State Value Function은 수학식 5와 같이 π를 표기하여 나타낼 수 있다.

[0087] [수학식 5]

[0088] $$v_\pi(s) = E_\pi[G_t \mid S_t = s]$$

[0089] Q 러닝의 기반이 되는 Bellman Equation은 다음 시간의 상태 S_{t+1}과 현재 상태 S_t의 State Value Function 관계를 수식으로 정의한 것인데, 위의 리워드의 총합에 관한 수식 및 Value Function을 사용하여 이를 구할 수 있다. 수학식 3을 수학식 5에 대입하면 아래의 수학식 6과 같이 나타낼 수 있다.

[0090] [수학식 6]

$$v_\pi(s) = E_\pi[G_t \mid S_t = s]$$
$$= E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots\ldots \mid S_t = s]$$

[0091]
$$= E_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots\ldots) \mid S_t = s]$$

[0092] 상기 수학식 6에서 $R_{t+2} + \gamma R_{t+3} + \ldots\ldots$ 은 수학식 3의 관계를 이용하면 다음의 수학식 7과 같이 만들 수 있다.

[0093] [수학식 7]

[0094]
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

[0095] 상기 수학식 7에서 G_{t+1}은 $v_\pi(S_{t+1})$로 나타낼 수 있으므로 이를 대입하면 Bellman Equation은 최종적으로 수학식 8과 같이 정리가 가능하다.

[0096] [수학식 8]

[0097]
$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

[0098] 상기 수학식 8에서 R_{t+1}은 현재의 상태 S_t가 주어졌을 때 환경으로부터 즉시 받는 리워드이며, $\gamma v_\pi(S_{t+1})$는 시간 t + 1에서 상태 S_{t+1}이 주어졌을 때 받을 수 있는 Discounted State Value Function이다.

[0099] 여기서 Expectation은 어떠한 액션 a를 취할 확률을 나타내는 정책에 의존하기 때문에 나온 것이라고 할 수 있다. 확률과 기대 이득의 곱의 합이라는 Expectation의 정의에 따라 수학식 8은 다음의 수학식 9와 같이 표현할 수 있다.

[0100] [수학식 9]

[0101]
$$v_\pi(s) = [R_{t+1} + \gamma \sum P(S_{t+1} \mid S_t, A_t) v_\pi(s_{t+1}) \mid S_t = s]$$

[0102] DQN을 수행하면서 에이전트는 이러한 Value Function을 이용해 해당 시간의 상태 S_t에서 최고의 리워드를 얻을 수 있는 상태 S_{t+1}로 이동시켜야 한다. 그런데 상기 수학식 9에서 식 $P(S_{t+1} \mid S_t, A_t)$는 에이전트가 환경에 대하여 확률적으로 움직이는 것을 뜻하며 이는 환경에 대한 모델링을 반드시 필요로 한다. 다시 말하자면 에이전트는 다음의 시간 t+1에서 가능한 모든 상태에 대한 리워드 값을 알아야 하며 동시에 그 상태로 이동하기 위한 액션 및 확률 또한 알아야 한다는 것을 뜻한다. 이것은 모델에 의한 결정론적인 방법이라고 할 수 있는데 실제 환경에 대한 모델링을 하는 것은 매우 어려운 일이다.

[0103] 따라서 DQN에서는 환경에 대한 모델링이 필요 없는 모델 프리(Model Free) 방법을 위하여 Action Value Function을 사용한다. 현재의 상태 S_t에서 액션을 취한 후의 Value Function을 구하면 다음 시간의 모든 가능한 상태 S_{t+1}에 대한 모든 Value Function을 구할 필요가 없어지므로 식을 간략화 할 수 있는데, 이를 Action Value Function 혹은 Q Function라고 하며 수학식 10과 같이 나타낸다.

[0104] [수학식 10]

[0105]
$$Q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

[0106] Action Value Function 또한 앞서 State Value Function과 같은 방식으로 수학식 11과 같이 정리한다.

[0107] [수학식 11]

$$Q_\pi(S_t, A_t) = E_\pi[G_t]$$

[0108] $$= R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})$$

[0109] 상기 Action Value Function은 상태 S_t 및 액션 A_t에 따라 결정되며 강화학습에서 사용되는 Model Free 방법에서는 확률이 아닌 경험을 기반으로 하기 때문에 수학식 11과 같이 정리된다. 해당 수학식에서 받을 수 있는 최대의 값인 Optimal Action Value Function $Q_*(S_t, A_t)$는 다음의 수학식 12로 정의할 수 있다.

[0110] [수학식 12]

[0111] $$Q_*(S_t, A_t) = \max Q_\pi(S_t, A_t)$$

[0112] 여기서 $Q_*(S_t, A_t)$는 상기 수학식 12에 따라 현재 상태 S_t를 알 때 에이전트가 취할 수 있는 액션 중 $Q_\pi(S_t, A_t)$를 최대로 하는 액션 A_t를 통해 구할 수 있으며 이 Optimal Action Value Function을 얻을 수 있는 정책을 최적 정책(Optimal Policy)이라고 한다. 최적 정책은 다음의 수학식 13과 같이 표현할 수 있다.

[0113] [수학식 13]

[0114] $$\pi_*(A_t|S_t) = A_t \text{ (단, } A_t = \operatorname{argmax} Q_*(S_t, A_t))$$

[0115] 여기서 최적 정책은 Action Value Function을 최대로 하는 액션 A_t를 선택하게 되는데 이를 greedy 방법이라 한다. 상기 설명을 정리하면 DQN은 Action Value Function을 인공신경망으로 구현하고 현재의 상태 S_t 및 최적 정책에 따라 최대의 Q Value 혹은 리워드를 가지는 액션 A_t를 선택하는 것이다. 따라서 DQN의 목적은 이 Q Function을 나타내는 인공 신경망을 구성하고 학습하는 것이라고 할 수 있다.

[0116] 이러한 Q Function은 액션의 리워드를 평가하는 Evaluation 및 신경망의 업데이트를 나타내는 Improvement의 반복 과정인 Policy Iteration을 통해 구할 수 있다.

[0117] Value Evaluation은 시간 t에서 주어진 정책 및 상태에 대하여 얻을 수 있는 리워드의 예측 값을 계산 하는 것을 뜻하며 이 값을 이용해 Q Function에 대한 업데이트를 수행 한다. 이 과정을 Improvement라고 칭하며 DQN에서는 $\epsilon-greedy$ 방법이 사용된다. 이 Policy Iteration에는 크게 두 가지 방법이 사용되는데, 먼저 Dynamic Programming으로 대표되는 Model 기반의 방법은 한 에피소드 내의 상태에 따른 Bellman Equation을 먼저 전부 계산 후에 Optimal Value를 구하게 되며, 이에 따라 타임 스텝마다 마다 업데이트 할 수 있다는 장점이 있다. 그러나 Dynamic Programming의 경우 처음부터 모든 환경에 대한 상태들을 알아야 하므로 모르는 지점을 찾아가기 위해 본 발명에서는 Model Free 방법 중 하나인 Temporal Difference Method를 사용한다. 다른 Model Free 방법 중 하나인 Monte Carlo Method 방법에서는 Value Function을 구하기 위해 우선 한 에피소드를 진행하면서 각 시간에서의 상태 및 리워드를 시계열 데이터로써 메모리에 모두 저장하고 그 후에 메모리에 저장된 데이터로 Value Function을 계산하고 Policy를 업데이트한다. 하지만 에피소드의 종료까지 시간이 과도하게 소요될 경우 업데이트가 너무 늦게 되어 학습이 힘들다는 단점이 존재하며 학습을 이제 막 시작하는 무인 이동체에서는 타겟 지점에 도착하기까지 시간이 오래 소요될 가능성이 있다.

[0118] 이와 반대로 Temporal Difference Method는 Dynamic Programming처럼 타임 스텝마다 업데이트가 가능하다는 장점이 있다. Monte Carlo Method은 하나의 에피소드가 종료되었을 때 즉, 정확한 목표 지점에 도착해야만 업데이트를 시작하므로 정답에 대한 학습이 가능하지만 그로 인하여 도착이 늦기 때문에 비교적 Variance가 높다. 반대로 Temporal Difference는 목표에 도착하기 전에도 계속 학습을 수행하므로 Variance가 크지는 않으나 가는 도중에도 학습해야 하므로 정확한 목표를 사용할 수 없어 Bias가 비교적 높다. 따라서 Temporal Difference를 사용할 때 매 타임 스텝이 아닌 Interval의 적절한 선택을 통해 Variance와 Bias를 고려해 Trade Off한다.

[0119] 한편, Temporal Difference Method에서 Q Function을 업데이트하고 사용하는 방법에는 크게 On-Policy와 Off-Policy의 2가지 방법이 존재한다. On-Policy에서는 에이전트를 제어하는데 사용되는 Policy와 업데이트 하는 Policy를 따로 구분하여 사용하지 않으며, 이 경우에는 하나의 Policy에 따라 액션을 수행하고 이 결과에 대한

학습을 반복한다. 이에 따라 탐험을 수행하기 힘들고 이 때문에 Local Minimum에 빠질 위험이 높으며 학습초기에 업데이트와 액션이 같은 신경망을 이용해 진행되기 때문에 업데이트에 의한 가중치의 변화가 심하여 안정적인 학습이 어렵다.

