Journal of KIIT. Vol. 23, No. 3, pp. 195-203, Mar. 31, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.3.195

Unity-Ros2 기반 무인보트 충돌 회피 시뮬레이터 개발

이세진*, 이화성**, 유선진***

Development of a Unity - ROS2 - based Collision Avoidance Simulator for Unmanned Surface Vehicles

Sejin Lee*, Hwasung Lee**, and Sunjin Yu***

본 과제(결과물)는 2024년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업의 결과입니다(2021RIS-003)

요 약

본 논문은 유니티와 ROS2를 연동한 분산 구조를 통해 무인보트 충돌 회피 알고리즘을 시뮬레이션하고, 가상 LiDAR(VLP-16 모델)로 획득한 3차원 포인트 클라우드를 활용하여 성능을 검증한다. 유니티 엔진은 센서 시뮬레이션과 물리 엔진을 담당하고, ROS2는 임계거리 기반의 지역 회피 로직을 구동함으로써 실제 무인보트 임베디드 환경과 유사한 아키텍처를 구현하였다. 단순 장애물 배치에서는 높은 성공률로 우회 가능했으나, 복잡·협소한 시나리오에서 회피 성능이 급격히 떨어져 전역 경로 계획(A*) 등 고도화 기법이 필요함을 확인했다. 이를 통해 해상시험 전 단계에서 센서 노이즈·해상도 등 파라미터를 조정하며 알고리즘 안정화를 도모할 수 있음을 시뮬레이션 결과로 제시한다. 또한 제안된 분산 구조는 범용 알고리즘(예: SLAM, 강화학습)과도 쉽게 결합 가능하여, 무인보트 자율운항 분야의 후속 연구에 기여할 것으로 기대한다.

Abstract

This paper presents a distributed architecture combining Unity and ROS2 to simulate a collision avoidance algorithm for Unmanned Surface Vehicles(USVs), utilizing a virtual LiDAR modeled after the Velodyne VLP-16 sensor. Unity is responsible for sensor simulation and the physics engine, while ROS2 executes a locally triggered avoidance logic based on critical distance thresholds, closely mirroring real onboard software structures. The results indicate that simpler obstacle layouts yield high success rates, yet performance significantly degrades in more complex or narrow scenarios, suggesting the need for advanced techniques such as A* global planning. Through this framework, users can systematically adjust parameters like noise or resolution before actual sea trials, thereby reducing cost and risk. Furthermore, the proposed approach is compatible with other general-purpose algorithms (e.g., SLAM, reinforcement learning), offering potential for broader development in the USV autonomy field.

Keywords
unmanned surface vehicle, USV, unity, ROS2, virtual LiDAR, collision avoidance

* 국립창원대학교 문화융합기술협동과정 석사과정
- ORCID: https://orcid.org/0009-0000-5720-0626
** 동명대학교 조선해양공학과 학사과정
- ORCID: http://orcid.org/0000-0001-4517-3288
*** 국립창원대학교 문화테크노학과 교수(교신저자)
- ORCID: https://orcid.org/0000-0001-9292-4099

· Received: Jan. 31, 2025, Revised: Feb. 27, 2025, Accepted: Mar. 02, 2025
· Corresponding Author: Sunjin Yu
Dept. of Culture Technology, Changwon National University, 20 Changwondaehak-ro, Uichang-gu, Changwon-si, Gyeongsangnam-do, 51140, Korea
Tel.: +82-55-213-3098, Email: sjyu@changwon.ac.kr

Ⅰ. 서 론

무인선박(USV, Unmanned Surface Vehicle)은 해양 산업 전반에서 비용 절감과 안전성 확보를 동시에 이룰 수 있는 핵심 기술로 각광받고 있다[1]. 최근 EU의 MUNIN(Maritime Unmanned Navigation through Intelligence in Networks) 프로젝트와 Rolls-Royce, DnV-GL 등의 무인 컨테이너 선박 연구가 활발하며, 국내에서도 현대중공업의 충돌 회피 지원시스템(HiCASS, Hyundai Intelligent Collision Avoidance Support System)와 삼성중공업의 선박 모니터링 시스템(VPS, Vessel Portal Service) 등이 자율운항 기술을 상용화하기 위한 노력을 본격화하고 있다[2]. 이처럼 국제해사기구(IMO, International Maritime Organization)가 제시한 자율화 등급에 따른 자율운항선박 정의가 마련되면서, 무인선박 기술은 실무 적용 단계로 빠르게 발전하는 추세다[3].

그러나 해상 운항 중 발생하는 충돌 사고는 단순한 경제적 손실을 넘어 인명 피해와 대규모 환경 오염을 야기할 수 있어, 무인선박이 실제 해역에서 운용되려면 충돌 회피 기술을 핵심 역량으로 갖춰야 한다[4]. 전통적으로 선박의 충돌 회피는 여러 보조 시스템을 활용하지만, 최종 판단은 승조원의 경험에 크게 의존해 왔다. 그 결과 중소형 선박에서는 보조 시스템 설치가 어렵고, 숙련도 부족 등으로 인해 사고 원인 대부분이 인간 오류로 지목되기도 한다[5]. 이를 극복하기 위해 자율운항 선박에 센서 융합과 경로 계획 기술을 결합한 충돌 회피 알고리즘이 시도되고 있으나, 실제 해역에서 충분한 반복 실험은 비용·안전 문제로 제약이 크다[6].

