포인트 클라우드 기반 현실 세계의 가상화 기술

김도형[1], 진영훈[2*]
[1]백석대학교 소프트웨어 융합 석사과정, [2]백석대학교 첨단IT학부 교수

Point Cloud-Based Virtualization of Real-World Environments

Do-Hyeong Kim[1], Young-hoon Jin[2*]
[1]Master's Student of Software Convergence,
[2]Professor, Division of Advanced IT, Baekseok University

요 약 최근 가상 현실과 로봇 공학의 발전으로 현실과 가상의 융합에 대한 관심이 높아지고 있다. 포인트 클라우드 데이터는 카메라와 LiDAR 센서로 환경의 3차원 정보를 획득하여 이러한 융합에 핵심적이다. 본 연구에서는 레이캐스팅 기법을 활용하여 가상 환경에서 실시간 포인트 클라우드 데이터를 생성하고, 이를 기반으로 에이전트가 환경을 효율적으로 탐색하는 방법을 제안한다. 에이전트는 포인트 클라우드 밀도를 분석해 미탐색 영역을 우선 탐색하고, RANSAC 알고리즘으로 바닥과 장애물을 구분하며, A* 알고리즘과 캣멀롬 스플라인으로 최적 경로를 계획한다. Unity 엔진에서 구현된 이 알고리즘은 복잡한 전처리 없이 실시간 처리 가능하며, 실험 결과 에이전트는 환경을 탐색하고 포인트 클라우드 데이터를 수집할 수 있었다. 본 연구는 포인트 클라우드 기반 가상 환경 구축과 실시간 탐색의 실용성을 검증하고 다양한 분야에서의 응용 가능성을 제시한다.

주제어 : 포인트 클라우드, 레이캐스팅, 경로 탐색, 실시간 처리, 가상 환경

Abstract Recent advancements in virtual reality (VR) and robotics have increased interest in integrating physical and virtual environments. Point cloud data from cameras and LiDAR are essential for capturing 3D environmental details. This study proposes a ray-casting-based method for generating real-time point cloud data, enabling efficient virtual environment exploration. The agent prioritizes unexplored regions by analyzing point cloud density, separates ground and obstacles using RANSAC, and plans paths with the A* algorithm and Catmull-Rom splines. Implemented in Unity, the method supports real-time processing without complex preprocessing. Experiments confirm its effectiveness, highlighting its potential for various applications.

Key Words : Point Cloud, Raycasting, Path Planning, Real-time Processing, Virtual Environment

1. 서론

최근 가상 현실(VR)과 로봇 공학의 발전은 현실 세계와 가상 세계의 융합 가능성을 크게 확대하며, 다양한 분야에서 혁신을 이끌고 있다. 교육, 엔터테인먼트, 제조업, 의료 등 여러 산업에서 가상 환경을 활용한 시뮬레이션과 데이터 처리는 점차 필수 기술로 자리 잡고 있다[1-3]. 이러한 융합의 핵심은 현실 세계의 공간 정보와 물체 형상을 정확히 수집하고 이를 가상 환경에 효과적으로 통합하는 기술이다. 이 과정에서 포인트 클라우드(Point Cloud) 데이터는 중요한 역할을 한다. 포인트 클라우드는 카메라와 라이다(LiDAR) 센서를 통해 수집된

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2022R1G1A1012974).
*교신저자 : 진영훈(devjay@bu.ac.kr)
접수일: 2024년 12월 09일 수정일: 2024년 12월 15일 심사완료일: 2024년 12월 19일

3차원 좌표를 기반으로 환경의 구조와 물체 형상을 정밀하게 표현하는 데이터 형식으로, 자율주행, 로봇 내비게이션, VR/AR 환경 구축 등 다양한 응용 분야에서 널리 활용되고 있다[4-8]. 그러나 이러한 데이터를 실시간으로 처리하고 상호작용에 활용하기 위해서는 높은 품질의 데이터와 효율적인 처리 기술이 요구된다. 기존 연구들은 데이터를 최적화하거나 필터링하여 처리 효율성을 높이는 데 초점을 맞췄으나, 복잡한 전처리 과정과 높은 계산 비용으로 인해 실시간 응용에는 여전히 한계가 있었다. 본 연구는 가상공공간에서 레이캐스팅(Ray Casting)을 활용하여 포인트클라우드 생성을 위한 탐색 방법을 제안한다. 가상 공간에서 에이전트는 주변 환경을 세밀하게 이해하고, 경로 계획 및 이동 전략에 필수적인 데이터를 수집한다. 따라서 수집된 데이터는 실시간으로 시각화되며, 제안된 접근법의 높은 효율성과 실용성을 보여주는지 입증한다. 또한, 실제 공간에서 포인트 클라우드 데이터를 수집하여 가상 공간에 실제 환경을 구현할 수 있는지 검증한다.

