대한조선학회논문집 Research Paper
Journal of the Society of Naval Architects of Korea
eISSN:2287-7355, Vol. 60, No. 6, pp. 406-415, December 2023
https://doi.org/10.3744/SNAK.2023.60.6.406

Unity-ROS 시뮬레이터 기반의 자율운항 시스템 개발 및 검증

김기원[1,*] · 방현태[2,*] · 서정화[2] · 윤원근[2,†]
한화시스템 해양연구소[1]
충남대학교 자율운항시스템공학과[2]

Development of Autonomous Navigation System Using Simulation Based on Unity-ROS

Kiwon Kim[1,*] · Hyuntae Bang[2,*] · Jeonghwa Seo[2] · Wonkeun Youn[2,†]
Naval R&D Center, Hanwha systems[1]
Department of Autonomous Vehicle System Engineering, Chungnam National Univeristy[2]

In this study, we focused on developing and verifying ship collision avoidance algorithms using Unity simulator and ROS(Robot Operating System). ROS is used to establish an environment where communication between different operating systems is possible, and a dynamic model of a ship is constructed within Unity simulator. The Lidar data collected in Unity environment is passed to the system based on python through ROS. In the system based on python, control command values were created through the logic of the collision avoidance algorithm using data, and the values were transferred back to Unity to control the movement of the virtual ship. Through the developed simulation system, the reliability of the collision avoidance algorithm of ships with two different forms in an environment similar to the actual physical world was confirmed. As a result, it was confirmed on the simulator that it could be avoided without collision even in an environment with various types of obstacles, and that the avoidance characteristics according to the dynamics of the ship could be analyzed.

Keywords : Collision avoidance(충돌회피), Simulator(시뮬레이터), Ship dynamics(선박 동역학), Robot Operating System(로봇 OS)

1. 서 론

선박 운항 중 충돌 사고는 경제적인 피해뿐만 아니라, 이로 인한 인명 피해 및 환경 오염 등의 대규모 재앙으로 이어질 수 있다(Yang et al., 2004). 이러한 이유로 선박의 충돌 회피 기술은 선박 운항 시 안전을 확보하는 데 있어 중요한 요소로 인식됐다. 최근에는 자율운항 선박 기술의 발전이 가속화되고 있으며, 자율운항 선박의 충돌 회피 기술의 적용은 안전성을 한 단계 더 향상할 것으로 기대된다. 또한, 충돌 회피 기술의 개발은 항로 최적화와 연계하여 해상 교통 관리 시스템의 혁신과 선박 운항 비용 절감에도 크게 이바지할 것으로 전망된다.

기존 선박의 충돌 회피는 다양한 시스템의 보조를 받고는 있으나 최종적으로 선박에 탑승한 조종 관계자의 경험에 기반을 둔 판단에 의존하였다. 하지만 중소형 선박의 경우 충돌 회피를 보조하는 시스템의 설치가 어렵고 조종 관계자의 경험도 충분치 않은 경우가 있어 선박의 충돌 사고는 계속하여 발생하고 있고, 이러한 사고 원인 대부분이 인간의 오류로 조사되고 있다 (Hong, 2015). 이러한 문제를 해결하기 위해 선박 충돌 회피 기술에도 자율운항 기술을 적용하고자 하는 연구가 진행되고 있다. 실제 선박 운항 상황에서는 안전성 등의 요인들로 인해 충돌 회피 알고리즘을 실증하기 어려우므로, 동역학 모델 기반 시뮬레이션은 충돌 회피 알고리즘을 개발하고 검증하는 데 있어 중요한 도구로 인식되고 있다.

다수의 선행 연구들이 충돌 회피 알고리즘의 개발과 검증에 중점을 두고 있다. Jo et al. (2021)은 보트에 다양한 센서를 탑재한 뒤 로봇운영체제(Robot Operating System, ROS)를 이용하여 자율운항 알고리즘을 개발하여, 실제 시험을 수행하고 그 성능을 검증하였다. 또한, Jo et al. (2022)은 저가형 LiDAR 센서를 활용

Received : 15 June 2023 | Revised : 8 September 2023 | Accepted : 25 September 2023
† Corresponding author : Wonkeun Youn, wkyoun@cnu.ac.kr
* These authors contributed equally to this work.