이에 시뮬레이션 환경을 통해 미리 알고리즘을 검증하고, 이후 해상 시험으로 확장하는 전략이 중요해지고 있다. 대표적인 해양 시뮬레이터로 Gazebo, VRX 등이 있으나 설정 난이도가 높고, 해상 물리 모델을 구체화하는 과정이 복잡하다는 지적이 있다[7]. 유니티(Unity) 엔진은 원래 게임 개발 및 인터랙티브 시뮬레이션을 위한 플랫폼이지만, 실시간 렌더링과 물리 엔진을 통해 사실적인 3D 환경을 구현하고, 사용자 인터페이스 관리가 용이해 해양 분야의 자율운항 연구에서도 활용이 늘고 있다[8]. 특히 로봇 운영체제(ROS)와 결합하여 센서 모델링 및 제어 로직을 검증하거나, 강화학습을 통해 자동 소화 시스템이나 배관 경로 설계를 시험한 사례도 보고되었다[9][10]. 그럼에도 무인선박에 3차원 센서(예: 3D 광학 센서)를 도입해 충돌 회피 전 과정을 시뮬레이션으로 구현한 연구는 상대적으로 적으며, 실제 해역 적용 이전 단계에서 비용·시간을 얼마나 절감할 수 있는지 체계적으로 분석한 사례는 많지 않다[6]. 한편, 레이더(RADAR), 카메라, 초음파 등 다양한 센서가 있으나, 최근에는 높은 정밀도와 실시간 3차원 데이터 수집 능력을 가진 LiDAR가 자율주행·로봇 분야에서 활발히 연구되고 있다[11]. 해양 환경에서도 LiDAR를 활용하면 장애물의 높이나 형태를 파악하기 수월하지만, 파도·조도·해상 반사 등에 의해 센서 노이즈가 커지고, 초기 연구 시 고비용·고위험 실험이 필요하다는 지적이 있다[12]. 이러한 한계를 해소하기 위해, 가상환경에서 LiDAR를 레이캐스팅(Ray casting) 방식으로 모델링하고, 무인선박이 지형·장애물 정보를 바탕으로 안전한 경로를 계획·추종할 수 있게 하는 시뮬레이션 접근이 부상하고 있다[13]. ARKit, ARCore 등 모바일 플랫폼을 이용한 증강현실(AR) 기반 기술도 존재하지만, 파랑·조명 변화가 심한 해양 환경에는 적용이 까다로우므로[14], 유니티에서 직접 가상 센서를 구성해 로직을 시험하는 방법이 보다 안정적으로 초기에 검증을 수행할 수 있다.

이에 본 연구에서는 유니티 시뮬레이터를 이용해 3D LiDAR 센서를 가상 구현하고, 무인선박이 장애물을 회피하는 자율운항 알고리즘을 시험함으로써 실제 해역 운용 이전에 충분한 반복 테스트를 진행하는 방안을 제시한다. 유니티 엔진에서 레이캐스팅으로 포인트 클라우드를 생성하고, 이를 기반으로 경로 계획을 적용하여 충돌 회피를 수행한다. 이렇게 가상환경을 통해 자율운항 코드를 검증하면, 해상 실험에서 발생할 비용과 위험 부담을 낮출 수 있을 것으로 기대된다. 본 논문의 구성은 다음과 같다. 2장에서는 무인선박 충돌 회피와 유니티 기반 시뮬레이션에 대한 배경을 정리하고, 3장에서 본 연구의 시스템 구조 및 알고리즘 설계를 설명한다. 이어 4장에서는 시뮬레이션 결과와 성능 분석을 제시하고, 5장에서 결론과 향후 연구 방향을 제안한다

II. 배경 이론

2.1 무인선박 충돌 회피 연구

자율운항 선박의 충돌 회피 기술은 전역 경로와 실시간 지역 경로를 결합하여 안전하게 운항하는 것이 일반적이며, 국제 해상충돌 예비규칙(COLREGs)을 준수하도록 알고리즘을 보정하기도 한다[1][2]. 기존 연구들은 센서 다양화, ROS 통합, 실제 소형 보트 시험 등을 진행해 왔으나[3][4] 해상 시험 자체가 잦은 반복이 어려워 연구 초기 단계에서 알고리즘을 신속히 보완하기가 쉽지 않았다[5]. LiDAR 기반 장애물 인식 역시 파도·반사 등으로 인해 해상 노이즈가 심해, 초기 연구 단계부터 고가 장비와 복잡한 측정 환경이 필요하다는 단점이 지적된다[8].

2.2 유니티 엔진을 활용한 시뮬레이션

유니티는 원래 게임 및 인터랙티브 시뮬레이션을 위한 플랫폼이지만, 실시간 물리 엔진·그래픽 처리·사용자 인터페이스 관리 측면에서 강력한 기능을 제공한다[14]. 최근에는 자동차나 드론 분야에서 유니티+ROS를 결합해 센서·제어 로직을 실험하고, 교육·연구 목적의 로봇 시뮬레이터를 만드는 사례가 보고되고 있다[9][13]. 국내 해양 분야에서도 강화학습 기반 제어 등에 유니티를 적용한 연구가 있으나, 실제 무인선박 충돌 회피 전 과정을 시뮬레이션한 예는 제한적이다[10].