2. 관련연구

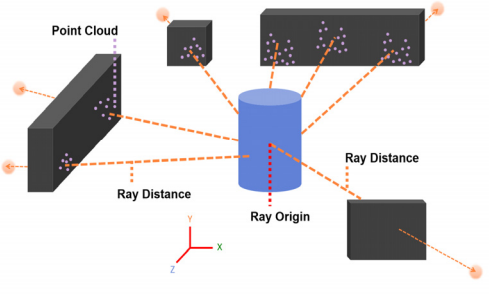
포인트 클라우드 데이터는 가상 현실과 로봇 공학에서 현실 세계와 가상 세계를 연결하는 중요한 매개체로 자리 잡고 있다. 이를 효과적으로 처리하고 다양한 응용 분야에 통합하기 위해 여러 연구가 진행되었으며, 크게 데이터 렌더링, 환경 재구성, 경로 계획의 세 가지 주요 축으로 나눌 수 있다. 첫째, 초기 연구들은 대규모 포인트 클라우드 데이터를 실시간으로 렌더링하는 데 초점을 맞췄다. 객체 공간 연산을 최소화하고 이미지 공간에서의 연산을 통해 고품질 렌더링을 실현하는 기법들이 제안되었으며[12-13], GPU 기반의 이미지 필터링과 보간 기법은 렌더링 속도와 품질을 동시에 향상시켰다. QSplat은 계층적 데이터 구조를 활용해 대규모 데이터를 점진적으로 렌더링하며 자원 소모를 줄이는 데 기여했다[14]. 둘째, SLAM은 환경 데이터를 활용하여 정밀한 위치추정과 환경 지도를 생성하는 기술로, 로봇 내비게이션 및 자율주행에 필수적인 역할을 한다. SLAM 기반 시스템은 LiDAR 데이터를 통해 지도작성과 탐색을 수행하며[15-16], LiDAR 시뮬레이터는 SLAM 알고리즘의 성능 평가를 위해 현실적인 데이터를 생성하는 데 활용되었다[17]. 셋째, 포인트 클라우드 데이터를 활용한 경로 계획 연구는 3D 환경에서 효율적인 탐색과 내비게이션을 목표로 발전해 왔다. 예를 들어, 지형의 기울기와 장애물을 분석하여 불필요한 경사를 회피하는 경로를 생성하거나[18], 복셀 그리드와 A* 알고리즘(Astar)을 확장해 위치 불확실성을 고려한 경로 탐색 방식이 제안되었다[19]. LOAM은 실시간으로 데이터를 처리하며 위치와 환경 지도를 동시에 생성해 자율주행 경로 계획의 중요한 기초를 제공했다[20]. 이처럼 기존 연구들은 데이터 렌더링, 환경 재구성, 경로 계획 등 다양한 측면에서 발전을 이루었지만, 높은 계산 비용과 복잡한 처리 과정은 자원 제한 환경에서 여전히 제약되고 있다. 본 연구는 이러한 한계를 극복하기 위해, 간단하고 자원 효율적인 접근법을 제안하며, 실시간으로 포인트 클라우드 데이터를 처리하고 환경을 탐색할 수 있는 새로운 방법을 제시하고자 한다.

3. 본론

본 연구에서는 실시간 포인트 클라우드 생성과 처리를 통해 환경을 효율적으로 탐색하는 방법을 제안한다. 에이전트는 경로 탐색 중 포인트 클라우드를 생성하며, 바닥과 장애물을 구분하여 공간 밀도를 분석하고 환경 데이터를 수집한다. 제안된 접근법은 간단한 알고리즘을 활용하여 높은 연산 비용이나 복잡한 전처리 과정 없이 실시간 응용이 가능하도록 설계되었다.

3.1 포인트 클라우드 생성

본 연구에서는 가상 환경에서 주변의 3차원 정보를 수집하기 위해 레이캐스팅 기법을 활용한다.



[Fig. 1] Point Cloud Generation Employing Raycasting

레이캐스팅은 카메라 또는 센서에서 광선을 발사하여 물체와 충돌하는 지점의 좌표를 기록하는 방식으로 라이다(LiDAR) 센서의 원리와 유사하며, 간단한 구현으로도

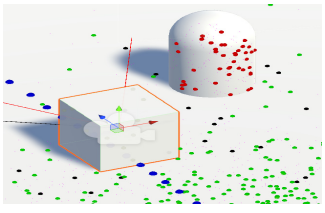
정밀한 데이터 수집이 가능하다. 레이캐스팅 기법을 통해 생성된 포인트 클라우드 데이터는 환경의 구조와 물체의 형상을 표현하며, 이는 이후 경로 탐색과 환경 이해를 위한 자료로 활용된다. [Fig. 1]은 레이캐스팅의 작동 원리를 시각적으로 나타낸 구조이다.