[0120] 이러한 단점을 개선하기 위해 하나의 Policy를 두 가지로 나눈 것이 Off-Policy인데, 이는 에이전트의 제어를 위한 behavior Policy 외에 Target Policy를 따로 두고 업데이트를 수행하는 방법을 뜻하며, 이것이 일반적인 Q Learning의 구조라고 할 수 있다.

[0121] 이러한 구조의 장점은 behavior Policy와 다른, Target Policy를 목표로 업데이트함과 동시에 에이전트가 액션을 임의로 수행하며 경험을 쌓고 환경에 대한 탐험을 할 수 있다는 점이다.

[0122] 먼저 Q Learning에서 Target Policy는 가장 큰 Q Value를 가지는 값을 선택하여 업데이트하는 greedy 방법을 사용한다. 이를 적용한 Q의 Target Value는 다음의 수학식 14와 같이 표현할 수 있다.

[0123] [수학식 14]

[0124] $$\pi_{target}(S_{t+1}) = \mathrm{argmax}\, Q(S_{t+1}, A_{t+1})$$

[0125] 이를 수학식 11에 대입하면 다음의 수학식 15와 같이 정리할 수 있다.

[0126] [수학식 15]

[0127] $$\begin{aligned} Q_{target}(S_t, A_t) &= R_{t+1} + \gamma\, Q(S_{t+1}, A_{t+1}) \\ &= R_{t+1} + \gamma\, Q(S_{t+1}, \mathrm{argmax}\, q(S_{t+1}, A_{t+1})) \\ &= R_{t+1} + \gamma \max Q(S_{t+1}, A_{t+1}) \end{aligned}$$

[0128] 상기 수학식 11의 정의 및 Target Policy의 역할에 따라 수학식 15는 인공 신경망의 Target Value가 된다. 하지만 이 방법을 사용할 경우 신경망은 항상 바로 앞의 Max Q Value만을 교육하기 때문에 미래에 더 높은 리워드를 받을 수 있는 액션에 대해 탐색이 불가능하며 다양한 환경에서 적용이 힘들다. 따라서 Q Learning에서는 이를 바로 업데이트에 이용하는 것이 아니라 $\epsilon-greedy$ 방법을 통해 좀 더 넓은 환경을 탐색하고 업데이트를 수행한다.

[0129] $\epsilon-greedy$ 는 Target Policy에 의한 액션의 선택 외에 임의의 상수와 1 이하의 상수인 ϵ 을 이용하여 확률적으로 임의의 행동을 취함으로써 greedy 방법으로는 관찰할 수 없는 주변 환경을 탐험하는 것을 뜻한다. 이때 얻은 Observed States로 Policy의 업데이트를 수행하는 것이다. 이것은 개념적으로는 에이전트가 액션을 취하였으나 외란 등에 의하여 의도한 곳과 다른 곳에 도착했음을 표현한다. 에피소드가 지나감에 따라 ϵ 은 1에서 시작해 0이 아닌 양의 실수까지 특정한 비율로 감소하게 되며 이를 통한 $\epsilon-greedy$ 방법은 수학식 16 및 17로 표현할 수 있다.

[0130] [수학식 16]

[0131] $$\pi_{behavior}(S_{t+1}) = \begin{cases} \mathrm{argmax}\, Q(S_t, A_t) & (\epsilon_t < Random\ Constant) \\ A_{random} & (\epsilon_t > Random\ Constant) \end{cases}$$

[0132] [수학식 17]

[0133] $$\epsilon_{t+1} = \epsilon_t \times 0.99 (\text{단},\ \epsilon_0 = 1\ ,\ 0.05 < \epsilon)$$

[0134] 상기 수학식 16에서 ϵ_t 는 시간에 따른 ϵ 의 변화를 나타내며 이는 에피소드의 진행에 따라 최소 0.05까지 감소된다. 따라서 에이전트의 액션은 기본적으로 Q Value를 최대로 하는 것을 선택하되 확률적으로 ϵ 에 따라 임의의 액션을 취하여 탐험을 수행하게 된다.

[0135] 상기 $\epsilon-greedy$ 방법을 통해 에이전트의 액션이 정해지면 수학식 15에 대입하여 $Q(S_{t+1}, A_{t+1})$를 구하고 다음의 수학식 18에 의해 Policy를 업데이트 한다.

[0136] [수학식 18]

[0137] $$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

[0138] 상기 수학식 18에서 $R_{t+1} + \gamma V(s_{t+1})$은 Q Learning을 위한 Target이 되며 그에 대한 Error는 $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$와 같이 나타낼 수 있다.

[0139] 한편 DQN에서는 성능을 높이기 위해 몇 가지 알고리즘이 추가된다. Experience Replay는 DQN이 발전할 수 있도록 한 방법 중 하나로 각 시간에서의 데이터를 바로 신경망에 업데이트하는 것이 아니라 우선 메모리에 저장을 한 후, 데이터가 어느 정도 쌓였을 때 Batch Size만큼의 임의의 샘플을 선택하여 학습한다. 무인 이동체가 주행하면서 얻어진 시계열 데이터를 바로 신경망의 입력으로 사용하여 학습할 경우 유사한 값과 형태의 데이터가 연속적으로 입력 및 학습되어 네트워크가 Local Minimum에 빠질 위험성이 있기 때문이다. 따라서 본 발명에서 DQN을 구현할 때에도 이를 따랐으며 Batch Size는 64로 설정한다. 그리고 학습의 시간 단축을 위해 도 4에 표기한 타겟 사이즈의 변화 방법을 사용한다.

[0140] 도 4는 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션에 적용된 타겟 사이즈를 설명하기 위한 도면이다.

[0141] 도 4에 도시된 바와 같이, DQN에서는 상기 설명한 바와 같이 탐험과 탐색을 통하여 리워드를 가장 많이 받을 수 있는 Optimal Policy를 찾는다. 다만 이 과정에서 Trial and Error(직접 부딪치며 배우는 것)를 반복하는 강화학습의 특성으로 인해 시간이 과다하게 소요될 가능성이 존재한다. 따라서 본 발명에서는 타겟 지점의 범위를 넓혀서 에이전트가 탐색하는 범위를 좁히는 방안을 제시한다.

[0142] 이 방법은 Epsilon Greedy 방법과 비슷하게 교육 초기에는 도 4와 같이 초기 타겟 지점의 크기를 2배로 키워 무인 이동체의 탐색 범위를 좁힌다. 그리고 무인 이동체가 타겟 지점으로 도착함으로 인해 에피소드가 종료됨에 따라 타겟 지점의 사이즈를 점차 감소시켜서 결국에는 기존의 타겟 지점의 크기와 동일하도록 한다. 예를 들어 본 발명은 상기 도 3과 같이 구현된 가상의 공간에서 초기 타겟 지점의 크기를 2배로 조정하였으며, 상기 무인 이동체가 각 에피소드에 의한 학습에 따라 타겟 지점에 도착할 때마다 1%씩 감소시켜 학습을 진행한다. 상기 학습은 원래의 타겟 크기가 될 때까지 진행된다. 이는 무인 이동체가 DQN을 수행하면서 타겟 지점이 어려운 위치에 존재할 때 타겟 지점을 찾지 못하고 시간이 과도하게 오래 걸리는 상황을 방지하기 위함이다.

[0143] 또한, 본 발명의 실험 환경에서는 도 5에 나타낸 바와 같이 환경에 따른 리워드를 설정한다.

[0144] 도 5는 본 발명에 적용된 DQN 기반 학습을 수행할 때 리워드 타입 및 값을 설명하기 위한 도면으로서, 본 발명에서는 주행 리워드(driving reward), 타겟 도착 리워드(goal reward), 충돌 리워드(collision reward)로 구분한다.

[0145] 상기 주행 리워드는 매 시간 t에서 받는 리워드로서, 타겟 지점과 무인 이동체의 헤딩 방향에 따라 Heading Reward 및 Distance Reward를 더하여 계산한다. 시간 t에서 상기 무인 이동체와 타겟 지점이 이루는 각도를 $\theta_{goal}(t)$, 상기 무인 이동체와 타겟 지점간의 거리를 $d_{goal}(t)$ 이라할 때 주행 리워드는 다음의 수학식 19 내지 21로 나타낼 수 있다.

[0146] [수학식 19]

[0147] $$Heading\ Reward = 5 \times (1 - \theta_{goal}(t))$$

[0148] [수학식 20]

[0149] $$Distance\ Reward = -2 / d_{init} \times d_{goal} + 2$$

[0150] [수학식 21]

[0151] $$Driving\ Reward = Heading\ Reward + Distance\ Reward$$

[0152] 상기 수학식 19에서 θ_{goal} 은 상기 무인 이동체와 타겟 지점이 이루는 각도를 의미하고, 수학식 20에서 d_{init}는 에피소드가 시작하였을 때 상기 무인 이동체와 타겟 지점간의 거리를 나타낸다.

[0153] 한편, 본 발명에서는 상기 무인 이동체의 주변 환경에 따른 주행을 학습하기 위해서 도 6의 네트워크를 구성한다.