하여 소형선박을 위한 자율운항시스템을 개발하였는데, 여기서는 물체를 회피하는 데 있어 부드러운 선회 반경을 가지는 것을 특징으로 하는 알고리즘을 제안하였다. Cho et al. (2021)은 비용함수를 활용한 협수로에서의 충돌 회피 알고리즘을 제안하였으며, 가상의 협수로를 곡선 좌표계를 활용하여 표현하고, 시뮬레이션을 통해 알고리즘을 검증하였다. Son and Kim (2021)은 ROS 기반의 Gazebo 시뮬레이터를 활용하여 2D LiDAR와 무인선박을 구현하고 시뮬레이션을 통해 충돌 회피 알고리즘을 검증하고 제안하였다. 이러한 연구에서 ROS로 축약되는 로봇 운영체제를 사용하는 경우가 많은데, ROS는 노드(node)와 노드 간에 통신을 가능하게 하는 운영체계의 일종으로 센서 등의 하드웨어와 제어를 위한 시스템의 통신을 가능하게 해주는 유용한 도구로서 이동체의 하드웨어 시스템 구축이나 시뮬레이션에 많이 사용된다.

가상 환경을 구현하는 시뮬레이션을 위한 유용한 도구로, Unity를 이용한 시뮬레이션 환경 구축에 관한 연구도 다수 수행되고 있다. Unity는 Unity Technologies사에서 제작하고 배포하는 3D 모델링을 위한 제작 도구로서 실제 물리 세계를 모사한 시각적 환경이나 물리적 환경을 제작하여 시뮬레이션할 수 있으므로 다양한 이동체의 시뮬레이터 제작에 유용한 도구로써 사용되고 있다. Hussein et al. (2018)은 Unity와 Gazebo를 활용하여 자율주행 자동차의 시뮬레이션 환경을 구현하여 두 시뮬레이터의 기능 및 성능을 비교하였다. Hu and Meng (2016)은 Unity와 ROS를 결합하여 다중 UAV를 위한 시뮬레이션 환경을 구현하고 실제 비행 시스템과 유사한 시뮬레이터를 개발하였다. 이를 통해 비행 제어 및 항법 알고리즘을 검증하고 비행 테스트에 상당한 시간을 절약할 수 있음을 보였다. 이러한 연구는 Unity가 높은 수준의 그래픽 및 물리 엔진을 제공함으로써 복잡하고 현실적인 시뮬레이션 환경을 구축할 수 있음을 보여주었다. 선박해양공학 분야에서 Unity 개발 환경은 자동 소화 시스템의 제어 알고리즘 개발 (Lee et al., 2020), 배관경로 설계 (Shin et al., 2020)에 적용되는 등, 선박 운용의 시뮬레이션보다는 강화학습을 위한 작업 환경의 제공에 초점을 맞추어 활용되어왔다.

본 연구에서는 물리 세계에 대한 모사가 가능한 Unity를 사용하여 선박 운용의 검증에 필요한 장애물과 선박에 장착되는 센서를 포함하는 시뮬레이션 환경을 구축하고 이를 하드웨어의 운용과 동일한 통신환경을 제공할 수 있는 ROS기반의 통신환경에 연동하여 시뮬레이터를 구축하여 자율운항시스템을 검증하였다. Unity와 ROS를 사용하여 개발된 시뮬레이터 환경을 활용하여 동역학 모델 기반 선박의 충돌 회피 알고리즘을 포함하는 자율운항 시스템을 개발하고 검증하였다. Unity 시뮬레이터에서 가상 LiDAR 센서를 구현하고, 자율운항 선박의 제어와 운동을 구현하기 위한 시뮬레이션 환경을 구성하였다. 구체적으로는 ROS을 활용하여 서로 다른 운영체제 간 통신이 가능한 환경을 구축하고, Unity 내에 선박의 동역학 모델을 구성하였다. 구성된 동역학 모델과 Unity에 구현된 가상의 LiDAR 센서를 활용하여 선박의 상황을 시뮬레이션하고, 이에 대한 데이터를 ROS를 통해 Python 기반으로 구성된 자율운항 시스템으로 전달한다. 자율운항 시스템에서는 이 데이터를 이용하여 충돌 회피 알고리즘을 통한 제어 명령 값을 생성하고, 이를 다시 Unity로 전달하여 가상 선박의 움직임을 제어하였다. 이러한 방식의 시뮬레이션을 기반으로 선박의 자율운항 시스템을 검증하여 하드웨어를 제작하고 실제 해상 환경에서 실험하기 위한 비용과 시간을 절감할 수 있으며 선박의 운용 실험 이전에 필요한 운항 시스템의 검증에도 사용할 수 있어 선박의 자율운항 연구에 다양하게 적용될 수 있으리라 전망된다.