3.2 바닥과 장애물 처리

포인트 클라우드 데이터는 환경의 3차원 정보를 제공하지만, 바닥과 장애물을 효과적으로 구분하기 위해 추가적인 처리가 필요하다. 본 연구에서는 RANSAC (Random Sample Consensus) 알고리즘을 활용하여 바닥 평면을 추정하고, 이를 기반으로 장애물을 분리하였다. RANSAC은 평면 방정식을 이용하여 주어진 데이터에서 특정 평면에 부합하는 데이터를 반복적으로 평가하고 식별하는 알고리즘으로, 데이터 내 잡음을 제거하고 모델에 가장 적합한 데이터를 추출할 수 있다. 추정된 평면은 평면 방정식으로 나타내며, 각 포인트와 평면 간의 거리는 식 (1)을 통해 계산된다.

$$Dist(P_i) = \frac{|ax_i + by_i + cz_i + d|}{\sqrt{a^2 + b^2 + c^2}} \qquad (1)$$

계산된 값은 포인트 (P_i) 와 바닥 평면 사이의 수직 거리를 나타내며, 이 거리가 특정 임계값 이하일 경우 해당 포인트는 바닥으로 분류되고, 초과하면 장애물로 정의된다. [Fig. 2]는 바닥 평면과 장애물을 분리한 결과를 보여준다. 바닥과 장애물의 분리는 에이전트의 이동 경로를 조정하고 장애물의 위치와 크기를 반영하는 데 활용된다.



[Fig. 2] Ground Plane and Obstacle Differentiations (Green point : floor, Red point : Obstacle)

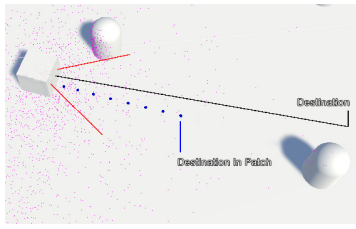
3.3 경로 지정

에이전트는 주변 환경에서 획득한 포인트 클라우드 데이터를 활용하여 가장 효율적인 이동 경로를 계획한다. 수집된 포인트 클라우드 데이터는 먼저 바닥과 장애물을 구분하는 전처리 과정을 거치며, 이를 토대로 에이전트는 A* 알고리즘을 적용하여 최적의 경로를 산출한다. 이때 비용 함수 $f(n)$은 노드 n 에서의 총 비용을 의미하며, 이는 시작 지점에서 해당 노드까지의 실제 누적 비용인 $g(n)$ 과 노드 n에서 목표 지점까지의 추정 비용인 $h(n)$ 합으로 정의된다. 또한 에이전트는 이동하려는 방향을 나타내는 벡터 u 와 포인트 클라우드 내 각 포인트의 위치를 나타내는 벡터 v 사이의 내적을 계산함으로써 목표 지점과 가장 가까운 포인트를 선택하고, 이를 현재 포인트 클라우드에서의 목적지로 설정한다. 내적 계산은 이동 방향과 포인트 간의 관계를 정량적으로 평가할 수 있게 하며, 내적 값이 최대가 되는 포인트는 이동 방향과 가장 가까운 지점으로 간주되어 경로의 다음 목표 지점으로 설정된다. 이는 식(2), 식(3)으로 나타낼 수 있다.

$$f(n) = g(n) + h(n) \qquad (2)$$

$$u \cdot v = |u||v|\cos\theta \qquad (3)$$

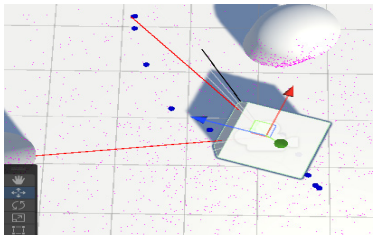
이를 통해 에이전트는 환경의 3차원 정보를 효과적으로 활용하며, 이동 경로를 최적화하고 장애물을 회피하는 동시에 목표 지점에 도달할 수 있는 효율적인 경로를 유지한다.



[Fig. 3] Path Estimation from Target and Point Cloud Data

[Fig. 3]은 에이전트가 바닥 영역의 포인트 클라우드를 분석하여 최적의 목표 지점을 선택하고, A* 알고리즘을 기반으로 경로를 계산하는 과정을 보여준다.

에이전트는 안전한 이동을 위해 캣멀롬 스플라인 (Catmull-Rom Spline) 보간 기법을 활용하여 [Fig. 4]와 같이 부드러운 곡선 형태의 이동 경로를 생성한다.