[0154] 도 6은 본 발명에 적용된 DQN의 구조를 설명하기 위한 도면으로서, 네트워크의 입력은 라이더의 스캔 데이터와 타겟 지점까지의 거리 및 방향이다.

[0155] 본 발명의 인공 신경망은 도 6에서와 같이 신경망들을 서로 직접적으로 연결하는 Fully Connected Network 및 Dropout Layer를 사용하여 구성한다.

[0156] 상기 Drop Out Layer는 인공 신경망을 교육할 때 의도적으로 일부 뉴런에서 다음 뉴런으로 데이터가 전파되는 것을 차단한다. 이는 신경망이 학습에 사용되는 러닝 데이터 세트(Learning Data Set)에 대해 오버피팅(Over Fitting)되는 것을 방지하며, 본 발명에서는 20%로 설정한다. 각 신경망은 가우시안 분포로 초기화되며, 입력 레이어(Input Layer) 및 히든 레이어(Hidden Layer)의 활성화 함수는 ReLU, 출력 레이어(Output Layer)에서는 활성화 함수로 Linear 함수를 사용하여 결과를 출력한다.

[0157] DQN의 입력으로 사용되는 관찰된 상태(Observed States)는 라이더(LiDAR)를 통해 얻은 6도 간격의 측정 거리, 그리고 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리로서, 62개의 데이터가 입력된다. 출력 레이어는 상기 무인 이동체의 액션을 결정하며, 결정된 액션에 따른 상기 무인 이동체의 속도는 도 7과 같다.

[0158] 도 7은 본 발명에 적용된 무인 이동체의 모든 출력 액션을 설명하기 위한 도면으로서, 0 내지 4의 5개의 액션별 선속도(linear velocity)와 각속도(angular velocity)가 설정되어 있다.

[0159] 업데이트의 방식을 결정하는 Optimizer는 Stochastic Gradient Descent의 한 종류인 RMSprop를 사용한다. Stochastic Gradient Descent는 모든 데이터를 계산하는 기존의 Gradient Descent와 달리 Mini Batch Size를 사용해 업데이트 속도를 빠르게 한 것이며 Geoffrey Hinton이 제안한 방법에서 기인한다.

[0160] 일반적인 Gradient Descent의 경우 i번째 히든 레이어의 가중치를 w_i, 입력을 x_i, 출력 레이어의 출력을 out_i 그리고 E는 Cost Function이라 할 때 다음의 수학식 22 내지 24를 이용하여 인공 신경망의 가중치를 업데이트한다.

[0161] [수학식 22]

[0162] $$net_i = \sum w_i x_i$$

[0163] [수학식 23]

[0164] $$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial out_i} \frac{\partial out_i}{\partial net_i} \frac{\partial net_i}{\partial w_i}$$

[0165] [수학식 24]

[0166] $$w_i^+ = w_i - \eta \frac{\partial E}{\partial w_i}$$

[0167] 상기 수학식 22는 i번째 히든 레이어의 출력을 나타내며, 수학식 23은 Cost Function과 가중치의 상관관계를 나타낸다. 가중치에 대한 Cost Function의 변화량이 계속 감소하는 방향으로 가중치를 업데이트하면 최적의 가중치를 찾을 수 있다는 것에 기인하며 이는 수학식 24와 같이 표현된다. 수학식 24에서 η는 Learning Rate를 뜻하는데 이때 각각의 가중치를 업데이트하면서 가중치의 변화량에 따라 이를 변화시키는 것이 RMSprop이다. 기본적인 개념은 많이 변화한 가중치는 Optimal Value에 상대적으로 가까울 것이므로 Learning Rate를 감소시켜

Local Minimum에 빠지는 것을 방지하고 그렇지 않은 것은 많이 변화시켜 Optimal Value에 가까이 이동하도록 하는 것이다. RMSprop를 수학식으로 나타내면 다음의 수학식 25와 같다.

[0168] [수학식 25]

[0169] $$G_{t+1} = \gamma G_t + (1-\gamma)(\frac{\partial E}{\partial w})^2 \text{ (단, } 0 < \gamma < 1 \text{)}$$

[0170] 상기 수학식 25는 가중치에 대한 Cost Function을 제곱한 값의 일정 비율을 계속 축적하여 sum of square로 나타낼 수 있는 G_t를 통해 가중치간의 변화량 크기를 비교할 수 있다.

[0171] 이를 이용해 기존의 Gradient Descent 업데이트의 수식인 수학식 24를 변경하면 다음의 수학식 26과 같이 나타낼 수 있다.

[0172] [수학식 26]

[0173] $$w_i^+ = w_i - \frac{\eta}{\sqrt{G+\alpha}}\frac{\partial E}{\partial w_i}$$

[0174] 상기 수학식 26에서 α는 10^{-4} 이하의 작은 숫자로 설정하며 Learning Rate가 0으로 나눠지는 것을 방지한다.

[0175] 도 8과 도 9는 본 발명에 적용된 타겟 크기를 줄이는 방식과 일반적인 타겟 크기로 DQN 기반 학습을 수행할 때의 결과를 설명하기 위한 도면이다.

[0176] 상기 도 8 및 도 9에서 No Decreasing은 타겟 지점 크기의 증감 없이 실험한 결과를 나타내며, Decreasing은 타겟 지점의 크기를 변화시켜가며 시뮬레이션을 수행한 결과를 나타낸다. (a)는 각 에피소드에서 얻은 리워드의 합계, (b)는 Max Q Value 그리고 (c)는 정확도를 나타낸다. 타겟 지점의 크기는 1%씩 감소하며 최종적으로는 원래 크기까지 감소하게 되는데 목표지점에 70회 도착했을 때 목표지점의 크기는 원래 크기와 같은 타겟 지점의 크기를 가진다. 도 8의 경우 에피소드 600회를 지났을 때 타겟 지점에 70회 도착하였으며, (c)에서 2가지 경우를 비교하였을 때 정확도는 에피소드 1000회에서 약 27% 차이를 나타내고 있다.

[0177] 또한 추가적으로 실험한 도 9의 결과를 도 8의 (c)의 정확도를 비교하였을 때, 도 8과 에피소드 1500 이하에서 정확도가 약 40%까지 빠르게 상승하는 결과를 보였으며, 그 이후에도 20% 이상의 차이를 보임을 확인할 수 있다.

[0178] 도 10은 본 발명에 적용된 사람이 서있거나 걸어 다니는 환경에서 DQN 기반 학습을 수행할 때의 결과를 설명하기 위한 도면이다.

[0179] 도 10에 도시된 바와 같이, 본 발명에서는 사람이 서있거나 걸어 다니는 환경에서 맵리스 내비게이션에 대한 테스트를 수행하였고, 각각의 경우에 대하여 1000회의 에피소드를 수행한 결과는 (a) 및 (b)에 나타낸 바와 같다. 여기서 타겟 지점에 도착한 모든 경우를 goal, optimal path driving의 경우는 optimal로 나타내었으며, optimal path driving은 도 10의 (c)와 같이 장애물 사이를 피해 최단 거리로 주행하는 것을 뜻하며, 이는 에피소드 종료까지의 소요 시간 및 획득한 보상의 크기를 통해 구분하였다.

[0180] 다음에는, 이와 같이 구성된 본 발명에 따른 장애물의 특성에 따른 오프셋 알고리즘에 대하여 도 11 내지 도 16을 참조하여 보다 상세하게 설명한다.

[0181] 먼저 본 발명에서는 사람 및 물체를 포함한 장애물을 검출하고 종류에 따라 안전거리의 가중치를 변경하여 무인 이동체가 안전하게 최적의 경로로 이동할 수 있도록 학습한다. 따라서 각 장애물에 대하여 영상으로 인식하기 위해 YOLO 알고리즘을 사용한다.

[0182] 이미지를 인식하기 위한 이미지를 인식하기 위한 YOLO 네트워크의 입력으로는 가상으로 구현된 뎁스 카메라로부터 입력받은 640x480 로우 이미지 데이터가 사용되며, 이를 이용해 영상 내에서 장애물들의 위치를 찾고 종류를 구분할 수 있다. 상기 YOLO의 특징은 CNN을 기반으로 하여 많고 다양한 사물들을 빠르게 구분할 수 있다는 장점이 있다. 또한 해당 장애물의 종류와 위치에 대한 데이터를 가지는 바운딩 박스(Bounding Box) 데이터를 얻을 수 있다.

[0183] 종래의 SLAM에서는 단순히 라이더의 거리 데이터만 이용하여 맵을 만들기 때문에 공간의 출입 금지 구역, 장애물의 종류 등은 따로 사용자가 수동으로 표기하거나 들어가지 못하도록 조치를 해야 할 필요성이 있다. 따라서

YOLO 네트워크를 이용해 장애물의 종류를 분류한 후, 종류에 따라 안전거리에 대한 가중치를 부여하고, 무인 이동체가 SLAM을 수행하는 시점에서 미리 장애물의 중요성에 따라 맵 위에 안전거리를 다르게 표현함으로써 이점을 가질 수 있다.