ARKit, ARCore 등 증강현실 플랫폼은 모바일 기기에서 공간 매핑과 SLAM을 간단히 구현할 수 있게 해주지만 [6][7] 해양 환경의 조도·반사 문제로 인해 높은 정확도를 얻기 어렵다는 보고가 있다[14]. 반면, 유니티는 물리 및 광학 특성을 사용자가 임의로 설정할 수 있으므로, 해양 시뮬레이션에서 3D 센서(예: 레이캐스팅 LiDAR) 모델을 비교적 쉽게 구현할 수 있다. 본 연구는 바로 이 점에 주목하여, 유니티 시뮬레이터에서 3D LiDAR를 통한 충돌 회피 로직을 설계하고, 해상 시험 전 단계에서의 반복 디버깅 효율 향상을 목표로 한다.

III. 시스템 개발

3.1 환경 구성 및 유니티 - ROS 연동

본 연구에서는 유니티 엔진(2022.3.9f1 버전)과 ROS2 Humble 간 연동을 통해 무인선박 자율운항 알고리즘을 시뮬레이션한다. 유니티는 Windows 10 환경에서 구동하며, 가상의 해양 환경(수면, 장애물)과 무인선박 오브젝트, Raycasting 기반 센서 생성을 담당한다. PC 사양은 다음과 같다.

운영체제: Windows 10 (64bit)
CPU: Intel(R) Core(TM) i7-10700K @ 3.80GHz
GPU: NVIDIA GeForce RTX 3090
RAM: 32GB

자율운항 알고리즘(충돌 회피·경로 계획 등)은 Ubuntu 22.04에 ROS2 Humble을 설치한 노트북에서 수행하며, 해당 노트북의 사양은 아래와 같다.

운영체제: Ubuntu 22.04 (64bit)
CPU: AMD Ryzen 9 5900X (12코어/24스레드)
GPU: NVIDIA GeForce RTX 3080
RAM: 32GB
ROS2: Humble Hawksbill

그림 1에서 보는것 처럼 두 시스템은 LAN 혹은 Wi-Fi로 연결되고, ROS-TCP-Connector나 ros2socket 같은 브릿지를 활용해 유니티 ↔ ROS 간 데이터를 주고받는다. 즉, 유니티에서 가상의 LiDAR·카메라 센서 정보를 실시간 토픽(sensor_msgs/PointCloud2 등)으로 Publish하고, ROS에서 이를 Subscribe하여 보트의 속도·회전 명령(geometry_msgs/Twist 등)을 계산한 뒤 다시 유니티로 Publish한다. 유니티가 이 명령을 Subscribe해 무인선박 오브젝트의 Rigidbody에 힘과 토크를 가하는 방식으로 시뮬레이션이 전개된다.

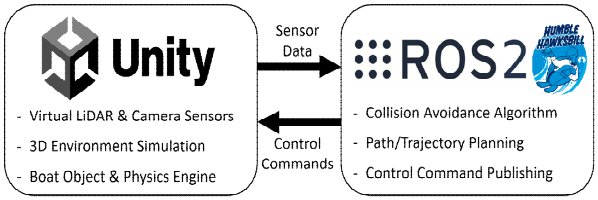

그림 1. 유니티 - ROS 전송 흐름도
Fig. 1. Unity - ROS transmission flow

이런 분산 구조를 통해, 센서 모델·환경 설정 등은 유니티 측에서, 알고리즘(충돌 회피·경로 계획)은 ROS 측에서 독립적으로 개발하여, 실제 무인선박 하드웨어와 유사한 네트워크 구조를 모사할 수 있다는 장점이 있다.

3.2 Raycasting 기반 LIDAR 센서 구현

그림 2는 Velodyne VLP-16 센서를 모델링하여, 유니티 엔진에서 레이캐스팅 기법을 통해 3차원 포인트 클라우드를 생성한다. VLP-16은 수평 360°와 수직 ±15° 범위에서 최대 약 100m까지 스캔하며, 16개 채널로 수직 레이어를 구성한다. 이를 시뮬레이션에 반영하기 위해, 일정 각도 해상도(δ_v,δ_h)로 광선을 나누어 발사한 뒤, 충돌 지점의 좌표와 거리를 측정한다.

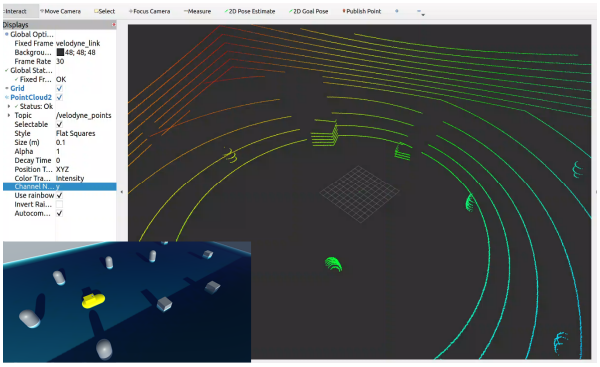

그림 2. 유니티 LIDAR - ROS2 전송 화면
Fig. 2. Unity LIDAR - ROS2 transfer screen

식 (1)은 이러한 방식으로 얻을 수 있는 총 레이 개수를 나타낸다.$\Delta\theta_v\delta_v\Delta\theta_h$

$$N_{\text{rays}} = \frac{\Delta\theta_v}{\delta_v} \times \frac{\Delta\theta_h}{\delta h} \times (\text{채널 수}) \tag{1}$$