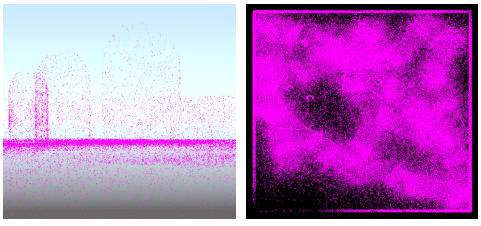


[Fig. 4] Catmull-Rom Spline Algorithm for Waypoint Designation

캣멀롬 스플라인은 주어진 점들을 정확히 통과하면서도 곡선의 연속성을 유지해 각 점에서의 접선 방향을 자연스럽게 결정한다. A* 알고리즘으로 생성된 경로는 장애물과의 최소 거리를 기반으로 검증 및 보완 과정을 거친다. 장애물과의 거리가 설정된 임계값 이하로 좁혀질 경우, 경로는 수정되며, 캣멀롬 스플라인을 다시 적용해 이동의 연속성을 확보한다. 이를 통해 에이전트는 다양한 환경에서도 매끄럽게 이동하도록 설계되었다.

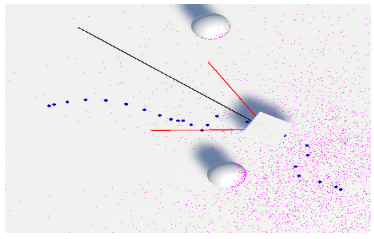
3.4 밀도 기반 공간 탐색

에이전트는 효과적인 환경 탐색과 최적의 경로 계획을 위해 포인트 클라우드의 밀도를 기반으로 공간을 분석한다.



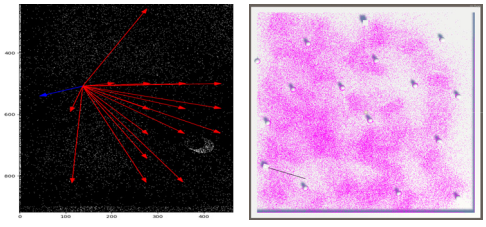
[Fig. 5] Scanning the Space from Multiple Perspectives (Left: Front View, Right: Top View)

[Fig. 5]는 에이전트가 전방 시점(Front View)과 상단 시점(Top View)의 두 관점을 활용하여 실시간으로 생성된 포인트 클라우드를 시각화하는 과정을 보여준다. 전방 시점에서는 바닥, 장애물, 벽 등을 식별하며, 지정된 영역의 포인트 클라우드 밀도가 임계값(0.0003)을 초과하지 않도록 데이터 수집을 제한한다. 수집된 데이터를 기반으로 공간 밀도를 분석하고 이를 통해 다음 경로를 결정한다. 두 번째 그림은 상단 시점으로 전체 공간의 포인트 클라우드 밀도와 분포를 파악하는 데 사용된다. 이를 통해 에이전트는 미탐색 영역을 식별하고 탐색 전략을 최적화하며, 분석 결과를 바탕으로 탐색 우선순위를 설정할 수 있다.



[Fig. 6] Point Cloud Generation and Distribution Along Waypoint Paths

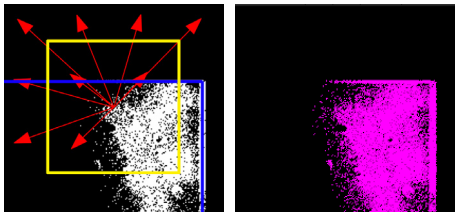
에이전트는 웨이포인트 경로를 따라 이동하며 포인트 클라우드를 생성한다. [Fig. 6]은 에이전트가 웨이포인트를 지나칠 때 주변 환경의 포인트 클라우드를 분석하고, 이를 기반으로 환경의 3D 지형 정보를 구축하는 과정을 보여준다. 이렇게 수집된 데이터는 에이전트가 공간적 이해도를 높이고, 이후 경로 계획과 장애물 회피에 필요한 정보를 확보하는 데 활용된다. 또한, 이동 경로 상에서 포인트 클라우드 밀도가 높은 영역은 추가 생성을 생략해 데이터 중복을 줄이고 처리 효율성을 높인다. 반대로, 밀도가 낮은 영역에서는 적극적으로 포인트 클라우드를 생성하여 미탐색 영역을 탐색한다.



[Fig. 7] Exploring the Space to Increase Point Cloud Density

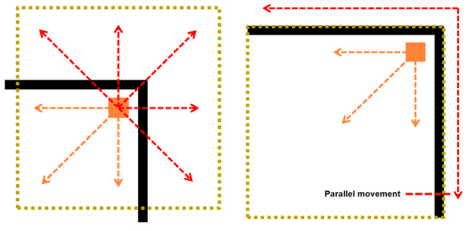
[Fig. 7]은 에이전트를 중심으로 복셀로 구분된 영역에서 포인트 클라우드 밀도를 계산하고, 밀도가 임계값 이하인 방향을 우선 탐색하도록 탐색 우선순위를 설정하는 과정을 보여준다. 첫 번째 사진은 복셀로 구분된 영역에서 포인트 클라우드 밀도를 계산하여 이를 바탕으로 탐색 방향을 설정하는 과정을, 두 번째 그림은 분석한 밀도를 기반으로 밀도가 낮은 방향에 경로를 지정하고 이동하는 모습을 나타낸다. 이를 통해 에이전트는 미탐색 영역을 효과적으로 탐색하며, 환경의 구조를 점진적으로 파악한다.