[0184] 이를 위해서 YOLO 네트워크에 의해 감지된 장애물의 종류와 위치에 따라 SLAM에 사용하기 위해 라이더 데이터를 오프셋하여 장애물의 크기를 가상으로 키워주어야 하는데, 본 발명에서는 뎁스 카메라와 라이더 측정기를 이용한 알고리즘을 제시한다. 무인 이동체의 전방에는 뎁스 카메라와 라이더가 장착되어 있으며 바운딩 박스의 좌표는 도 11에 나타낸 좌표계를 따르고 (0, 0)부터 (640, 480) 사이의 값을 가진다.

[0185] 도 11은 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 적용된 뎁스 카메라의 시야각을 설명하기 위한 도면으로서, 뎁스 카메라의 시야각(Frame of View)은 60도이며 SLAM을 위한 라이더 데이터의 해상도는 1도로 설정하였다. 따라서 라이더의 전방 좌우각 30도 이내에서 뎁스 및 영상 데이터가 중첩되며 센서 퓨전을 통해 전방의 장애물까지의 거리를 구할 수 있다.

[0186] 상기 도 11의 데이터는 이러한 방법을 통해 구할 수 있으며, d_h는 무인 이동체로부터 장애물까지의 거리, P_x는 시야각에서 장애물의 x축 좌표를 나타낸다. 그리고 P_x를 이용하여 무인 이동체의 중심과 장애물이 이루는 각도 θ_h를 다음의 수학식 27과 같이 구할 수 있다.

[0187] [수학식 27]

[0188] $$\theta_h(t) = P_x(t) \times 30/320$$

[0189] 여기서 θ_h는 화면을 y축을 기준으로 반으로 나누었을 때 화면상 x축의 좌표는 320, 각도는 30도이기 때문에 위와 같이 나타내었으며 x 좌표의 변화량은 다음의 수학식 28과 같이 나타낼 수 있다. 그리고 수학식 27의 관계에서 각도의 변화량은 수학식 29와 같이 구할 수 있다.

[0190] [수학식 28]

[0191] $$\dot{P}_x(t) = P_x(t) - P_x(t-1)$$

[0192] [수학식 29]

[0193] $$\dot{\theta}_h(t) = \dot{P}_x(t) \times 30/320$$

[0194] 무인 이동체로부터 시간 t-1 및 시간 t에서의 장애물의 위치를 알 때 다음의 수학식 30의 삼각형 법칙을 활용하여 화면 내에서 장애물의 단위 시간당 움직인 거리, 즉 속력을 구할 수 있다.

[0195] [수학식 30]

[0196] $$\dot{r}(t) = \sqrt{d_h^2(t) + d_h^2(t-1) - 2d_h(t-1)d_h(t)\cos(\dot{\theta}_h(t))}$$

[0197] 이를 통해 무인 이동체 전면의 장애물의 위치 및 속력의 동적 특성을 확인 할 수 있으며 이에 따라 안전거리의 가중치를 차등적으로 설정한다.

[0198] 장애물의 특성에 따른 가중치를 $r_{safezone}$, i는 안전거리 경계면의 길이라 할 때, 장애물의 위치로부터 설정해야 할 경계면의 거리 $r_{boundary}$는 수학식 31과 같이 표현된다.

[0199] [수학식 31]

[0200] $$r_{boundary}(i) = (vel - i)^2/(1000 + 500(vel - 10)) - r_{safezone}$$
(단, $i \in (1 : 2\,vel)$)

[0201] 오프셋의 길이는 장애물의 속력 $\dot{r}(t)$에 차등적으로 결정되는데, 도 12와 같이 각각 [10, 12, 14, 16]로 설정되는 4단계의 vel 값으로 나눈다.

[0202] 도 12는 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 적용된 속도에 따른 라이더 오프셋의 차이를 설명하기 위한 도면으로서, 라이더 데이터의 오프셋은 속력이 빠를수록 좌우로 길어지며, 가치가 높은 장애물일수록 안전거리가 커지기 때문에 장애물에 대한 경계선이 넓어지게 된다.

[0203] 이를 SLAM에 적용하기 위해서는 장애물이 감지된 위치의 라이더 데이터를 변경하여야 한다. SLAM을 위한 라이더의 해상도는 1도이며, 도 13의 알고리즘을 통해 위치를 추정한다.

[0204] 도 13은 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 적용된 라이더 오프셋 알고리즘을 설명하기 위한 도면이다.

[0205] 도 13에 도시된 바와 같이, 예를 들어 뎁스 카메라의 PointCloud 데이터 중 무인 이동체의 높이와 대응되는 y의 좌표를 440이라 할 때, 모든 해당 y축 좌표의 Point Cloud Data P_x를 구하고, 이를 통해 무인 이동체와 장애물 간의 각도 θ_h 및 속력 $\dot{r}_h$을 계산한다. 그 후 장애물의 위치를 파악하기 위하여 라이더 데이터의 차이(difference) 값을 사용한다. 라이더 데이터는 장애물이 존재하는 구간에서 급변하기 때문에 이를 이용해 영상을 통해 측정된 위치 θ_h에서의 장애물의 존재 여부를 판단하고 보조 역할을 수행한다.

[0206] 영상에 의한 지점과 라이더에 의해 발견된 장애물의 위치가 유사할 경우 해당 위치에 장애물이 존재한다고 판정할 수 있으며, 이 지점을 기준으로 $r_{boundary}$ 만큼 오프셋을 수행한다. 그리고 SLAM 패키지에 적용될 수 있도록 하여 장애물에 대한 경계면이 SLAM의 화면 및 맵 상에도 적용될 수 있도록 한다.

[0207] 이와 같이 $r_{boundary}$를 계산하고 라이더 데이터의 오프셋을 수행한 결과는 도 14와 같다.

[0208] 도 14는 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의한 영상 인식과 라이더 데이터의 오프셋을 설명하기 위한 도면으로서, 무인 이동체가 영상을 통해 사람을 인지하였을 때 라이더 데이터가 실제 장애물의 위치 앞에 안전거리에 의한 경계선을 생성하고 있음을 확인할 수 있다.

[0209] 또한 도 14를 통해 사람이 서 있거나 이동하는 환경에서의 수행 결과는 도 15 및 도 16과 같다.

[0210] 도 15는 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의해 서있는 사람과 물체의 결과를 나타낸 도면이며, 도 16은 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의해 움직이는 사람과 물체의 결과를 나타낸 도면이다.

[0211] 도 15 및 도 16에서 녹색 표시는 현재 SLAM 프로그램에서 인식되고 있는 라이더 데이터를 나타내며, 검정색 표시는 SLAM에 의하여 장애물로 인식된 곳을 나타낸다. 이를 통해 실제 장애물이 존재하는 위치의 전면에 안전거리를 나타내는 경계선이 맵에 적용되고 있음을 확인할 수 있다.

[0212] 또한 라이더 데이터를 비교하였을 때 사람이 정지해 있는 도 15가 사람이 걷고 있는 환경인 도 16의 경계선보다 짧은 것을 확인할 수 있으며, 이에 따라 라이더 데이터의 오프셋이 적용되고 있음을 확인할 수 있다.

[0213] 또한 YOLO를 통해 인식된 2가지 장애물 이동중인 사람(Walking Human) 및 모니터(Monitor)를 동시에 인식하고, 이에 대한 안전거리가 동시에 적용되고 있음을 확인할 수 있다.

[0214] 도 17은 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 시스템에 의해 무인 이동체가 움직이고 있는 공간에 대해 자율적인 SLAM을 수행한 결과를 나타낸 도면으로서, 상기 맵리스 내비게이션 및 상기 오프셋 알고리즘을 동시에 수행하여 무인 이동체가 움직이고 있는 공간에 대해 자율적인 SLAM을 수행한 결과이다.

[0215] 다음에는, 이와 같이 구성된 본 발명에 따른 장애물의 특성을 고려한 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법의 일 실시예를 도 18을 참조하여 상세하게 설명한다. 이때 본 발명의 방법에 따른 각 단계는 사용 환경이나 당업자에 의해 순서가 변경될 수 있다.

[0216] 도 18은 본 발명의 일 실시예에 따른 DQN 및 SLAM 기반의 맵리스 내비게이션 처리 방법의 동작과정을 상세하게 나타낸 순서도이다.

[0217] 우선, 가상 환경 구축 모듈(110)을 통해 무인 이동체와, 사람, 물체 또는 이들의 조합을 포함한 장애물을 가상 환경으로 구현한다(S100). 즉 무인 이동체를 실제 환경에서 학습할 경우 안전성 및 효율이 떨어지기 때문에, 가상 시뮬레이션을 구성하는 것이다.

[0218] 이후, 상기 학습 모듈(130)은 사람, 물체 또는 이들의 조합을 포함한 장애물이 위치한 소정의 공간에서, 현재 상태에서 다음 상태로의 무인 이동체에 대한 액션을 출력하기 위하여, 상기 무인 이동체와 타겟 지점 사이의 각도 및 거리 데이터, 상기 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 학습하는 학습 단계를 수행한다.

[0219] 상기 학습 단계를 상세하게 설명하면, 상기 학습 모듈(130)은 에피소드를 최초에 수행할 때 상기 타겟 지점의 반경을 원래 크기의 정수 배(예: 2배)로 확장하여 설정하는 단계를 수행한다(S200). 이때 본 발명에서는 상기 무인 이동체가 타겟 지점에 도착하거나, 벽, 장애물 및 사람과 충돌이 발생하거나, 또는 상기 타겟 지점에 기 설정된 시간 이내에 도착하지 못했을 경우를 하나의 에피소드로 설정한다.