여기서 $\Delta\theta_v$는 수직 범위(±15°), $\Delta\theta_h$는 수평 범위(360°), δ_v와 δ_h는 각각 수직·수평 각도 해상도이며, (채널 수)는 실제 VLP-16의 16개 레이어를 모사하기 위함이다. 유니티에서 $\{0,\ldots,N_{rays}-1\}$각각에 대해 광선을 생성하면, 다음 단계로 Raycast 함수를 통해 충돌 정보를 얻는다. 식 (2)는 센서 원점 O_c에서 광선방향 $\hat{d}_i$로 레이를 발사했을 때, 충돌 지점을 월드 좌표계에서P_i로 표현한다.

$$P_i = O_c + r_i \times \hat{d}_i \tag{2}$$

P_i : 센서 원점(월드 좌표)
$\hat{d}_i$: 단위 방향 벡터
(수직·수평 각도와 센서 자세가 반영됨)
r_i : 광선이 충돌한 거리(없으면 $r_{\max}$로 표기

충돌 지점이 없으면 r_i=$r_{\max}$ 로 설정하거나 포인트 클라우드에서 제외한다. 또한, 실제 LiDAR에서 발생하는 잡음을 간단히 Gausian noise를 추가하여 측정값을 보정한다.

$$r_i' = r_i + \epsilon, \quad \epsilon \sim N(0,\sigma^2) \tag{3}$$

여기서 σ는 노이즈 파라미터이며, 충돌 거리가 $r_i' < r_{\min}$이면 최소 감지 범위를 벗어난 것으로 간주해 포인트 클라우드에 반영하지 않는다. 이와 같이 얻어진 포인트(P_i)는 유니티의 List<Vector3> 혹은 NativeArray로 보관되며, 매 프레임(또는 일정 주기)에 새로 계산된 점들이 누적·갱신된다. 실제 VLP-16의 16개 채널 간 세부 수직각 차이를 완전히 동일하게 반영할 수도 있지만, 본 연구에서는 일정한 각도 해상도로 단순화하였다. 이를 통해 가상 LiDAR 센서는 물리적인 복잡성 없이도 충분히 VLP-16 수준의 스캔 밀도와 범위를 제공할 수 있다

3.3 단순 충돌 회피 알고리즘

본 논문에서는 지역 충돌 회피(Local collision avoidance) 방식에 초점을 맞추어, 전역 경로 계획 없이도 보트가 임계거리 이하로 접근한 장애물을 우선 회피하도록 설계하였다. 구현은 $\pm\beta^\circ$ (±35°)범위의 전방 스캔 데이터를 분석하고, 좌·우 어느 쪽이 상대적으로 안전한지를 결정해 선회 방향을 결정하는 로직이다. Raycasting으로 생성된 포인트 클라우드에서, 보트 센서 원점O_c기준으로 수평각 $\alpha \in [-\beta,+\beta]$에 해당하는 점들을 추출한다. 해당 점들의 거리를 식 (4)으로 정의한다.

$$r_{front} = \min_{\alpha \in [-\beta,+\beta]} \|P_i - O_c\| \tag{4}$$

이때 r_{front} 안전거리보다 작으면 보트가 장애물과 가깝다고 판단한 후 회피 모드로 진입한다. 이어서 좌우 회피 방향 결정 단계에서는, 장애물이 전방에 있음을 감지한 후, α<0(왼쪽)과 α>0(오른쪽) 각 영역에서 평균 거리나 점의 밀도를 비교해, 여유 공간이 더 큰 쪽으로 선회 명령(ω)을 우선하도록 구성한다. 필요 시 국제 해상충돌예방규칙(COLREGs)을 반영해 기본적으로 '우현 회피'를 우선시하도록 설정할 수도 있다. 마지막으로 회피 동작 및 복귀 단계에서는 유니티 물리 엔진에서 전진 속도 v보다 회전 속도 ω를 우선 적용해 일정 시간 선회한 뒤, 장애물이 사라지면 원래 항로(전진)로 복귀한다. 전진·회전 명령을 힘·토크로 변환함으로써 보트 오브젝트는 매 프레임 위치와 자세가 갱신되는데, 여기서 v는 전진 속도 명령, ω는 회전(선회) 속도 명령이고 Kv와 Kω는 추진·회전 계수로 사용된다. 이러한 단순 지역 회피 방식은 복잡한 해양조선학적 동역학이나 전역 경로 없이도, 가상 LiDAR 데이터만으로 장애물을 즉시 우회하기에 초기 연구 단계에서 구현이 쉽다는 장점을 갖는다.

