**로봇제어프로그래밍 텀프로젝트 보고서**

자동차공학과 23512093 곽제우

**1. 프로젝트 목표**

TurtleBot3 World 및 House 환경에서 장애물을 피해서 목적지에 도달할 수 있도록 경로 계획 또는 제어 모듈을 구현한다.  
전체의 지도가 주어지지 않은 상황에서 사용자가 임의로 지정한 목적지까지 이동할 수 있고, 별도의 패키지 빌드 없이 목적지를 변경할 수 있다.  
장애물의 위치는 라이다 센서만을 이용하여 감지한다.

**2. 이론적 배경**

**A\* algorithm**A\* 알고리즘은 주어진 출발점에서부터 목표점까지 가는 최단 경로를 찾아내는 그래프 탐색 알고리즘 중 하나이다. A\* 알고리즘의 특징은 각 꼭짓점 x에 대해 그 꼭짓점을 통과하는 최상의 경로를 추정하는 순위값인 휴리스틱 추정값 h(x)을 매기는 방법을 이용한다는 것이다. 이 알고리즘은 휴리스틱 추정값의 순서로 꼭짓점을 방문한다. 그러므로 A\* 알고리즘을 너비 우선 탐색의 한 예로 분류할 수 있다.  
A\* 알고리즘은 출발 꼭짓점으로부터 목표 꼭짓점까지의 경로를 탐색하기 위한 것이다. 이를 위해서는 각각의 꼭짓점에 대한 평가함수를 정의해야 한다. 이를 위한 비용 함수 f(n)은 다음과 같다.



*  : 출발 꼭짓점으로부터 꼭짓점 n까지의 경로 가중치.
*  : 꼭짓점 n으로부터 목표점까지의 추정 경로 가중치.

작동 방식은 다음과 같다.

* 시작 노드부터 탐색을 시작한다.
* 현재 노드의 인접 노드들을 열린 목록(open list)에 추가한다.
* f(n)이 가장 작은 노드를 선택하여 탐색을 계속한다.
* 이미 방문한 노드는 닫힌 목록(closed list)에 추가한다.
* 목표 노드에 도달할 때까지 이 과정을 반복한다.

특징: A\* 알고리즘은 휴리스틱 함수가 적절하다면 항상 최적의 경로를 찾는다. 따라서 게임, 로봇 경로 계획, 내비게이션 시스템 등 다양한 분야에서 활용되고 있다.

**Cartographer**Cartographer는 주변환경을 인지하여 grid map을 nav\_msgs/OccupancyGrid 메시지 타입 사용하여 퍼블리시한다.  
각 셀은 점유 확률값(0-100)을 갖는다. 0-59는 비어있는 공간, 60-100는 장애물이 있는 공간, -1은 미지의 공간을 나타낸다. resolution(해상도), width(폭), height(높이), origin(원점) 정보 포함하고, 메모리 효율적이고 계산이 빠르다는 특징을 갖는다.  
**Octomap**Octomap은 3D 공간을 8진 트리 구조로 표현한다. 각 노드가 8개의 하위 노드를 가질 수 있고, 공간을 재귀적으로 분할하여 표현한다. 3D 환경 매핑에 효율적이고, 확률적 업데이트 지원하며 메모리 효율적 (필요한 부분만 세밀하게 표현 가능)이고, 동적 환경 변화 대응 가능하다는 특징을 갖는다.

**DDPG(Deep Deterministic Policy gradient)**

DDPG 알고리즘은 확정적 정책 그래디언트를 사용한다. 확정적 정책 그래디언트란 주어진 상태변수 값에서 행동변수의 값을 확정적으로 계산할 수 있다는 뜻이다. 즉,  이다. 따라서 심층 신경망을 정책 신경망으로 이용하면 정책 신경망의 출력 층 뉴런 개수를 행동변수의 개수와 같도록 설정하면 된다.  
참고로 PPO나 SAC과 같이 확률적 정책 그래디언트의 경우 확률밀도함수가 를 사용하는 것과 대조적이다.  
DDPG는 Actor Critic의 두개의 네트워크를 학습시킨다. Actor은 agent(로봇)의 행동을 출력하는 네트워크이고, Critic은 Actor의 출력과 Q-value를 비교해서 더 나은 방향으로 학습을 한다.

DDPG Pseudo code

1. 크리틱과 액터 신경망의 파라미터 와 를 초기화한다.
2. 크리틱과 액터의 타깃 신경망의 파라미터 와 에 와 를 복사한다.
3. 리플레이 버퍼 D를 초기화한다.
4. Repeat {

[1] 행동 탐험용 랜덤 프로세스 을 초기화한다.

[2] 초기 상태변수 를 측정한다.