[0220] 상기 S200 단계를 통해 타겟 지점의 크기를 확장한 이후, 상기 학습 모듈(130)은 에피소드에 따라 무인 이동체와 타겟 지점 사이의 각도 및 거리, 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 입력으로 하여, 다음 상태의 액션을 출력하기 위한 학습을 진행한다(S300).

[0221] 이때 상기 학습 모듈(130)은 각 에피소드에 따른 학습을 수행할 때, 각 스텝 타임마다 상기 무인 이동체에 구비된 라이더 측정기(122)를 통해 주변을 스캔한 데이터들을 모으고 쌓아서 만든 스택(즉 라이더에 의한 거리 데이터)을 참조하여 액션을 선택한다.

[0222] 상기 S200 및 S300 단계를 통해 특정 에피소드의 학습을 수행한 이후, 라이더 데이터 오프셋 모듈(140)은 상기 학습을 수행하는 과정에서, 상기 장애물이 감지되면 상기 장애물의 종류, 속도 또는 이들의 조합을 포함한 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋하는 라이더 데이터 오프셋 단계를 수행한다(S400).

[0223] 상기 S400의 라이더 데이터 오프셋 단계를 보다 상세하게 설명하면, 상기 라이더 데이터 오프셋 모듈(140)은 상기 무인 이동체에서 촬영한 상기 장애물의 뎁스 데이터 및 영상인식에 따른 바운딩 박스 데이터를 토대로 상기 장애물을 감지하여 상기 장애물의 종류를 확인한다. 그리고 상기 장애물이 감지되면 상기 무인 이동체에서 측정한 현재 상태와 이전 상태간의 라이더 데이터의 차이를 계산하고, 상기 계산한 라이더 데이터의 차이를 통해서 상기 장애물의 위치를 확인한다. 그리고 상기 확인한 장애물의 종류에 따라 상기 확인한 장애물의 위치로부터 설정해야 할 안전거리를 차등적으로 부여하고, 상기 장애물과, 상기 차등적으로 부여한 안전거리에 대한 라이더 데이터의 오프셋을 수행한다.

[0224] 이처럼, 상기 S400 단계를 통해 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋한 이후, 맵 생성 모듈(150)은 상기 오프셋한 라이더 데이터를 입력받아 상기 장애물에 대한 경계면을 맵 상에 업데이트하여 표시하는 맵 생성 단계를 수행한다(S500).

[0225] 이후, 상기 학습 모듈(130)은 상기 에피소드의 수행에 따라 상기 무인 이동체가 상기 타겟 지점에 도착하여 에피소드가 종료되는지를 판단한다(S600).

[0226] 상기 S600 단계의 판단결과, 상기 무인 이동체가 상기 타겟 지점에 도착하여 에피소드가 종료되면, 상기 학습 모듈(130)은 상기 타겟 지점의 반경을 기 설정된 범위(예: 1%)로 줄인 다음, 다음의 에피소드를 수행한다(S700).

[0227] 그러나 상기 S600 단계의 판단결과, 상기 무인 이동체가 상기 타겟 지점에 도착하지 못하고, 상기 무인 이동체가 장애물과의 충돌이나, 기 설정 시간 이내에 타겟 지점에 도착하지 못하여 에피소드가 종료되면, 상기 학습 모듈(130)은 현재의 에피소드를 종료하고 다음의 에피소드를 진행하여(S800), 상기 S300 단계 이후를 반복하여 수행한다.

[0228] 상기 S700 단계를 통해 상기 타겟 지점의 반경을 기 설정된 범위(예: 1%)로 줄이고 다음의 에피소드를 진행하는 경우, 상기 학습 모듈(130)은 타겟 지점의 크기가 원래의 크기(즉, 모든 에피소드의 종료 조건)와 동일하거나 혹은 가장 유사한 크기인지를 판단하고(S900), 상기 판단한 결과 타겟 지점의 크기가 원래의 크기보다 크면, 상기 타겟 지점의 반경이 원래 크기가 될 때까지 각 에피소드별로 상기 S300 단계 이후를 반복한다.

[0229] 그러나 상기 S900 단계의 판단결과 타겟 지점의 크기가 원래의 크기가 되면, 상기 학습 모듈(130)은 각 에피소드에 따른 학습을 종료하고, 각 장애물별 가중치를 차등적으로 부여한 SLAM을 최종적으로 완성한다(S1000).

[0230] 이처럼, 본 발명은 무인 이동체를 위한 DQN 기반의 맵리스 내비게이션을 구현하여 타겟의 크기를 최초에 2배로 조정한 후 점차 줄여 학습 시간을 감소시키고, 장애물의 종류, 속도 등의 특성을 파악하여 각 장애물의 특성에 따라 안전거리의 가중치를 차등적으로 SLAM에 적용하여 맵 상에 표시함으로써, 상기 무인 이동체가 SLAM을 자율

적으로 수행할 수 있으며, 사람과 여러 장애물이 존재하는 스마트 팩토리 환경에서 사람이나 장애물과의 충돌 없이 최단 경로로 타겟 지점에 도달할 수 있다.

[0231] 이상에서와 같이 본 발명은 도면에 도시된 실시예를 참고로 하여 설명되었으나, 이는 예시적인 것에 불과하며, 당해 기술이 속하는 분야에서 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 따라서 본 발명의 기술적 보호범위는 아래의 특허청구범위에 의해서 판단되어야 할 것이다.