3.4 코드 구조 및 통합 운영

유니티 내부 C# 스크립트와 ROS2 노드가 네트워크를 통해 통신하는 분산 구조를 지향한다. 이렇게 함으로써 실제 무인선박 임베디드 환경에서 보편적으로 사용되는 ROS의 소프트웨어 생태계를 그대로 재현하면서, 유니티 엔진을 이용해 가상 센서를 생성하고 보트 오브젝트의 움직임을 시뮬레이션할 수 있다. 먼저 RaycastLiDARSensor.cs에서 RaycastCommand를 사용해 광선 수백~수천 개를 병렬로 발사하여 충돌 정보를 받아들인 뒤, 생성된 포인트들을 리스트 형태로 보관한다. 이러한 포인트 클라우드는 노이즈나 최대·최소 거리 등의 파라미터를 적용한 후에 sensor_msgs 형태로 ROS에 Publish하거나, 초기 단계를 위해 유니티 내부에서 그대로 처리할 수도 있다.

ROS2 측 노드(CollisionAvoidance.py)는 포인트 클라우드 메시지를 Subscribe하여 전방 최소 거리, 좌우 평균 거리 등을 계산하고, 전진 속도와 회전 속도를 포함하는 제어 명령을 geometry_msgs/Twist 등으로 Publish한다. 유니티에서는 BoatController.cs가 이 메시지를 Subscribe하여, 전진 속도와 회전 속도를 Rigidbody 컴포넌트의 힘과 토크로 변환함으로써 보트 오브젝트를 움직인다. 이때 구현 편의에 따라 전진 속도와 회전 속도의 최대 허용 범위나 PID 계수 등을 Inspector 파라미터로 조정할 수 있다.

초기 연구 단계에서 ROS 없이 유니티 내부 CollisionAvoidance.cs를 통해 센서 데이터부터 충돌 회피 계산, 제어 적용까지 단일 구조로 운용하는 것도 가능하나, 궁극적으로는 분산 환경으로 전환해 실제 무인선박의 소프트웨어 아키텍처와 유사한 형태를 구성하는 편이 이점이 크다. 예를 들어, ROS 측에서 SLAM이나 경로 계획 라이브러리를 손쉽게 적용하고, 유니티에서 다양한 해양 환경과 센서 파라미터를 모사할 수 있기 때문이다. 또한, 센서와 제어 알고리즘이 독립적으로 운용되므로, 노이즈 수준이나 광선 해상도, 장애물 배치 등 시뮬레이션 요소를 바꿔가며 반복 실험이 용이하고, 실제 하드웨어로 이식할 때 코드 구조 변경을 최소화할 수 있다는 장점이 있다.

결론적으로, 가상 센서를 담당하는 유니티 측 스크립트와 충돌 회피나 경로 계획을 담당하는 ROS 노드를 분리 운용하면, 해상 시험 이전 단계에서 필요한 센서·제어 로직을 자유롭게 교체·확장할 수 있다.

Ⅳ. 실험 환경 및 결과 분석

4.1 실험 환경 및 시나리오

본 연구에서 제안한 가상 LiDAR 센서와 충돌 회피 알고리즘은 두 대의 컴퓨터 간 네트워크로 연결된 환경에서 동작한다. 하나는 Windows 10 기반의 데스크톱 PC로, 유니티 엔진(2022.3.9f1 버전)을 구동하여 해양 환경과 무인선박 오브젝트를 시뮬레이션한다. 다른 하나는 Ubuntu 22.04를 설치한 노트북이며, ROS2 Humble 배포판에서 자율운항 알고리즘을 개발·실행한다. 두 장치는 WIFI으로 연결되어, 유니티에서 생성된 가상 센서 데이터가 실시간으로 ROS 노트북에 전송되고, 노트북 측에서 계산된 회피 명령이 다시 유니티 보트 오브젝트에 반영된다.

시뮬레이션된 해양 환경은 파도나 조류 등을 단순화하여 구성하며, 장애물을 일정 범위 안에 임의로 배치하여 무인선박이 회피 동작을 수행하도록 설정한다. 구체적으로, 초기 보트 위치를 원점 근방에 두고 목표 지점을 x=+100, z=0 인근으로 설정하되, 경로 중간에 부표나 벽 형태의 장애물을 놓아 충돌 위험을 유도한다. 가상 LiDAR 센서는 Velodyne VLP-16 스펙을 간소화해 수평 360°, 수직 ±15°의 레이를 발사하며, 최대 거리 약 100m, 노이즈 수준은 0.05로 설정했다. 각 시나리오는 장애물 배치와 보트 초기 위치를 반복 변경하여, 전반적인 충돌 회피 성공률과 항해 시간을 측정한다. 이 과정을 통해 본 논문에서 제안한 Raycasting 기반 LiDAR 모델과 ROS 연동 방식이 다양한 상황에서 얼마나 안정적으로 작동하는지 정량적으로 평가할 수 있다.

4.2 시나리오별 실험 결과

본 절에서는 4.1에서 설명한 장애물 배치 시나리오를 대상으로, 가상 LiDAR 센서를 통해 획득된 포인트 클라우드와 충돌 회피 알고리즘을 연동하여 무인선박이 얼마나 효과적으로 장애물을 피할 수 있는지를 측정하고 분석한다. 실험은 그림 3에서 보는 Unity가 구동되는 데스크톱 PC와, ROS2 알고리즘이 실행되는 노트북을 LAN으로 연결하여 진행하였으며, 각 시나리오마다 보트가 시작점에서 목표 지점까지 항해하는 과정을 10회 반복 측정하여 평균 항해 시간, 충돌 횟수, 성공률 등을 산출하였다.