이에 본문 2장에서는 연구에서 사용한 선박 동역학 모델을 소개한다. 3장에서는 Unity 시뮬레이터에서 가상 LiDAR 구성 방법, ROS와 python을 이용한 통신 및 충돌 회피 알고리즘의 적용 방법에 대해 상세하게 기술한다. 4장에서는 제안하는 방법론의 성능을 평가하기 위한 실험 설계 및 결과를 제시하며, 5장에서는 본 연구의 결론과 향후 연구 방향에 대해 논의한다.

2. 동역학 모델

충돌 회피 알고리즘의 효율성과 신뢰성 평가를 위해, 본 연구에서는 두 가지 다른 조종 메커니즘을 가진 동역학 모델을 사용하였다. 첫 번째는 방향타의 타각 조절을 통해 선회 모멘트를 발생하는 전통적인 선박 모델이며, 두 번째는 좌우 현의 프로펠러 회전수의 차이를 활용하여 선수 요 방향 모멘트를 생성하고 방향을 조절하는 무인선박 모델이다.

2.1 ONRT 동역학 모델

첫 번째 조종모델로서 Kim et al. (2022)의 데이터 기반 모델링 기법을 활용하여 개발된 ONR Tumblehome의 조종모델을 사용하였다. ONR Tumblehome은 연구목적으로 공개된 미 해군 함형의 선형 모델로, 그 구체적인 제원은 Table 1에 기재되어 있다. 사용된 시뮬레이션 모델은 실제 크기의 1/48.9로 축소되었다. Fig. 1는 ONR Tumblhome의 선형이다.

ONR Tumblehome 모델은 방향타 제어 방식을 사용하여 선박의 선회를 제어한다. 조종모델에 사용된 선박 동역학 방정식은

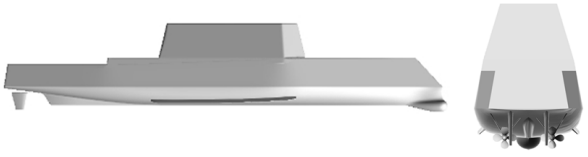


Fig. 1 Geometry of ONR Tumblehome

Table 1 Principal dimensions of ONR Tumblehome

Item		Model
Scale ratio	λ	1/48.9
Length (m)	L	3.147
Depth (m)	D	0.266
Beam (m)	B	0.384
Draft (m)	T	0.112
Displacement (kg)	△	72.6
Propeller diameter (m)	D	0.107
Advance speed (m/s)	U	1.11

Component 모델이며, 식 (1)-(4)와 같다. MMG 모델로 대표되는 Component 모델은 선박에 작용하는 외력을 선체, 추진기, 그리고 방향타가 받는 힘으로 분류하여 해석한다 (Matsumoto and Suemitsu, 1983). 방향타의 제어로 인해 선회하는 대상 선형의 특성상, 방향타의 유체동역학 모델링이 전체 조종에서 중요한 역할을 한다. MMG 식 (1)-(4)에서 첨자 H, P, R은 Hull, Propeller, Rudder를 나타낸다.

$$(m+m_x)\dot{u}-(m+m_y)vr = X_H+X_P+X_R \quad (1)$$

$$(m+m_y)\dot{v}+(m+m_x)ur = Y_H+Y_R \quad (2)$$

$$(I_{xx}+J_{xx})\dot{p} = K_H+K_R-mgGZ\phi+m_x z_H ur \quad (3)$$

$$(I_{zz}+J_{zz})\dot{r} = N_H+N_R \quad (4)$$

여기서 m은 질량을 I_{xx}, I_{xx}는 관성 모멘트를 의미하며 m_x, m_y는 선박의 부가질량을 J_{xx}, J_{zz}는 선박의 부가 관성 모멘트를 의미한다.