3.5 공간 좌표 재정렬



[Fig. 8] Coordinate and Point Cloud Distribution Determined Around the Agent

[Fig. 8]는 에이전트가 탐색 중 경로의 끝에 도달하거나 구석진 영역에 고립될 경우 발생하는 좌표 체계 부정확성을 보여준다. 이는 포인트 클라우드가 환경의 중심이 아닌 에이전트를 기준으로 생성되기 때문이며, 이로 인해 가상 환경과 실제 환경 간 괴리가 발생할 수 있다. 이를 해소하기 위해 본 연구에서는 전역 중심점과 경로 중심점을 도입하여 에이전트 경로를 전역 좌표계에 재정렬(shift)하는 방법을 제안한다. 전역 중심점은 전체 포인트 클라우드 상의 평균 위치로, 환경의 기준점 역할을 한다. 반면, 경로 중심점은 에이전트의 이동 궤적 상의 평균 위치로, 에이전트 중심 좌표계의 편차를 드러낸다.



[Fig. 9] Path Realignment Using Global Coordinates (Left: Original, Right: Adjusted)

재정렬 과정은 [Fig. 9]와 같이 전역 중심점과 경로 중심점 간의 차이를 활용하여, 에이전트 경로를 전역 좌표계 기준으로 평행 이동하는 방식으로 이루어진다. 이를 통해, 에이전트가 환경 구석이나 막다른 지점에 도달하더라도 포인트 클라우드의 전역적 일관성이 유지된다. 결과적으로 오차 누적을 줄이고, 환경을 보다 안정적으로 이해할 수 있게 된다. 이 두 중심점 사이의 편차를 반영한 재정렬 과정을 통해, 에이전트가 환경의 변두리나 막다른 구역에 도달하더라도 전체 포인트 클라우드에 대한 전역적 공간 정합성을 유지할 수 있다.

4. 실험

실험은 〈Table 1〉과 같은 절차를 따라 수행되었다.

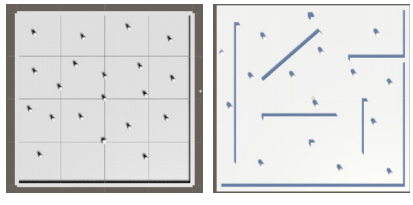
〈Table 1〉 Experimental Procedure

```
Initialize Environment:
Create a virtual environment using the Unity engine.
Deploy the agent at the starting position.
Start Exploration:
Begin the agent's exploration task.
Data Collection:
while the agent is moving do
    Generate point cloud data.
    Collect and store the generated data.
end while
Divide Space into Voxel Grid:
Divide the virtual environment into a 6 × 6 × 6 voxel grid.
Calculate Density:
for each voxel in the grid do
    Compute the density of point cloud data in the voxel.
    if density 〈 threshold then
        Mark the voxel as a low-density area.
    end if
end for
Generate Path:
Create a path that prioritizes the exploration of low-density voxels.
Exploration Task:
for each waypoint in the generated path do
    Move the agent to the waypoint.
    Update point cloud data during movement.
end for
Terminate Experiment:
if goal is achieved OR failure conditions are met then
    Stop the exploration.
end if
```

4.1 실험 환경

본 실험은 Microsoft Windows 11 PRO 운영체제 환경에서 진행되었다. 중앙처리장치(CPU)로 Intel i9-12900K를, 그래픽처리장치(VGA)로 NVIDIA RTX 2080을 사용하였으며, 약 6400 MT/s 클럭으로 동작하는 총 32GB 용량의 주기억장치(RAM)를 구성하여 안정적인 컴퓨팅 성능을 확보하였다. 시뮬레이션은 Unity 2022.3.22f를 활용하였으며, 소스코드는 C#과 Python 언어를 기반으로 구현하였다.

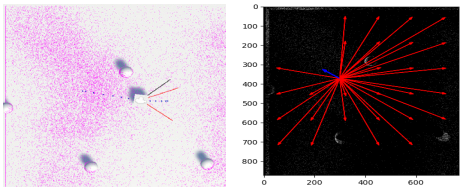
4.2 가상 환경 구성



[Fig. 10] Virtually Simulated Environment Implemented Using Unity (Left : Normal Area, Right : Expended Area)

Unity를 이용하여 가로 4m, 세로 4m의 넓은 환경을 구성하였으며, 에이전트 및 장애물의 크기는 각각 1m로 설정하였다. [Fig. 10]은 이 실제로 제작한 가상 환경을 나타낸다. 환경은 단순 장애물 배치로 구성된 기본 환경(Normal Area)과 벽 및 다양한 장애물로 구성된 확장 환경(Expended Area)으로 구분하였다. 또한 에이전트의 이동속도는 초당 1.2m로, 회전 속도는 약 1.75 rad/s로 설정하였다. 이를 통해 에이전트가 다양한 환경 조건에서 장애물 회피 및 경로 탐색을 수행하는 상황을 재현하고자 하였다.