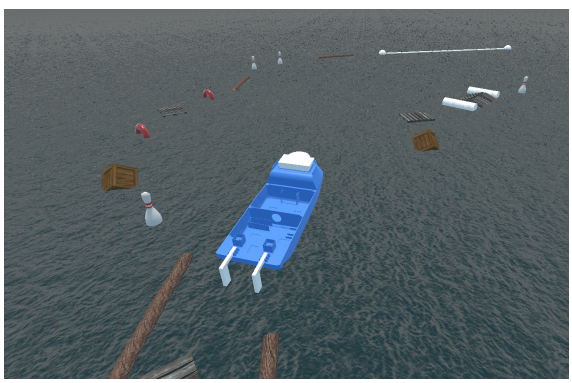

그림 3. 시나리오 C 화면
Fig. 3. Scenario C

표 1은 세 가지 시나리오(A, B, C)를 대상으로 수행한 충돌 회피 실험 결과를 요약한 것이다. 시나리오 A에서는 항로 중앙에 직경 5m 내외의 단일 부표를 배치하였고, 10회 시험 중 9회가 무사 항해에 성공하여 약 22초의 평균 항해 시간이 측정되었다. 단 한 번의 충돌은 노이즈 수준(σ=0.1)에서 부표가 제대로 인식되지 않은 사례였다. 반면 시나리오 B는 다수(35개) 장애물을 불규칙하게 배치함으로써 난이도를 높였다. 그 결과 70~80% 수준의 성공률을 기록하였고, 10회 중 2~3회 정도 충돌이 발생했으며, 선회 동작이 잦아져 항해 시간은 평균 30~35초로 증가하였다. 마지막으로 시나리오 C는 좁은 수로를 모사하여 보트가 S자 경로로 이동하도록 설정한 경우로, 60~70%의 성공률과 함께 약 40초의 항해 시간이 소요되었다. 노이즈가 큰 상황에서 높은 회전 속도를 적용하면 충돌 발생 빈도가 증가했으며, 이는 실제 해양 환경에서 센서 오차와 기동성 한계가 서로 맞물리는 복잡성을 잘 보여주는 사례라 할 수 있다. 전반적으로는 단순한 지역 회피 로직과 가상 LiDAR 데이터만으로도 어느 정도 장애물 우회가 가능함이 확인되었지만, 장애물이 복잡해지고 센서 노이즈가 커질수록 충돌 위험이 빠르게 높아지는 경향이 뚜렷했다. 이를 통해, 향후 전역 경로 계획(A*)나 보다 정교한 충돌 회피 알고리즘(DWA, VFH 등)과의 결합이 필요하다는 점을 시사한다. 또한 유니티 - ROS2 연동 방식 덕분에 다양한 센서 파라미터와 장애물 배치를 반복 시험할 수 있어, 해상 실험 전 단계에서 시스템 안정화와 비용·시간 절감 효과가 기대된다는 결론을 얻었다.

표 1. 시나리오별 실험 결과
Table 1. Experimental results by scenario

Scenario	Collision count	Average voyage time	Special notes
A (Single obstacle)	1 time (9/10 successful runs)	Approx 22s	Failure to recognize buoy at noise level (σ=0.1)
B (Multiple obstacles)	2–3 times (70–80% success rate)	Approx 30–35s	30–35s Increased difficulty due to diverse obstacle types
C (Narrow channel simulation)	30–40% collisions (60–70% success rate)	Approx 40s	Higher collision rate observed with increased noise and high turn speed

4.3 검토

본 절에서는 앞서 시나리오 A, B, C를 통해 얻은 충돌 횟수, 항해 시간, 성공률 등 주요 지표를 바탕으로, 가상 LiDAR 기반 지역 충돌 회피 알고리즘의 한계와 확장 가능성을 논의한다. 먼저 시나리오별 실험 결과를 종합해보면, 장애물 수·배치 복잡도가 증가할수록 성공률이 하락하고 항해 시간은 길어지는 경향이 뚜렷하게 나타났다. 이는 보트가 여러 차례 선회를 반복하면서 경로를 수정해야 하고, 센서 노이즈나 회전 속도 한계값이 일정 수준을 넘으면 충돌 위험이 급격히 증가하기 때문이다.

특히 시나리오 C에서 성공률이 60~70%로 떨어진 이유는 공간이 협소한 상황에서 단순 지역 회피만으로는 국부 최소점문제에 빠지기 쉽다는 점을 시사한다. 예를 들어 전방 장애물을 회피하는 도중에, 뒤이어 나타나는 벽면이나 다른 장애물을 충분한 선행 거리에서 파악하지 못하면, 후속 회피 동작이 늦어져 충돌로 이어질 가능성이 커진다. 이러한 현상은 실제 해양 환경에서도 긴 협수로·포트·항만 운항 시 비슷하게 발생할 수 있으므로, 추가적인 전역 경로 계획(A*)나 더 정교한 지역 알고리즘(DWA, VFH 등)을 병행하면 회피 성공률을 높일 수 있을 것으로 예상된다.