2.2 Otter 동역학 모델

두 번째 조종 모델은 Otter USV(Unmanned Surface Vehicle)를 사용하였다. Otter USV는 탐사 및 연구 작업에 활용되는 무인 선박이며, 이 모델은 4자유도의 동역학 모델을 사용한 ONR Tumblehome과 달리, Fossen (2011)에 의해 제시된 6 자유도 운동 방정식을 사용하였다. 이 6 자유도 시스템은 선박의 위치(x, y, z)와 자세(roll. pitch, yaw)를 모두 포함하므로, 선박의 모든 가능한 움직임을 고려할 수 있다 (Fig. 2 참조). Table 2에 대상 선형의 주요 재원을 나타내었다.

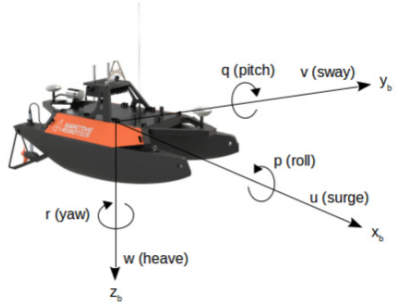


Fig. 2 Geometry of Otter USV

Table 2 Principal dimensions of Otter USV

Item		Model
Length (m)	*L*	2.0
Beam (m)	*B*	1.08
Height	*H*	0.82
mass (kg)	*m*	65
Maximum speed (m/s)	*U*	3.086

Otter USV 모델은 좌우 현 추진기의 회전수 차이를 조절하여 선박의 선수 각도를 제어한다. 본 연구에서 사용된 Otter USV 모델의 운동 모델은 다음과 같다.

$$M\dot{\nu}+C(\nu)\nu+D(\nu)\nu+g(\eta)+g_0 = \tau \quad (5)$$

식 (5)에서 M은 선박의 관성행렬, $C(\nu)$은 Coriolis 행렬, $D(\nu)$은 감쇠 행렬, $g(\eta)$은 중력과 부력 행렬, g_0은 정적 복원력 및 모멘트 행렬을 나타내며 ν, η는 각각 속도벡터, 위치벡터를 의미한다.

$$\nu = [u,v,w,p,q,r]^T, \eta = [x,y,z,\phi,\theta,\psi]^T \quad (6)$$

$$\tau = TKu \quad (7)$$

식 (7)에서 $\tau = [X,Y,Z,K,M,N]^T$는 제어력, T는 액추에이터 구성 Matrix이며, K는 추력 계수 값을 요소로 가지는 대각행렬, u는 제어 입력이다.

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} n_1 | n_1 | \\ n_2 | n_2 | \end{bmatrix}^T \quad (8)$$

제어 변수에 사용되는 n_i은 분당 프로펠러 회전수(revolution per minute rpm)이며, τ는 다음과 같이 구할 수 있다.

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -l_1 & -l_2 \end{bmatrix} \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (9)$$

3. 자율운항 시뮬레이션 시스템 구축

3.1 Unity 개발 환경

본 연구에서는 Unity 시뮬레이터를 사용하여 두 선박 모델의 움직임을 시뮬레이션하였다. Unity는 그래픽 렌더링과 물리 시뮬레이션을 지원하는 시뮬레이션 도구로, 복잡한 선박 동역학 해석 결과를 가시화하여 나타낼 수 있게 해준다. 이러한 시뮬레이션을 통해, 실제 선박 운동과 매우 가깝게 동역학 모델을 평가하고, 알고리즘의 성능을 측정할 수 있게 된다. Unity를 기반으로 선박과 그 주변의 물리 세계를 모델링하고 Python 기반의 ROS 시스템을 구축하였다. 구체적으로 ROS 노드를 작성하여 선박의 제어를 수행하고 제어가 수행된 후의 선박의 위치 좌표를 전달하는 노드를 통해 Unity에 전달하여 시뮬레이션을 수행하였다 (Fig. 3 참조).