[3] Repeat {

(1) 현재 정책 로 행동을 계산한다.

(2) 를 실행해 보상 와 다음 상태 변수 을 측정한다.

(3)기본 샘플 단위 을 리플레이 버퍼 D에 저장한다.

(4) D에서 N개의 기본 샘플 단위  미니배치를 무작위로 축출한다.

(5) 시간차 타깃 을 계산한다.

(6) 다음 손실함수를 이용해 크리틱 신경망을 업데이트한다.



(7) 다음 그래디언트 식으로 액터 신경망을 업데이트한다.



(8) 타깃 크리틱과 타깃 액터 신경망을 업데이트 한다.



}N 시간 스텝 반복

}

위 Pseudo 코드처럼 먼저 액터 신경망으로부터 행동 를 계산한다. 그다음 크리틱 신경망으로부터 상태변수와 행동에 대한 행동 가치함수 를 계산한다. 그리고 손실함수 의 그래디언트를 계산한다. 그리고 텐서플로우 Adam Optimizer를 이용해 액터 신경망을 학습한다.

**3. Navigation Architecture(네비게이션 아키텍처)**

Navigation Architecture는4가지 단계를 거쳐서 이뤄진다.

Map Processing 단계에서는 Grid Map이 생성된다. 이 단계에서는 Cartographer package를 사용해서 SLAM을 수행한다. Cartographer에서 출력한 0~100사이의 Occupancy 값을 통해서 Unknown(-1)을 포함한 60미만의 값을 사용하며, 점유도 60이상의 grid주변 30cm는 unsafe로 처리해 A\* 경로에서 제외된다.

Localization 단계에서는 우선 Lidar가 주변 환경을 스캔하고, 스캔데이터와 기존 맵을 매칭시킨 후 매칭 결과로 로봇의 현재 위치를 계산한다. 이 정보를 TF 변환으로 브로드캐스트한다. map에서 base\_scan까지 변환 행렬을 계산해서 맵 상에서 로봇의 위치(Lidar의 위치)를 추정한다. map->base\_scan으로 매칭한 이유는 SLAM이 Lidar 데이터를 기반으로 동작하기 때문이다.

Path Planning 단계에서는 A\*를 사용한다. 2D gird occupancy map을 사용하고 로봇이 holonomic 특성을 갖고 있기 때문에 대각선을 포함하여 8개의 방향으로 경로를 계산하는 A\*를 사용해도 충분히 navigation 기능을 수행할 수 있다고 판단했고 A\*는 최적해를 보장하기 때문에 A\*를 이용해 경로계획을 수행했다. A\* 세부 알고리즘은 다음과 같다.   
1) 경로 계획 초기화  
TF를 통해서 start\_point(로봇의 초기 위치)를 받고 rviz에서 2D\_goal\_pose로 map에 goal\_point를 정하면 이 두 점을 기반으로 A\* 알고리즘 수행한다.

1. A 알고리즘 구조 설정

우선순위 큐를 사용하여 open set 구현한다. f\_cost(총 예상 비용)가 가장 작은 노드부터 탐색하도록 설정했고, 시작 노드의 초기 비용 설정한다. (h\_cost는 휴리스틱 함수로 계산) 그리고 closed set에 방문한 노드를 표시해서 이미 방문한 노드 추적할 수 있도록 한다.

1. 메인 탐색 루프

메인 탐색 루프는open set이 비어 있을 때까지 반복한다. 우선 f\_cost가 가장 작은 노드를 현재 노드로 선택한다. 그 다음 선택된 노드의 유효성 검사 (장애물 충돌 여부)를 실행한다. 그 다음 closed set에서 중복 검사를 실행하고 마지막으로 목표 지점까지의 거리가 15cm 이내면 경로 생성 완료한다.

1. 이웃 노드 탐색

이웃 노드 탐색은 현재 노드에서 8방향(상하좌우 및 대각선)으로 이웃 노드 생성한다. 이웃 노드마다 15cm 간격으로 새로운 위치 계산, 장애물 충돌 검사로 유효성 확인, g\_cost(시작점부터의 실제 거리), h\_cost(목표점까지의 예상 거리, Manhattan distance 사용), f\_cost(g\_cost + h\_cost) 계산 그리고 유효한 노드는 open set에 추가하는 과정을 실행한다.

1. 안전성 확보

안정성을 확보하기 위해서 is\_point\_valid 함수로 모든 경로점이 경계(벽)과 30cm 이내 일 때 A\*의 경로로 사용하지 않는다. (open set에 추가하지 않는다)

이렇게 생성된 경로는 15cm 간격의 경로점들로 구성되며, 이는 로봇이 안전하게 따라갈 수 있는 최적 경로를 생성한다. 각 경로점은 로봇의 위치와 장애물을 고려하여 선택되므로, 안전하고 효율적인 네비게이션을 할 수 있다.