또한 센서 노이즈(σ값) 변화에 따른 충돌 횟수 증가 추세도 관찰되었다. 노이즈 수준을 0.0에서 0.1로 높일 경우, 시나리오 A처럼 비교적 단순한 상황에서도 인식 실패로 인한 소수 충돌 사례가 발생하였으며, 복잡한 배치에서는 충돌 횟수가 더욱 늘어났다. 이는 가상 LiDAR 노이즈가 일정 범위를 넘어가면 전방 최소 거리 계산의 안정성이 흔들려, 회피 시점이 지연되거나 방향 결정이 흔들리는 문제가 생김을 의미한다. 실해역에서도 파랑이나 반사 등에 의해 비슷한 오차가 발생할 수 있으므로, 노이즈가 큰 상황을 대비한 알고리즘 보완이 필요하다.

마지막으로, 유니티 - ROS2 연동을 통해 센서 파라미터나 충돌 회피 로직을 자유롭게 교체·조정할 수 있다는 점이 본 실험에서 큰 장점으로 확인되었다. 충돌 횟수가 많아지는 상황에서 회전 속도 한계를 낮추거나, 임계거리와 노이즈(σ)를 줄이도록 조정하면 성공률이 일부 개선될 수 있다. 이처럼 시뮬레이션 단계에서 다양한 시나리오·파라미터를 반복 시험함으로써, 실제 해양 시험에 앞서 알고리즘을 사전 검증하고 개선할 수 있는 가능성을 확인하였으며, 추후에는 전역 경로 계획이나 강화학습 기반 제어 기법 등 더 복잡한 로직과도 쉽게 결합할 수 있을 것으로 기대된다.

Ⅴ. 결 론

본 논문에서는 유니티 엔진과 로봇 운영체제(ROS2)를 연동하여, 가상 LiDAR 센서를 이용한 무인선박 충돌 회피 로직을 시뮬레이션하고 여러 장애물 배치 시나리오에서 그 성능을 검증하였다. 유니티 측에서는 해양 환경과 보트 오브젝트를 모델링하고 Raycasting 기법을 적용해 VLP-16과 유사한 3차원 포인트 클라우드 데이터를 생성했으며, ROS2 측에서는 센서 데이터를 받아 지역 충돌 회피 로직(전방 임계거리 분석, 좌우 비교)을 수행해 제어 명령을 다시 유니티로 전송하는 분산 구조를 구성하였다. 이를 통해 해상시험 전 단계에서 센서 노이즈, 회전 속도 한계, 장애물 배치 등을 자유롭게 조정하면서 충돌 횟수, 항해 시간, 성공률 등을 정량적으로 분석할 수 있음을 확인하였다.

실험 결과, 간단한 임계거리 기반 충돌 회피 로직과 가상 LiDAR 데이터만으로도 기본적인 장애물 우회가 가능함이 드러났으나, 장애물이 복잡하거나 수로가 좁은 환경에서는 성공률이 급격히 떨어져 전역 경로 계획(A* 등)과 같은 고도화 기법이 필요함을 확인했다. 또한 센서 노이즈(σ 값)나 보트의 회전 속도 한계가 달라질 때, 충돌 횟수와 항해 시간이 크게 변동하여 실제 해상에서도 센서 오차와 보트 기동 특성이 복합적으로 작용함을 시사한다. 한편, 본 논문에서 사용한 시뮬레이션 환경(표 1)은 파랑·풍향·유체력 등 해양조선학적 요소를 단순화해 구성하였기 때문에, 실제 상황에 대한 신뢰성이 충분하지 않다는 지적이 있었다. 이에 대한 보완책으로, 향후에는 파랑·해류·동적 장애물과 같은 복잡한 해양 조건을 추가로 모델링하고, COLREGs 규칙을 반영한 충돌 회피 로직을 결합하여 시뮬레이터의 현실성과 적용 범위를 확대할 계획이다.

본 연구의 의의는 유니티 - ROS2 연동을 통해 물리 환경과 소프트웨어 구조가 분리된 상황에서도, 센서·제어 알고리즘을 유연하게 교체·확장할 수 있다는 점에 있다. 특히 실제 무인선박 임베디드 환경과 유사한 네트워크 구조를 모사하여, 개발 단계에서 반복 실험과 파라미터 조정이 손쉽게 이뤄진다는 장점을 확인하였으며, 시뮬레이션 단계에서 축적한 노하우가 추후 실제 해상 시험과도 자연스럽게 연결될 수 있음을 시사한다. 추후 연구에서는 보트 운동 모델을 더욱 정교화하고, 전역·지역 알고리즘(예: DWA, 강화학습)을 하이브리드 방식으로 접목하는 등 다양한 로직을 추가 실험함으로써, 무인선박 충돌 회피의 안정성과 효율성을 동시에 확보할 수 있는 방안을 적극 모색할 예정이다.

References

[1] T. Statheros, G. Howells, and K. McDonald-Maier, "Autonomous ship collision avoidance navigation concepts, technologies and techniques", Journal of Navigation, Vol. 61, No. 1, pp. 129-142, Jan. 2008. https://doi.org/10.1017/S037346330700447X.