Unity에서 ROS와 통신을 위해 Unity에서 제공하는 ROS Asset을 활용하였으며, Asset은 Unity와 ROS 간의 통신을 단순화하고 표준화하여 ROS 기반 시스템과 Unity 환경 사이에서 데이터를 쉽게 주고받을 수 있도록 돕는다. ROS Asset은 Unity에서 생성된 데이터를 ROS 메시지 형태로 변환하거나, ROS 메시지를

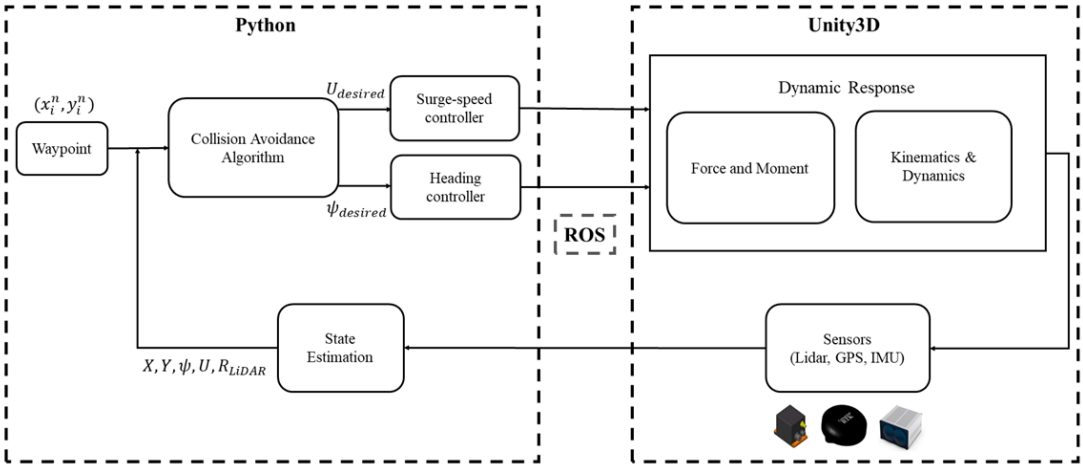
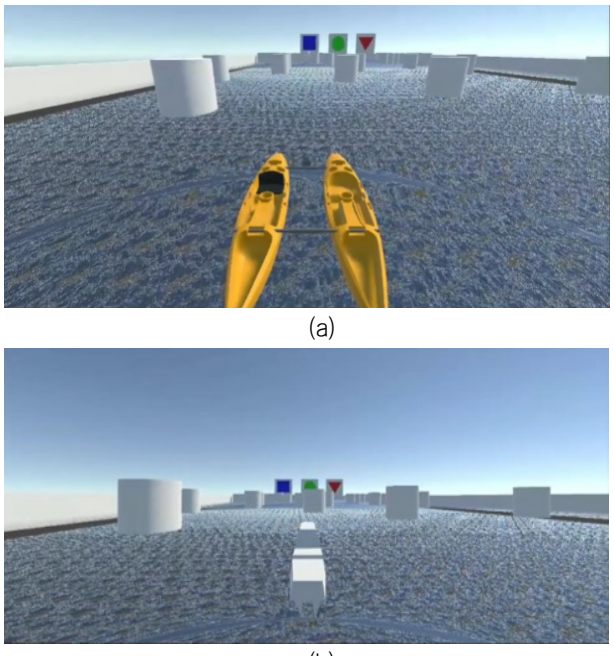


Fig. 3 Overview of simulation using Unity and ROS



(a)

(b)

Fig. 4 Ship test in the simulator environment of (a) Otter and (b) ONRT.

Unity에서 사용 할 수 있는 형태로 변환할 수 있다. 그러나 Unity에서 제공하는 표준 데이터 형식에 최적화되어 있어 특정한 데이터 형식이나 메시지를 사용하는 경우, 추가적인 작업이 필요하다.

본 논문에서는 ROS Asset과 선박의 외형을 고려하여 Fig. 4과 같이 선박의 동역학이 반영된 시뮬레이션 환경을 구성하였다.

3.2 가상 LiDAR 센서

Raycast는 Unity의 물리 함수 중 하나로, 특정 방향으로 가상의 광선을 발사하고 그 광선이 충돌하는 객체를 탐지하는 데 사용된다. 이 기능은 환경 인식, 인공지능, 사물 감지, 경로 탐색 등 다양한 애플리케이션에서 사용된다.