Motion Control단계에서는Look-ahead point 위치로 수렴하도록 제어를 수행한다. 우선 현재 위치에서 가장 가까운 경로점 찾고, 두번째 점을 목표점으로 설정한다. 로봇의 현재 위치와Look-ahead point 사이의 거리와 각을 제어에 사용한다.  
Motion Control은 로봇이 최종 목표 지점에 도달했는지 아닌지에 따라서 두가지 모드로 구분해서 수행한다.

먼저 경로 추종 모드에서는 Look-ahead point 따라 최종 목표점까지 이동한다. 이때 각도 차이가 크다면(>0.5 rad)거의 제자리에서 회전을 수행한다. 반대로 각도 차이가 작다면(<=0.5 rad) 전진하면서 방향 Look-ahead point으로 방향을 조절한다.

최종 자세 조정 모드는 최종 목표점에 도달했을 때 수행되며, 목표점에서 원하는 방향으로 회전한다. 최종 자세 조정 모드에서는 PID 제어를 사용해서 목표 방향에 수렴한다. 여기서 타깃 각도와 현재 로봇의 yaw 값의 차이를 angular error로 하여 이것을 0으로 수렴하도록 제어한다. PID 제어는 다음의 식과 gain을 통해서 수행한다.



안전 메커니즘으로 경로를 찾지 못한 경우(path node가 비어 있을 경우) 로봇은 3초 주기로 시계, 반시계 반향으로 제자리 회전을 반복하고 그 동안 Map을 업데이트한다.

텍스트, 스크린샷, 도표, 직사각형이(가) 표시된 사진

자동 생성된 설명

Figure 1 Navigation Artchitecture

**4. Autonomous Exploration with DDPG**

**4.1 DDPG 네트워크**

DDPG는 Actor-Critic 구조를 기반으로 하며, 각각의 네트워크는 다음과 같은 역할을 수행한다: 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Figure 2 DDPG Actor Critic Network

Actor 네트워크는 주어진 상태에서 최적의 행동을 결정하는 정책을 학습한다. 4개의 층으로 구성되어 있으며, 각 층은 점진적으로 크기가 줄어드는 구조를 가진다. 이는 고차원의 상태 정보를 낮은 차원의 행동 명령으로 효과적으로 매핑하기 위함이다.  
Critic 네트워크는 상태-행동 쌍의 가치를 평가한다. 상태와 행동 정보를 별도로 처리한 후 결합하는 구조를 가지며, 이를 통해 각 요소의 특성을 독립적으로 추출할 수 있다.

**4.2 State 설계**

상태 공간은 로봇의 효과적인 탐색을 위해 다음과 같은 요소들로 구성된다:

1. 로봇 위치 정보

전역 좌표계에서의 2D 위치 (x, y), 로봇의 방향각 (yaw), 현재 이동 속도 (선속도, 각속도)

1. 환경 정보

로봇 주변 5m x 5m 영역의 로컬 맵 데이터, 전체 탐험 된 영역의 비율, 전방 영역 탐색 (알려진 영역과 미지 영역의 경계) 정보를 사용한다.

이와 같이 상태 정보를 설계하여 로봇이 효율적인 탐색 경로를 계획하고 안전한 주행을 할 수 있도록 했다.

**3.3 Action 설계**

행동 공간은 로봇의 2차원 움직임을 제어하기 위한 두 가지 연속적인 값으로 구성된다:

1. 선속도: 전진/후진 속도를 제어

범위를 [-0.05, 0.05] (m/s)으로 하여 작은 범위로 제한하여 안전성 확보한다.

1. 각속도: 회전 속도를 제어

범위를 [-0.1, 0.1] (rad/s)으로 하여 급격한 방향 전환을 방지한다.

**4.4 Reward 설계**

보상 함수는 다음 세 가지 주요 요소를 고려하여 설계되었다:

1. 탐험 보상

새로 발견된 영역에 대한 격자 수를 계산해서 양의 보상한다. 즉, 발견된 영역의 크기에 비례하는 보상 값 제공한다.

1. 전방 영역 탐색 보상

미지 영역과 접한 경계를 발견할 때 추가 보상을 통해 효율적인 탐색 경로 생성 유도한다.

3.안전 보상

장애물과의 거리에 기반한 페널티를 준다. 그리고 30cm 이내 장애물이 접근 시 거리에 반비례하는 음의 보상하고, 충돌 회피 행동 학습 유도한다.