부호의 설명

[0232] 100 : DQN 및 SLAM 기반 맵리스 내비게이션 시스템

110 : 가상 환경 구축 모듈 120 : 센서

121 : 뎁스 카메라 122 : 라이더 측정기

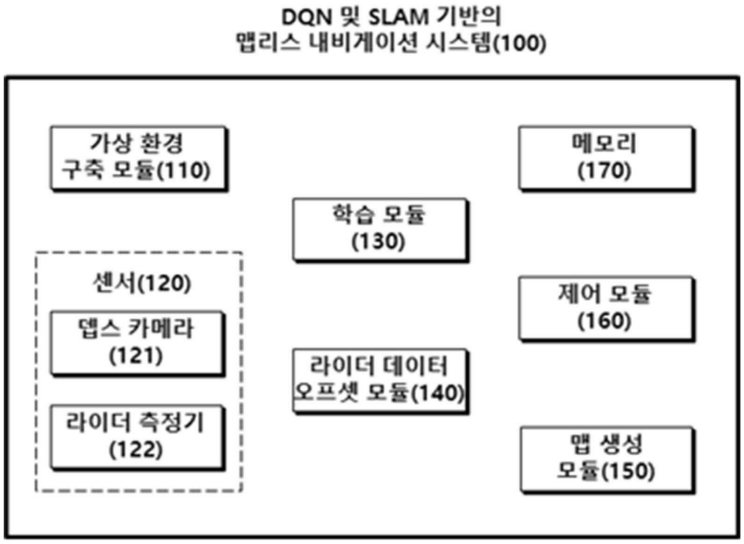
130 : 학습 모듈 140 : 라이더 데이터 오프셋 모듈

150 : 맵 생성 모듈 160 : 제어 모듈

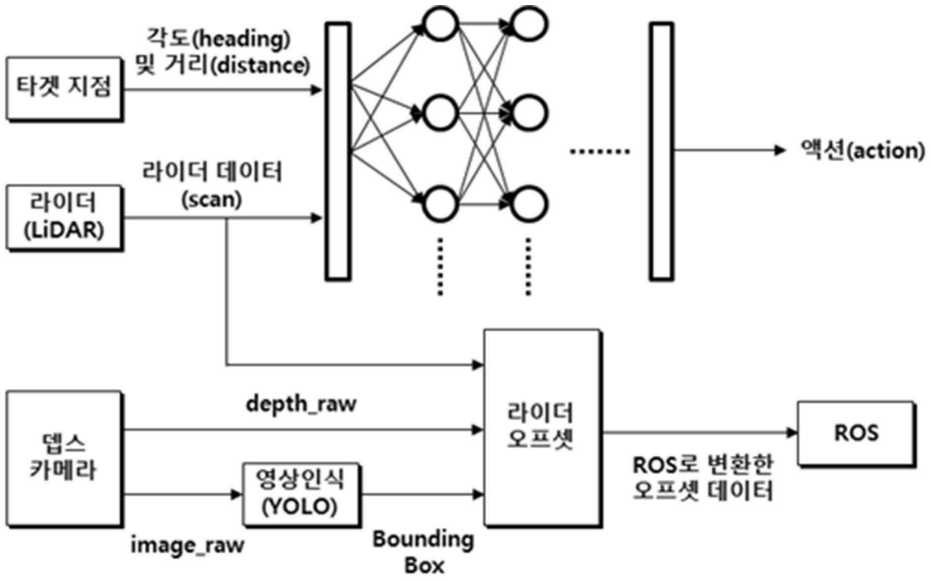
170 : 메모리

도면

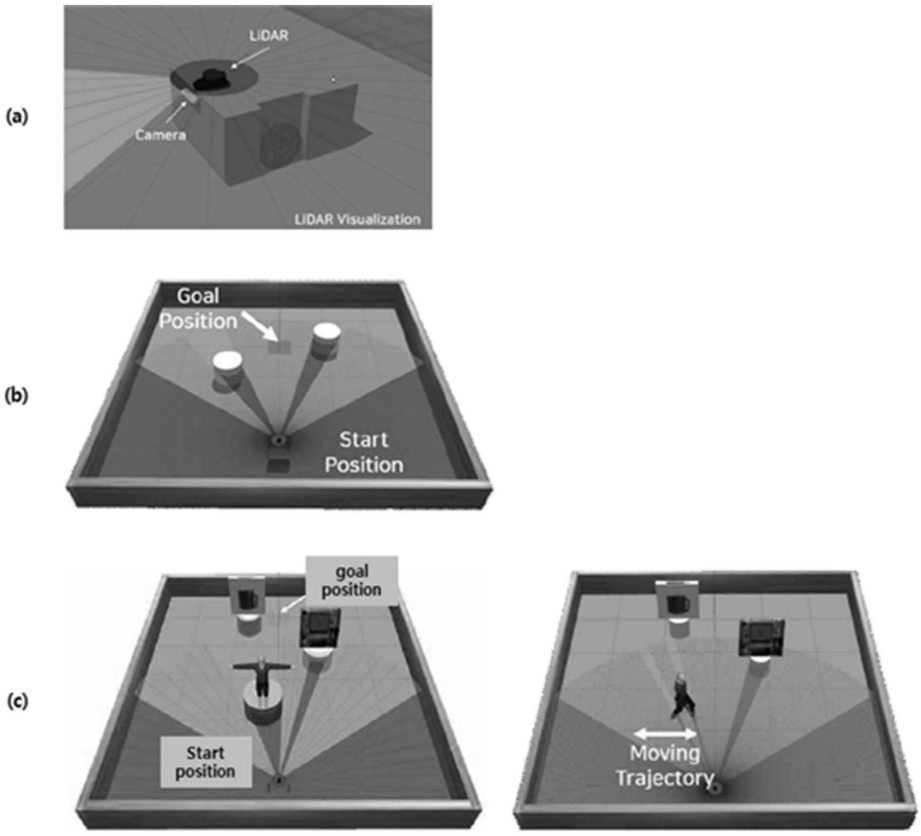
도면1



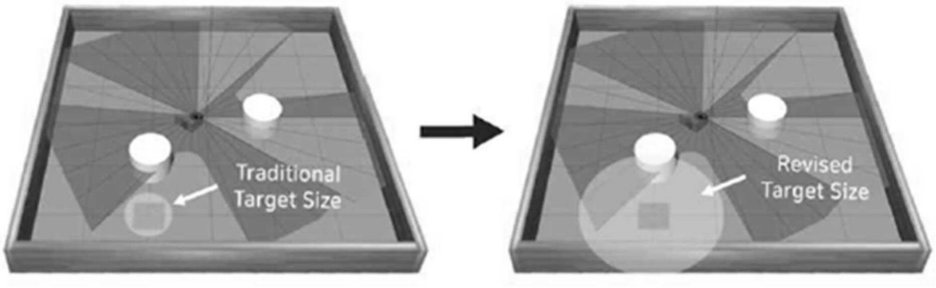
도면2


타겟 지점
각도(heading)
및 거리(distance)
라이더
(LiDAR)
라이더 데이터
(scan)
액션(action)
뎁스
카메라
depth_raw
영상인식
(YOLO)
image_raw
Bounding
Box
라이더
오프셋
ROS로 변환한
오프셋 데이터
ROS

도면3


(a)
LiDAR
Camera
LiDAR Visualization
(b)
Goal
Position
Start
Position
(c)
goal
position
Start
position
Moving
Trajectory

도면4



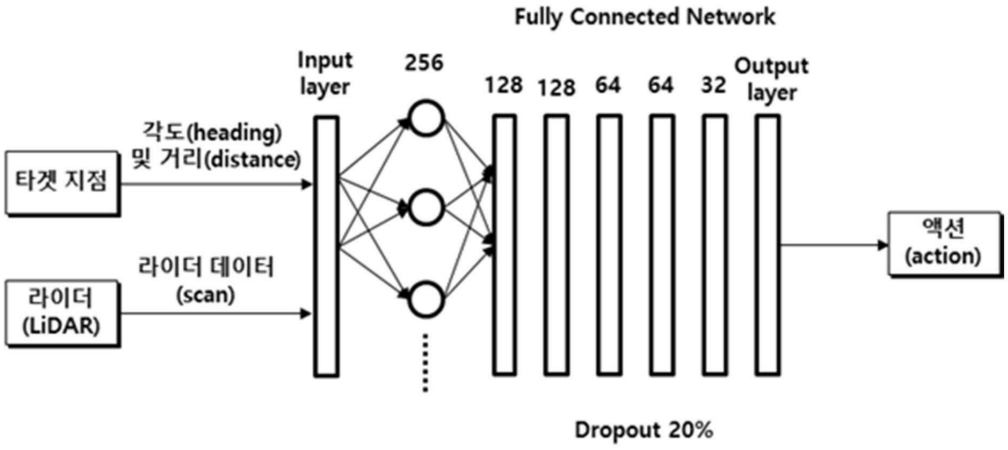
원래의 타겟 크기

타겟의 크기를 최초에 2배로 조정
-> 학습 진행에 따라 소정의 범위로
줄여 원래의 타겟 크기까지 학습 수행

도면5

리워드 타입	값(value)
주행 리워드 (driving reward)	$Reward = Heading\ Reward + Distance\ Reward$
타겟 도착 리워드 (goal reward)	$+100(200-t)/200$ (단, $t<200$)
충돌 리워드 (collision reward)	-150