Raycast는 원점(광선이 시작하는 위치)과 방향, 그리고 선택적으로 레이저의 길이를 지정함으로써 작동한다. Unity는 이 정보를 바탕으로 가상의 레이저를 발사하고, 그 레이저가 충돌하는 객체를 탐지한다. 이때 레이저는 물리적인 영향은 미치지 않고, 객체의 탐지에만 사용된다.

충돌 회피 알고리즘의 핵심 요소 중 하나는 선박의 주변 환경을 정확하게 인식하는 것이다. 본 연구에서는 물체의 위치, 거리, 방향 등을 정확하게 감지할 수 있는 Raycast 기능을 활용하여 가상의 LiDAR 센서를 구현하였으며, LiDAR 센서의 데이터는 식 (10), (11)과 같이 설계하고 주변 환경을 관측 가능한 가상의 센서를 구현하였다.

$$M_{Lidar} = [\theta_i, r_i] \quad where, i \in [1, N] \tag{10}$$

$$\theta_0 = \frac{\theta_{fov}}{2}, \theta_i = \frac{\theta_{fov}}{2} - R_{\angle} i \quad where, i \in \left[\frac{\theta_{fov}}{R_{\angle}}\right] \tag{11}$$

θ_i는 LiDAR 센서의 레이저 i번째 각도, r_i는 LiDAR 센서로 탐지한 i번째 각도의 계측된 물체와의 거리, N은 LiDAR 센서가 측정 가능한 각도의 총 개수를 의미한다. θ_{fov}는 라이다가 측정 가능한 최대값으로 0~360도 사이의 값을 설정할 수 있게 하였다. R은 LiDAR 센서가 측정하는 각도의 간격으로 하여 다양한 관측 범위의 시뮬레이션이 가능하도록 가상의 LiDAR 센서를 구현하였다.

개발된 가상 LiDAR 센서는 선박이 안전하게 운항할 수 있도록 환경 인식 정보를 제공한다. 레이저를 발사하여 선박 주변 환경을 스캔하고 장애물에 충돌할 경우 이를 감지한다. 이렇게 감지된 정보는 충돌 회피 알고리즘에 입력되어 선박의 조향 결정에 활용된다.

3.3 ROS를 활용한 통신환경 구성

ROS은 로봇 개발에 필요한 다양한 라이브러리와 도구를 제공하는 유연한 프레임워크이다. 이를 통해 ROS 기반의 알고리즘

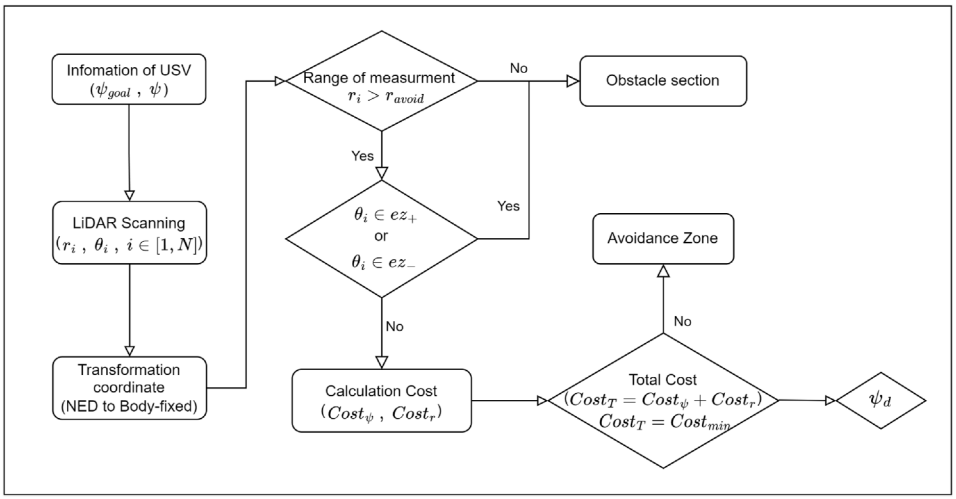


Fig. 5 Overview of Collision Avoidance.

을 Unity 환경에서 실행하고 테스트할 수 있다. 이는 Unity가 제공하는 다양한 가상 환경에서 실시간으로 센서 데이터를 전송하고 알고리즘의 결과를 반영하는 데 사용된다.