4.3 복셀 영역 분석 경로 지정



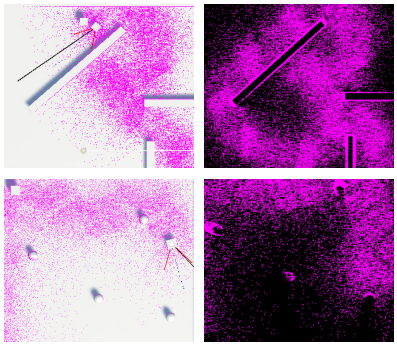
[Fig. 11] Path Planning Through Density Analysis Using Voxel Grid

에이전트는 목표 지점을 향해 이동하면서 포인트 클라우드 데이터를 수집하고, 이를 바탕으로 밀도가 낮은 영역을 우선적으로 탐색하여 데이터 분포를 점진적으로 균일화하였다. [Fig. 11]는 에이전트가 6×6 복셀로 나뉜 영역에서 포인트 클라우드 밀도를 계산하고, 부족한 영역을 중심으로 경로를 지정하여 밀도를 점차 높이는 과정을 보여주고 있다.

4.4 시뮬레이션 시 실시간 성능 검증

CPU, GPU, 메모리 사용량, 프레임 처리 시간, 경로 탐색 알고리즘 처리 속도를 측정한 결과, 평균 CPU 사용량 약 25%, GPU 사용량 약 11%, 메모리 사용량 약 0.84GB를 기록하였다. 전반적인 프레임 처리 시간은 약 16ms(60FPS) 수준으로 유지되었으나, 경로 재지정 시 경로 탐색 알고리즘 수행으로 인해 일시적으로 약 33ms(30FPS)까지 증가하는 현상이 관찰되었다. 경로 탐색 처리 시간은 평균 약 1초에서 최대 5초까지 소요되며, 포인트 클라우드 밀도가 증가할수록 연산 시간이 늘어나는 경향이 있었다. 그러나 경로 탐색 알고리즘의 평균 CPU 연산 시간은 전체 프레임 처리 시간(4~5ms) 중 약 0.24ms(약 5%)로 매우 적은 비중을 차지하였으며, GPU 사용량도 평균 약 11%로 자원 부담이 크지 않았다.

4.5 포인트 클라우드 수집 및 결과



[Fig. 12] Density Map of Point Clouds Generated by Movement Path

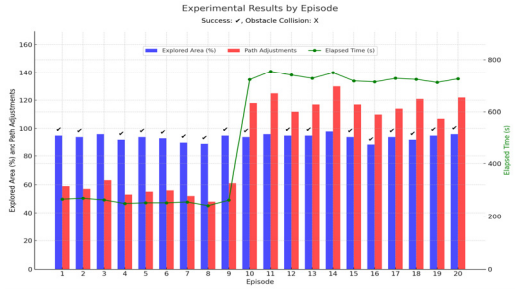
[Fig. 12]는 에이전트의 이동 경로와 포인트 클라우드 데이터의 변화를 나타낸다. 에이전트가 이동하면서 포인트 클라우드 밀도가 점진적으로 증가하며, 바닥, 장애물, 위치 등의 데이터를 수집하는 과정을 보여준다.

〈Table 2〉 Experimental Results Table by Episode

	Area (%)	Obs.	Cols.	Path Adj.	Time(s)
1	95	14	0	59	267
2	94	14	0	57	270
3	96	14	1	63	264
4	92	14	0	53	250
5	94	14	0	55	253
6	93	14	0	56	256
7	90	14	0	52	242
8	89	14	0	48	263
9	85	14	0	61	263
10	94	14	0	118	723
11	96	23	1	125	755
12	95	23	0	112	741
13	95	23	0	117	728
14	98	23	1	130	750
15	94	23	0	117	718
16	88	23	0	110	714
17	94	23	0	114	728
18	92	23	0	121	724
19	95	23	0	107	712
20	96	23	0	122	726

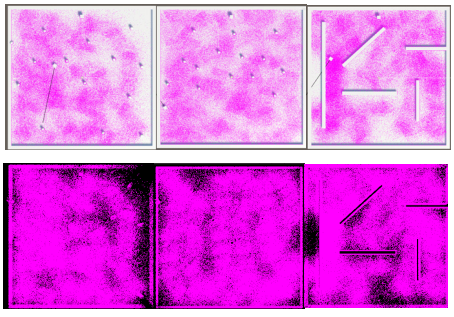
〈Table 2〉는 총 20회에 걸쳐 진행된 실험의 결과를 나타낸다. 그 결과, 에이전트는 17회에서 성공을 거두었으며, 3, 11, 14번 에피소드에서 장애물과의 충돌로 인

해 실패 사례가 확인되었다. 또한 장애물의 수가 증가함에 따라 경과 시간이 유의미하게 연장되는 경향이 나타났으며, 특히 경로 지정 횟수가 장애물의 개수에 비례하여 증가하는 양상이 관찰되었다.