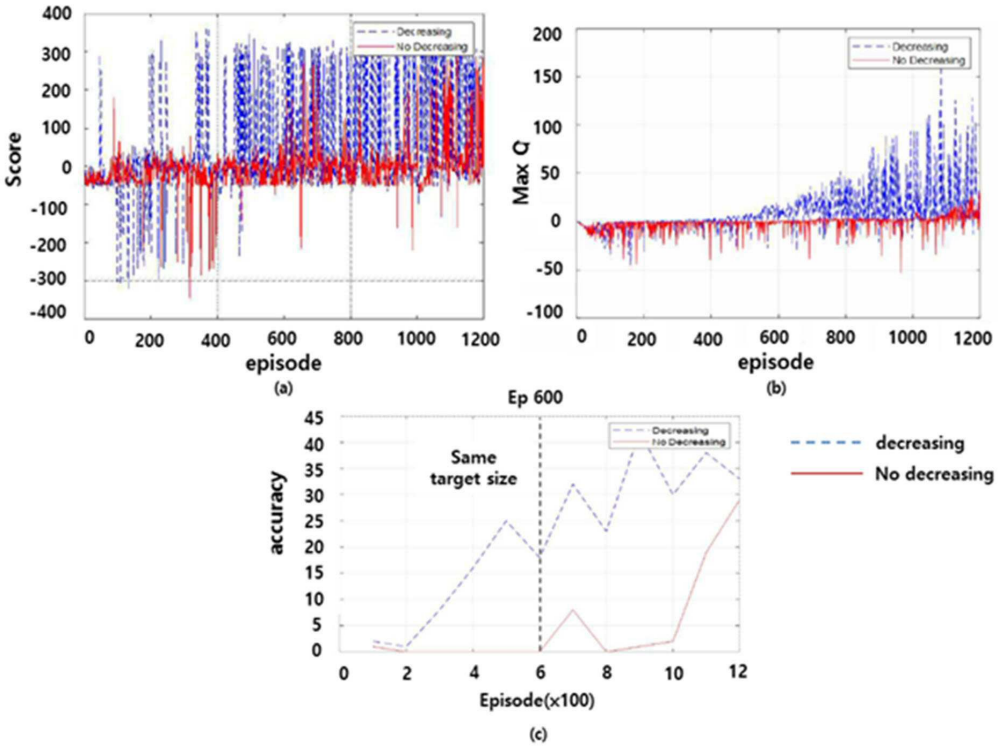
도면6



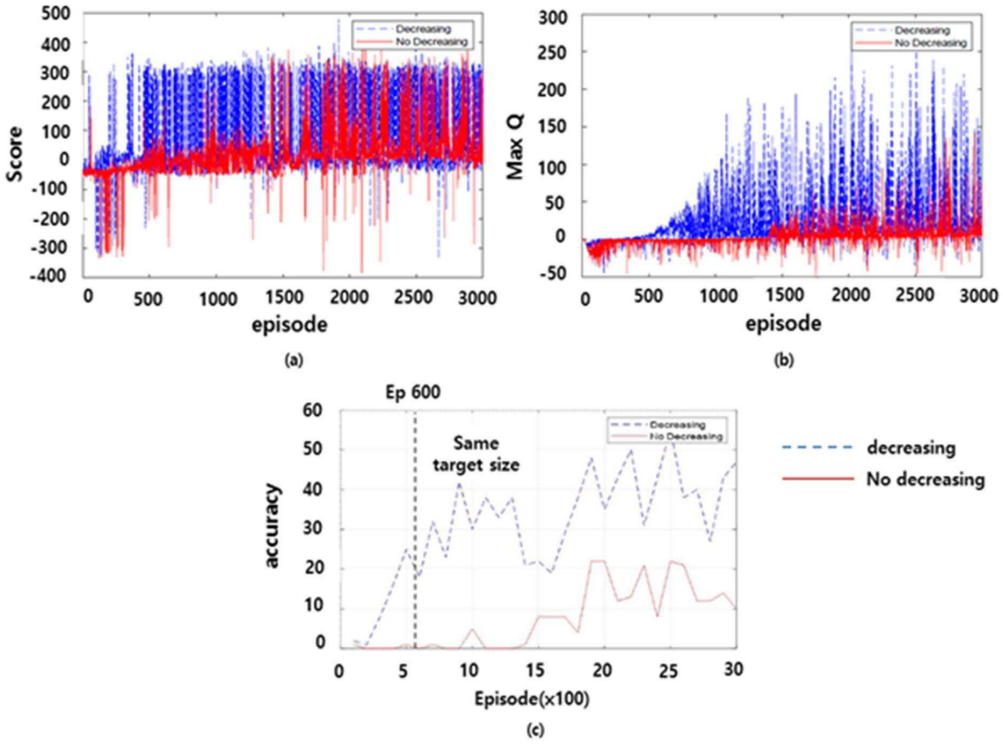
도면7

액션(action)	선속도(linear velocity)	각속도(angular velocity)
0	0.05 m/s	-1.5 rad/s
1	0.2 m/s	-0.75 rad/s
2	0.25 m/s	0 rad/s
3	0.2 m/s	0.75 rad/s
4	0.05 m/s	1.5 rad/s

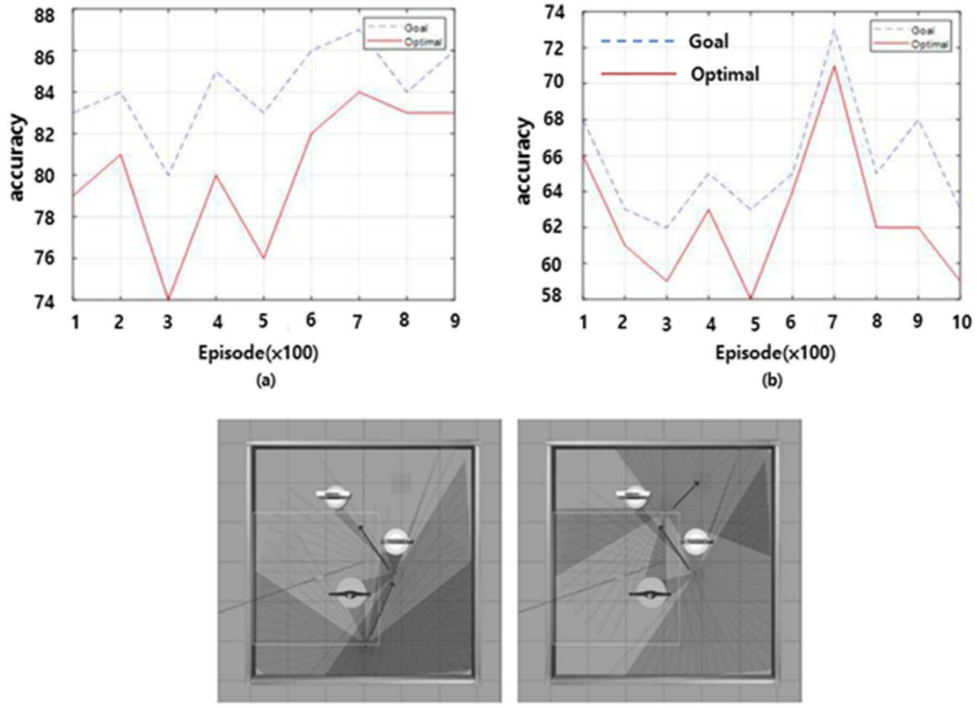
도면8



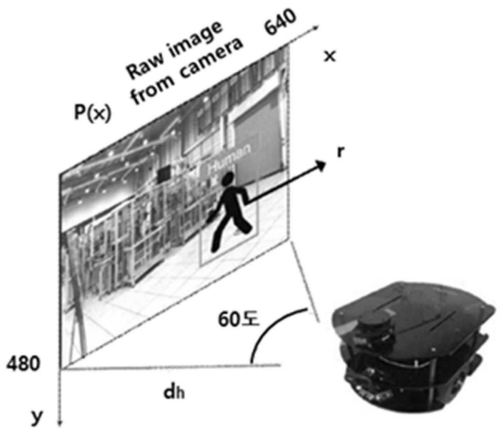
도면9


Score
episode
(a)
Max Q
episode
(b)
Ep 600
Same
target size
accuracy
Episode(x100)
(c)
decreasing
No decreasing

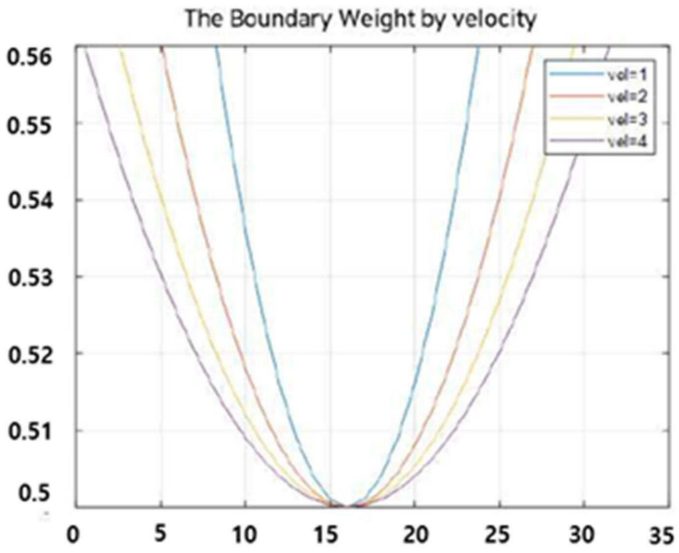
도면10


accuracy
Episode(x100)
(a)
Goal
Optimal
accuracy
Episode(x100)
(b)
(c) Optimal path driving

도면11



도면12



도면13

LiDAR Offset Algorithm

1 get Obstacles Position from YOLO and Point Cloud Data

$r_{boundary}(t) \leftarrow P_x(t),\ \theta_h(t),\ \dot{r_h}(t)$

2 calculate the Difference of LiDAR Data

$L_{diff} \leftarrow L[\theta_h - 10 : \theta_h + 9] - L[\theta_h - 9 : \theta_h + 10]$

3 check the obstacle position from L_{diff}

if $L_{diff}(\theta_h) > \min(L_{diff})$

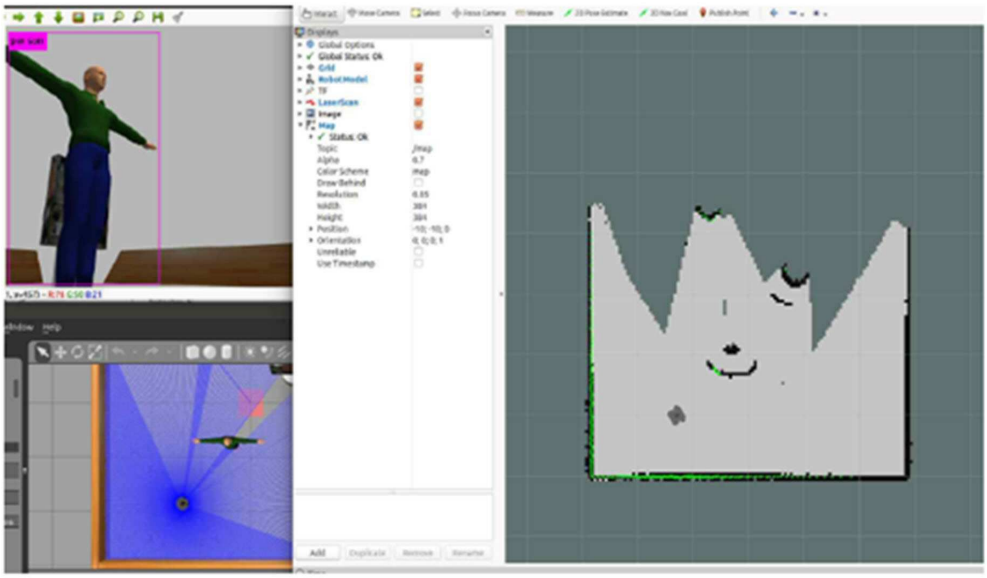
pass

4 for i in (1, 2vel)