본 연구에서는 웹 소켓을 생성하여 Unity와 ROS 사이의 메시지를 전달하는 ROS Bridge 서버를 활용하였다. 경유점을 설정하고, Unity 환경에서 개발한 가상 LiDAR 센서의 데이터와 Unity에 구현된 선박 동역학 모델에 의해 실시간으로 계산되는 선박의 위치와 자세를 ROS에 전달하였다. Linux 환경에서 실행되는 python에서 ROS를 통해 데이터를 전달받은 뒤 이를 통해 가상 LiDAR의 데이터로 얻은 선박의 주변 환경에 대한 실시간 정보와 선박의 실시간 좌표 및 선수각을 충돌 회피 알고리즘에 적용하고, 알고리즘을 통한 명령 값을 ROS Bridge를 통해 다시 Unity에 전달하여 선박의 움직임을 제어하였다. 이러한 방식은 실제 무인 선박의 실험 환경과 비슷한 제어 방식을 사용하기에, 가상 환경에서 실제 선박 운동과 유사한 시뮬레이션을 수행할 수 있게 한다. 또한, 실제 실험하기 어려운 복잡한 환경 및 변수에 대응할 수 있는 알고리즘을 개발하는데 용이하다. (Fig. 3 참조)

3.4 충돌 회피 알고리즘

Unity에 구현한 가상 LiDAR 센서 데이터를 활용하여 장애물을 감지하고 회피하는 알고리즘을 구현하였다 (Fig. 5 참조). LiDAR에서 관측된 데이터인 i 번째로 측정된 각도에서의 거리(r_i), i 번째의 각도(θ_i)를 바탕으로 안전구역(safety zone)을 설정하기 위해 위험 거리(r_{aviod})를 설정하였다. LiDAR에서 i 번째로 측정된 거리(r_i)가 지정된 위험거리(r_{avoid})보다 크면, 장애물이 존재하지 않는 안전구역($Zone_{safe}$)으로 설정한다. 만약 측정된 거리(r_i)가 지정된 위험거리(r_{avoid})보다 작다면, 장애물이 존재하는 구역($Zone_{obstacle}$)으로 설정한다. 위험 거리(r_{avoid})는 선박에 따라 선회 반경이 다르므로 선박의 동역학적 특성을 고려하여 적절한 값을 설정해 줄 수 있다 (식 (12), (13) 참조). 식에서 $Zone_{safe}$는 장애물이 없는 구간을 의미하고, $Zone_{obstacle}$은 장애물이 있는 회피해야하는 구간을 의미한다. r_i는 LiDAR에서 i 번째로 계측된 거리이며, θ_i는 LiDAR에서 계측된 i 번째의 각도이고, r_{avoid}는 상수로 지정한 장애물이라 판단하는 거리이다.

$$Zone_{safe} = [(\theta_1, r_1) \cdots, (\theta_i, r_i)] \quad \text{if}, r_i > r_{avoid} \tag{12}$$

$$Zone_{obstacle} = [(\theta_1, r_1) \cdots, (\theta_i, r_i)] \quad \text{if}, r_i < r_{avoid} \tag{13}$$

안전구역을 설정해 주었다 하더라도, 선박이 장애물을 피해 선회하는 동안 선박의 옆 부분이 장애물에 부딪힐 위험이 있다. 따라서 선박의 폭을 고려하여 장애물이 존재하는 각도의 양의 방향과 음의 방향에 선박의 폭을 고려하여 영향을 줄 수 있는 추가적인 위험지역(ez_+, ez_-)을 설정해 주었다 (Fig. 6 참조).