[2] S. Hong, "Analysis of factors influencing ship collision avoidance judgment of maritime officers", Journal of the Korean Institute of Plant Engineering, Vol. 20, No. 1, pp. 3-10, Mar. 2015.

[3] R. Polvara, T. Johansen, M. Breivik, and A. Sørensen, "Obstacle avoidance approaches for autonomous navigation of unmanned surface vehicles", Journal of Navigation, Vol. 71, No. 1, pp. 241-256, Jan. 2018. https://doi.org/10.1017/S0373463317000753.

[4] M. Kim, T. Jung, B. Jeong, and H. Park, "Autonomous shipping and its impact on regulations, technologies, and industries", Journal of International Maritime Safety, Environmental Affairs, and Shipping, Vol. 4, No. 2, pp. 17-25, Jun. 2020. https://doi.org/10.1080/25725084.2020.1779427.

[5] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles", IEEE Journal of Oceanic Engineering, Vol. 39, pp. 110-119, Jan. 2014. https://doi.org/10.1109/JOE.2013.2254214.

[6] H. Jo, J. Kim, S. Kim, J. Woo, and J. Park, "Development of autonomous algorithm for boat using robot operating system", Journal of the Society of Naval Architects of Korea, Vol. 58, No. 2, pp. 121-128, Apr. 2021. https://doi.org/10.3744/SNAK.2021.58.2.121.

[7] Y. Jo, H. Jung, S. Lee, and D. Jeong, "LiDAR sensor and pure pursuit algorithm-based autonomous navigation system for small ships", Journal of Korean Institute of Information Technology, Vol. 20, No. 4, pp. 1-11, Apr. 2022. https://doi.org/10.14801/jkiit.2022.20.4.1.

[8] H. Song, K. Lee, and D. Kim, "Implementation of an obstacle avoidance system based on a low-cost sensor for autonomous navigation of an unmanned ship", The Transactions of The KIEE, Vol. 68, No. 3, pp. 480-488, Mar. 2019. https://doi.org/10.5370/KIEE.2019.68.3.480.

[9] A. Hussein, F. Garcia, and C. Olaverri-Monreal, "ROS and unity based framework for intelligent vehicles control and simulation", 2018 IEEE Int. Conf. on Vehicular Electronics and Safety (ICVES), Madrid, Spain, pp. 1-7, Sep. 2018. https://doi.org/10.1109/ICVES.2018.8519522.

[10] D. Shin, B. Park, C. Lim, S. Oh, G. Kim, and S. Shin, "Pipe routing using reinforcement learning on initial design stage", Journal of the Society of Naval Architects of Korea, Vol. 57, No. 4, pp. 191-197, Aug. 2020. https://doi.org/10.3744/SNAK.2020.57.4.191.

[11] D. Son and D. Kim, "Autonomous navigation of unmanned surface vehicle based on fuzzy control considering COLREG", Journal of Korean Institute of Intelligent Systems, Vol. 31, No. 1, pp. 11-20, Feb. 2021. https://doi.org/10.5391/JKIIS.2021.31.1.011.

[12] Z. Oufqir, A. E. Abderrahmani, and K. Satori,

"ARKit and ARCore in serve to augmented reality", 2020 Int. Conf. on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, pp. 1-7, Jun. 2020. https://doi.org/10.1109/ISCV49265.2020.9204243.

[13] Y. Song, J. Qin, L. Dong, and Y. Zhao, "Design of Mobile Augmented Reality System Based on SLAM", 2021 Int. Conf. on Culture-oriented Science & Technology (ICCST), Beijing, China, pp. 43-46, Nov. 2021. https://doi.org/10.1109/ICCST53801.2021.00020.

[14] Unity Technologies, "Unity Engine Overview and Documentation", https://unity.com [accessed: Jan. 19, 2025.]

저자소개

이 세 진(Sejin Lee)

2023년 2월 : 국립창원대학교 문화테크노학과(학사)
2023년 3월 ~ 현재 : 국립창원대학교 문화융합기술협동과정 석사과정
관심분야 : 증강/가상현실, 실감콘텐츠 문화융합기술협동과정 박사과정
관심분야 : 대형언어모델, 증강/가상현실, 실감콘텐츠

이 화 성(Hwasung Lee)

2022년 3월 : 동명대학교 조선해양공학과 학사과정
관심분야 : 조선해양공학, 증강/가상현실, 자율운항

유 선 진 (Sunjin Yu)

2003년 8월 : 고려대학교 전자정보공학(공학사)
2006년 2월 : 연세대학교 생체인식공학(공학석사)
2011년 2월 : 연세대학교 전기전자공학(공학박사)
2011년 1월 ~ 2012년 5월 : LG전자기술원 미래IT융합연구소 선임연구원
2012년 5월 ~ 2013년 2월 : 연세대학교 전기전자공학과 연구교수
2013년 3월 ~ 2016년 8월 : 제주한라대학교 방송영상학과 조교수
2016년 9월 ~ 2019년 8월 : 동명대학교 디지털미디어공학부 부교수
2019년 9월 ~ 2024년 9월 : 국립창원대학교 문화테크노학과 부교수
2024년 10월 ~ 현재 : 국립창원대학교 문화테크노학과 정교수
관심분야 : 컴퓨터비전, 증강/가상현실, HCI