**5. 결과**

**1) A\* Navigation**

스크린샷, 픽셀이(가) 표시된 사진

자동 생성된 설명픽셀, 스크린샷이(가) 표시된 사진

자동 생성된 설명Figure 3 Turtle World

가구, 디자인이(가) 표시된 사진

자동 생성된 설명스크린샷, 아동 미술, 예술이(가) 표시된 사진

자동 생성된 설명

Figure 4 Turtle House

Fig 3~4의 결과와 같이 본 프로젝트에서 구현한 A\* 알고리즘 기반 네비게이션 시스템이 Turtle World와 Turtle House 환경에서 성공적으로 작동함을 확인했다. 특히 주목할 만한 점은 Cartographer SLAM의 미지 영역(occupancy = -1)에도 목표점 설정이 가능하도록 구현한 것이다. 이를 통해 로봇이 미지 영역을 탐색할 때, SLAM이 실시간으로 맵을 업데이트하고 새로운 장애물을 발견하면 A\* 알고리즘이 즉시 경로를 재계획하여 안전하고 효율적인 주행이 가능했다. 또한 최종 목표점 도달 시 목표점의 방향(화살표)에 맞춰 로봇이 제자리 회전하는 기능까지 구현하여, 정밀한 자세 제어도 성공적으로 수행함을 확인했다. 이러한 결과들을 통해 본 연구에서 제안한 네비게이션 시스템이 실제 환경에서 안정적이고 효과적으로 작동함을 검증하였다.

**2) Autonomous Exploration with DDPG**

예술, 디자인, 실내이(가) 표시된 사진

자동 생성된 설명

그래프, 텍스트, 라인, 도표이(가) 표시된 사진

자동 생성된 설명그래프, 라인, 텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명Figure 5 Turtle house Autonomous Exploration

Figure 6 Actor Critic Loss

Fig 5는 TurtleBot house 환경에서 DDPG를 이용해서 Autonomous Exploration을 수행한 결과이다. 학습은 총 7900 step을 진행했고, 초기 1000 step은 우선 Replay buffer을 쌓기 위한 step이고 Ornstein-Uhlenbeck Noise에 의해서 noise가 섞인 action을 수행했기 때문에 로봇의 이동경로도 역시 랜덤 하다. 1000 step이후 Actor의 출력에 의한 action을 수행하고 Fig 5에서 보이는 것처럼 일부 공간을 탐험한 것으로 보인다. Fig6은 Actor Critic의 Loss를 나타내며, 학습이 아직 안정화되지 않았음을 보여준다. Fig7은 Reward를 나타내는 그래프이며 텍스트, 그래프, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명초반에 Reward가 높은 이유는 grid가 새로 발견된 영역이 많기 때문으로 해석할 수 있다. 그리고 중간에 reward가 0에 가깝게 나타난 이유는 장애물과의 거리가 가까워졌기 때문으로 해석할 수 있다.

Figure 7 Reward

**6. Discussion**

A\*를 이용한 Navigation의 경우 프로젝트 목표를 달성했다. 다만 Motion Control에서 간단하게 P제어를 이용하고 종방향과 횡방향을 통합적으로 고려하지 못했다는 한계를 갖는다.   
추가 구현으로 Autonomous Exploration을 DDPG를 이용해서 수행하였지만 로봇이 TurtleBot house에서 일부 공간만 탐색하고 단순한 행동이 반복되는 것을 확인했다. Loss와 Reward 그래프 역시 안정적으로 수렴하지 않는 것을 보아 학습이 제대로 이뤄지지 않은 것을 알 수 있다. 이에 따른 향후 개선방안은 다음과 같다.   
먼Reward 함수를 구체적으로 변경해서 네트워크가 적절히 학습할 수 있도록 한다. 가령 Isaac Gym의 Humaniod, Shadow Hand의 Reward function을 확인했을 때 목표하는 행동을 도출하기 위해서 굉장히 세부적으로 보상을 주도록 설계되어 있다는 것을 확인했다. 따라서 목표 지향적이고 구체적인 Reward function 설계를 통해 학습 성능을 올릴 수 있을 것이다. 가령 reward 함수에 초기 위치에서 멀어질 수 록 더 큰 보상을 주거나, 새로 알게된 Grid의 개수가 증가하는 기울기에 따라 보상을 주는 방식이 있을 것이다. 또한robot이 Autonomous Exploration을 수행하다가 장애물과 충돌했을 때 충격량을 negative reward로 줄 수 있을 것이다.

**7. Reference**

Tang, Gang, Tang, Congqiang, Claramunt, Christophe, Hu, Xiong, Zhou, Peipei. (2021). Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment. IEEE access, 9, 59196-59210.

Lillicrap, TP. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971,.