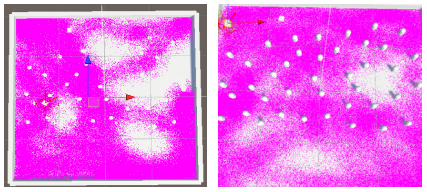
[Fig. 13] Experimental Results by Episode

[Fig. 13]은 탐색한 영역(Explored Area), 경로 지정 횟수(Path Adjustments), 경과 시간(Elapsed Time) 간의 상관관계를 나타낸다. 탐색한 영역은 에이전트가 공간을 탐색한 비율을 의미하며, 경로 지정 횟수는 알고리즘이 영역을 탐색하기 위해 생성한 경로의 수로 정의된다. 에피소드 1부터 9는 장애물 개수가 14개, 에피소드 10부터 20은 장애물 개수가 23개로 구성되었다. 에피소드 3, 11, 14를 제외하면 대부분의 에피소드에서 높은 탐색률과 성공률을 기록하였다. 이는 장애물 개수와 경로 지정 횟수의 증가는 경과 시간의 증가에 영향을 미치는 것으로 분석되었으며, 장애물이 많고 경로 지정 회수가 많을수록 실패 가능성이 있는 경향을 보여준다.



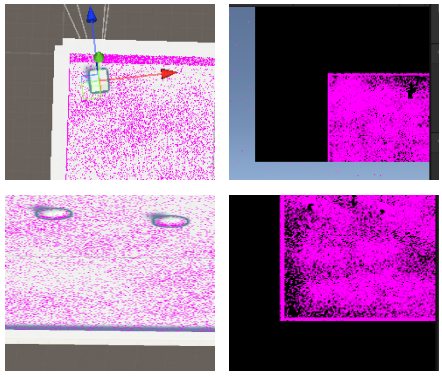
[Fig. 14] Distribution Map of Increased Density Across the Entire Explored Space

[Fig. 14]는 에이전트가 이동하면서 포인트 클라우드 밀도를 증가시키고, 이를 통해 전체적인 환경 탐색을 수행한 과정을 시각적으로 보여준다. 그러나 실험 중 몇 가지 한계점이 확인되었다.



[Fig. 15] Large Voids Unaffected by Point Cloud Formation

실험 중 에이전트가 특정 영역을 탐색하지 않아 '공동 현상'이 발생하는 문제가 관찰되었다. [Fig. 15]에 나타난 바와 같이, 공동 현상은 에이전트가 벽과 모서리 사이의 고립된 위치에서 이동이 제한될 때 발생하였다. 이를 해결하기 위해, 에이전트가 여러 방향으로 탐색을 시도했으나 경로를 찾지 못하는 경우, 수집된 포인트 클라우드 데이터의 중심 위치를 기준으로 A* 알고리즘을 적용하여 고립된 영역에서 탈출한 뒤 탐색을 재개하는 방법을 적용할 수 있다. 또한 에이전트는 [Fig. 16]과 같이 새로운 방향을 결정하지 못하고 경로 지정 수정을 수 회 시도하다 제한된 범위 내에서 회전하거나 정지하는 양상이 관찰되었다.



[Fig. 16] Isolation Due to Failure in Identifying Exploratory Regions

이러한 현상은 이미 환경 전역이 탐색되어 추가적인 경로 계획이 불가능하거나, 주변 포인트 클라우드 밀도가 지나치게 높아 경로 설정에 제약이 생기는 상황에서 발생하는 것으로 분석된다.

5. 결론

본 연구에서는 실시간으로 포인트 클라우드를 생성하고 이동 경로를 탐색하며 환경을 점진적으로 파악하는

방법을 제안하였다. 에이전트는 밀도가 낮은 영역을 우선 탐색하여 미지의 공간을 효율적으로 분석하고, 간단한 알고리즘으로 환경을 디지털화하며 장애물을 회피하였다. 또한 90%의 성공률을 기록하며 제안된 방법의 실용성과 효율성을 입증하였다. 특히, 에이전트의 전략은 복잡한 환경에서도 안정적으로 작동하여 실시간 환경 탐색에 유용한 가능성을 보여준다. 향후 연구에서는 복잡한 환경에서도 적용 가능한 알고리즘을 개발하고, 로봇 플랫폼과 실제 환경 데이터를 활용하여 기술의 실용성을 더욱 검증할 계획이다.