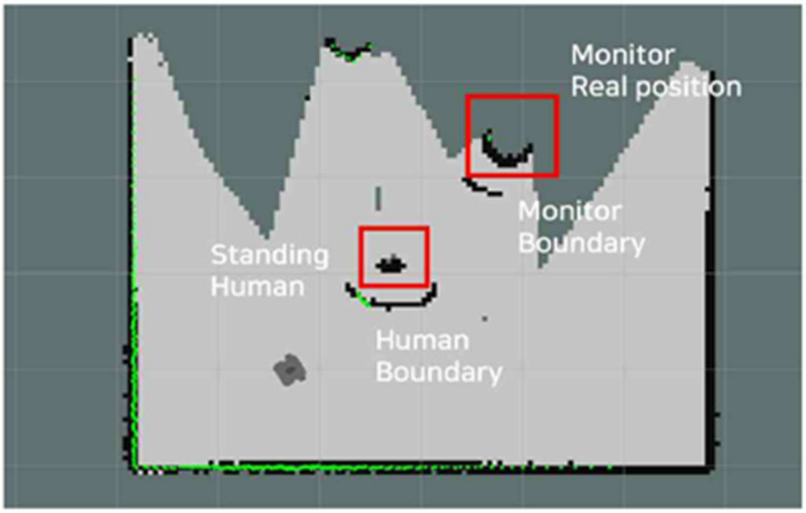
$L(\theta_h + i - vel) \leftarrow L(\theta_h + i - vel) - r_{boundary}(i)$

5 Publish Scan Data to SLAM program

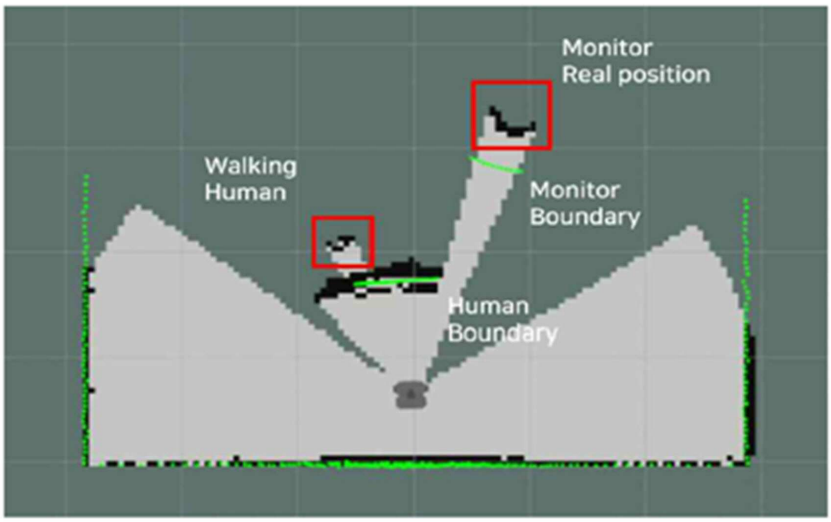
도면14



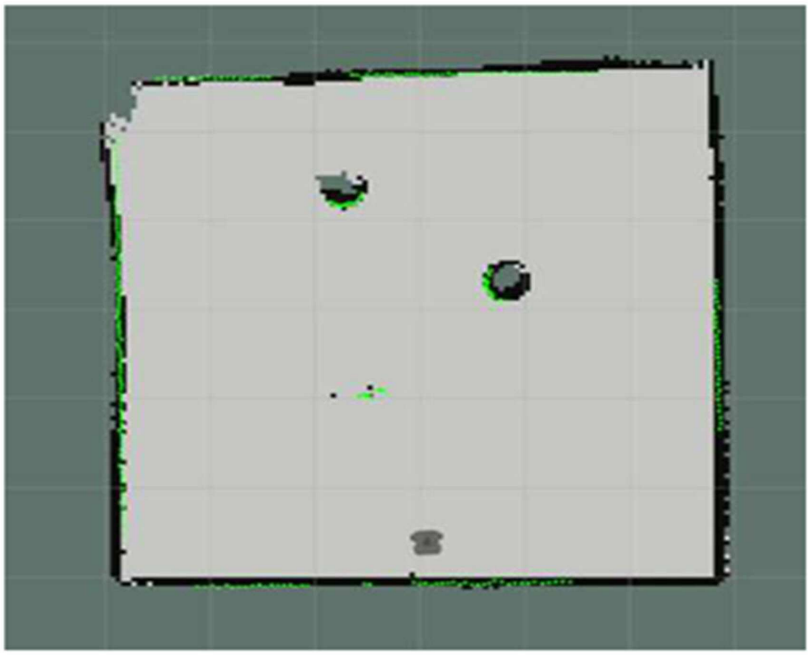
도면15


Monitor
Real position
Monitor
Boundary
Standing
Human
Human
Boundary

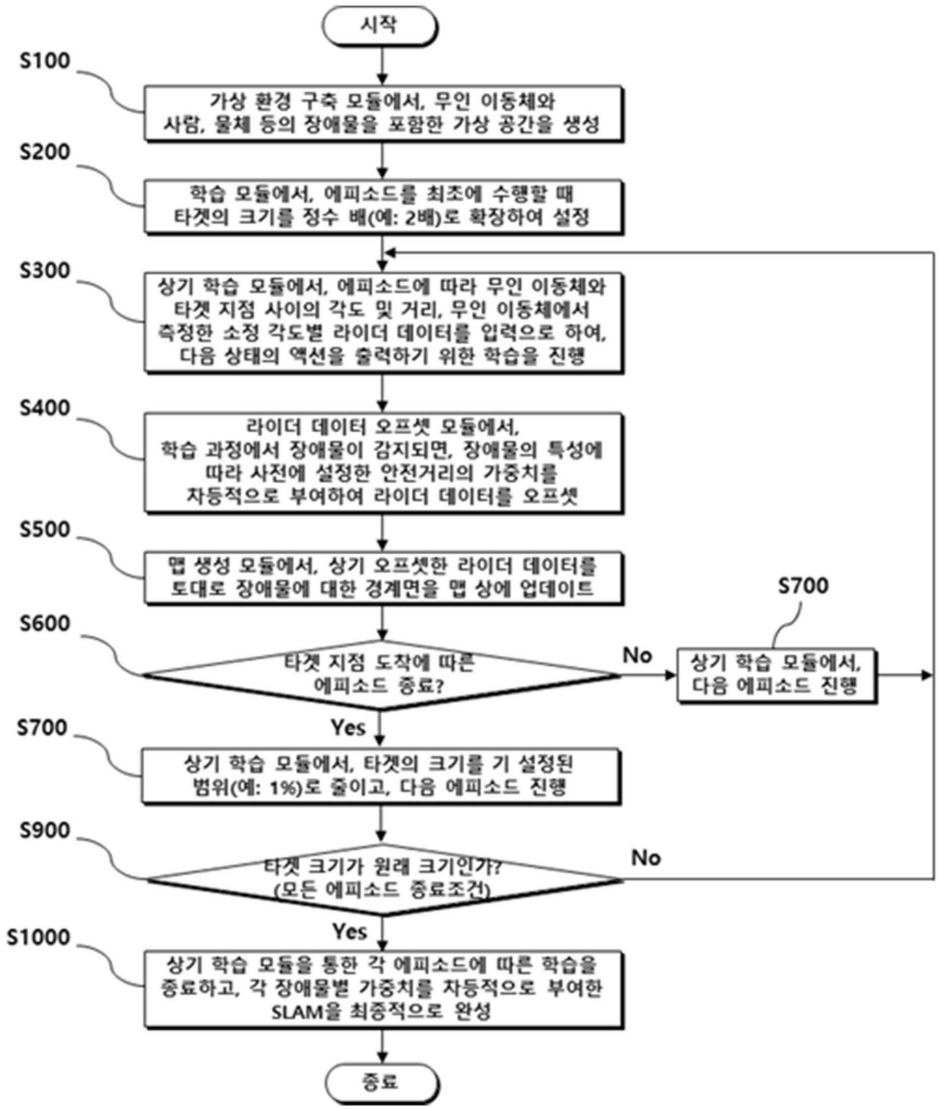
도면16


Monitor
Real position
Walking
Human
Monitor
Boundary
Human
Boundary

도면17



도면18


시작
S100
가상 환경 구축 모듈에서, 무인 이동체와 사람, 물체 등의 장애물을 포함한 가상 공간을 생성
S200
학습 모듈에서, 에피소드를 최초에 수행할 때 타겟의 크기를 정수 배(예: 2배)로 확장하여 설정
S300
상기 학습 모듈에서, 에피소드에 따라 무인 이동체와 타겟 지점 사이의 각도 및 거리, 무인 이동체에서 측정한 소정 각도별 라이더 데이터를 입력으로 하여, 다음 상태의 액션을 출력하기 위한 학습을 진행
S400
라이더 데이터 오프셋 모듈에서, 학습 과정에서 장애물이 감지되면, 장애물의 특성에 따라 사전에 설정한 안전거리의 가중치를 차등적으로 부여하여 라이더 데이터를 오프셋
S500
맵 생성 모듈에서, 상기 오프셋한 라이더 데이터를 토대로 장애물에 대한 경계면을 맵 상에 업데이트
S600
타겟 지점 도착에 따른 에피소드 종료?
No
S700
상기 학습 모듈에서, 다음 에피소드 진행
Yes
S700
상기 학습 모듈에서, 타겟의 크기를 기 설정된 범위(예: 1%)로 줄이고, 다음 에피소드 진행
S900
타겟 크기가 원래 크기인가? (모든 에피소드 종료조건)
No
Yes
S1000
상기 학습 모듈을 통한 각 에피소드에 따른 학습을 종료하고, 각 장애물별 가중치를 차등적으로 부여한 SLAM을 최종적으로 완성
종료