REFERENCES

[1] https://automation.honeywell.com/us/en/news/featured-stories/warehouse-automation/autonomous-mobile-robots-driving-warehouse-productivity

[2] J.Holland, L.Kingston, C.McCarthy, E.Armstrong, P.O'Dwyer, F.Merz, and M.McConnell, "Service robots in the healthcare sector," Robotics, Vol.10, No.1, p.47, 2021.

[3] P.Marmaglio, D.Consolati, C.Amici, and M.Tiboni, "Autonomous vehicles for healthcare applications: A review on mobile robotic systems and drones in hospital and clinical environments," Electronics, Vol.12, No.23, p.4791, 2023.

[4] K.Katona, H.A.Neamah, and P.Korondi, "Obstacle avoidance and path planning methods for autonomous navigation of mobile robot," Sensors, Vol.24, No.11, p.3573, 2024.

[5] H.Qin, S.Shao, T.Wang, X.Yu, Y.Jiang, and Z.Cao, "Review of autonomous path planning algorithms for mobile robots," Drones, Vol.7, No.3, p.211, 2023.

[6] M.Whitty, S.Cossell, K.S.Dang, J.Guivant, and J.Katupitiya, "Autonomous navigation using a real-time 3D point cloud," IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), pp.188-193, 2008.

[7] M.Eckhoff, D.Knobbe, H.Zwirmann, A.Swkir, and S.Haddadin, "Towards connecting control to perception: High-performance whole-body collision avoidance using control-compatible obstacles," IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), pp.1-7, 2023.

[8] M.V.L.deCarvalho, R.Simoni, and L.Yoshioka, "Autonomous navigation and collision avoidance for mobile robots: Classification and review," arXiv preprint, arXiv:2410.07297, 2024.

[9] K.Rajathi, N.Gomathi, M.Mahdal, and R.Guras, "Path segmentation from point cloud data for autonomous navigation," Applied Sciences, Vol.13, No.6, p.3977, 2023.

[10] X.Xie, H.Wei, and Y.Yang, "Real-time LiDAR point-cloud moving object segmentation for autonomous driving," Sensors, Vol.23, No.1, p.547, 2023.

[11] M.Mahmeen, R.D.DominguezSanchez, M.Friebe, M.Pech, and S.Haider, "Collision avoidance route planning for autonomous medical devices using multiple depth cameras," IEEE Access, Vol.10, pp.29903-29915, 2022.

[12] J.P.Grossman and W.J.Dally, "Point sample rendering," Rendering Techniques, Springer, pp.181-192, 1998.

[13] S.Rusinkiewicz and M.Levoy, "QSplat: A multiresolution point rendering system for large meshes," Proc. 27th Annual Conf. Computer Graphics and Interactive Techniques, pp.343-352, 2000.

[14] Y.Guo, H.Wang, Q.Hu, H.Liu, L.Liu, and M.Bennamoun, "Deep learning for 3D point clouds: A survey," IEEE Trans. Pattern Anal. Mach. Intell., Vol.43, No.12, pp.4338-4364, 2020.

[15] Ł.Grudzień, L.Ambroziak, and K.Orłowski, "LiDAR point cloud generation for SLAM algorithm evaluation," Sensors, Vol.21, No.5, p.1680, 2021.

[16] X.Chen, A.Milioto, E.Palazzolo, P.Giguère, J.Behley, and C.Stachniss, "SuMa++: Efficient lidar-based semantic slam," IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), pp.4530-4537, 2019.

[17] D.Wang et al., "Virtual lidar simulator for autonomous driving," IEEE Intelligent Vehicles Symp., pp.1627-1634, 2019.

[18] J.Petereit, F.Zimmermann, and S.Wachsmuth, "3D traversability analysis and path planning based on mechanical effort for UGVs in forest environments," J. Field Robot., Vol.34, No.5, pp.994-1018, 2017.

[19] J.Gonzalez and A.Stentz, "Planning with uncertainty in position using high-resolution maps," IEEE Int. Conf. Robotics and Automation (ICRA), pp.1015-1022, 2007.

[20] J.Zhang and S.Singh, "LOAM: Lidar odometry and mapping in real-time," Robotics: Science and Systems Conf., 2014.

김 도 형(Do-Hyeong Kim) [준회원]

- 2023년 2월 : 백석대학교 (컴퓨터공학부 학사)
- 2023년 3월 ~ 현재 : 백석대학교 일반대학원 (소프트웨어 융합 석사과정)

〈관심분야〉

VR, AR, Cybersecurity, Artificial intelligence, Robotics

진 영 훈(Young-Hoon Jin) [종신회원]

- 2019년 8월 : 중앙대학교 (영상학박사)
- 2021년 3월 ~ 현재 : 백석대학교 첨단IT 조교수

〈관심분야〉
XR, RL, IoT, Network, Reverse Engineering