최종적으로 안전구역이 결정되면, 안전구역에 저장된 LiDAR의 계측 거리(r_i) 및 각도(θ_i)와 목표하는 목표점과 현재 선박의 위치 사이의 각도(ψ_{goal})과 위험 거리정보를 사용하여 비용 함수가 계산되고, 비용함수가 최소가 되는 각도를 선택할 수 있도록 비용함수를 설계하였다. 각도의 경우 각각의 안전구역의 LiDAR의 거리 관측값이 가지는 각도와 목표점까지의 각도와의 차이(e_ψ)를 사용하여 비용을 계산한다. e_ψ가 커질수록 경유점으로부터 멀어짐을 의미하므로 비용함수가 증가하도록 적절한 값을 실험을 통해 설정하였다 (식 (14)-(15) 참조). 이를 통해 최종적으로 비용함수 값이 가장 작은 ψ_d값을 향하도록 선수각을 제어한다. 여기에 더하여 만약 목표 경유점까지의 각도인 ψ_{goal}가 안전구역 내에 존재한다면, ψ_{goal}로 향하도록 하는 비용 함수를 추가하여 목표점을 우선순위로 계산하도록 하여, 효율적으로 원하는 목표점을 추종하도록 알고리즘을 구현하였다 (식 (16) 참조). 식에서 C_ψ, C_r, C_T는 각각 선박의 헤딩 각도를 기반으로 계산된 가중치, LiDAR로 계측된 거리에 따라 계산된 가중치, 최종

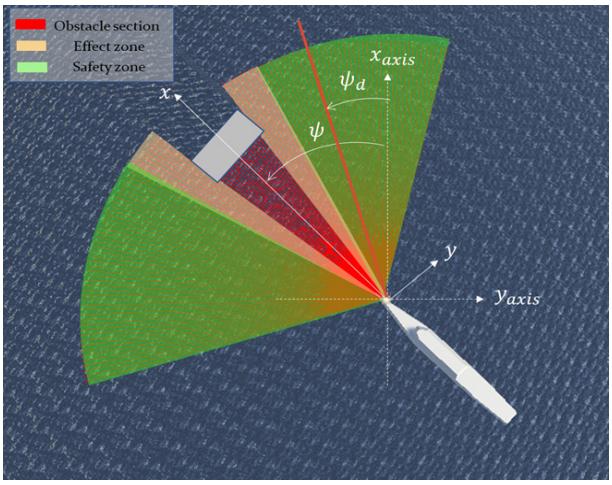


Fig. 6 Obstacle avoidance using LiDAR sensor

적으로 계산된 가중치로 정의된다. e_ψ는 선체 고정 좌표계에서 선박이 가야하는 목표점까지의 각도와 선박의 현재 헤딩의 차이로 정의할 수 있고 ψ는 현재 선박의 방위각, ψ_{goal}은 목표점 까지의 각도로 정의 할 수 있다.

$$C_\psi = \begin{Bmatrix} e_\psi & (e_\psi \le 10^\circ) \\ 10e_\psi & (e_\psi > 10^\circ) \end{Bmatrix} \tag{14}$$

$$C_r = \begin{Bmatrix} r_{avoid} - r_i, & where\ (r_i < r_{avoid}) \\ r_{avoid}\ , & else \end{Bmatrix} \tag{15}$$

$$C_T = \begin{cases} C_r(r_i) + C_\psi(\psi - \psi_{goal}) \\ C_r(r_i) + C_\psi(-\psi_{goal}) \end{cases} \quad (Zone_{sefty} \ni \psi_{goal}) \tag{16}$$

비용 함수에 의해 결정된 ψ_d와 현재 방위각인 ψ의 차이를 활용하여 오차(e)를 구하여 PID 제어를 적용하였다. 식 (18)에서 δ는 방향타이며, ONRT의 러더 동역학 모델을 통한 입력 값을, n은 Otter 모델의 프로펠러 회전수를 의미한다. k_p은 비례상수, k_p는 적분상수, k_p는 미분상수를 의미한다.

$$e = \psi_d - \psi \tag{17}$$

$$\delta, n = k_p e + k_i \int e dt + k_d \dot{e} \tag{18}$$

4. 시스템 검증

Unity와 ROS를 기반을 구축된 통합 시뮬레이션 환경을 대상으로 하여 제안된 충돌 회피를 포함한 자율운항을 위한 통합 알고리즘을 검증하였다. 두 선박에 대한 단일 경유점과 다양한 형태의 장애물을 가진 시나리오와 다수의 경유점과 복잡한 형태의 장애물이 복합된 환경을 가진 두 개의 시나리오를 계획하여 자율운항 시스템의 검증을 수행하였다.

또한, 3.4절에서 설명한 바와 같이 두 선박의 형상 및 동역학적 특성이 다르므로 충돌 회피 알고리즘에는 튜닝을 통해 적절한 파라미터를 각각 선정하여 사용하였다.

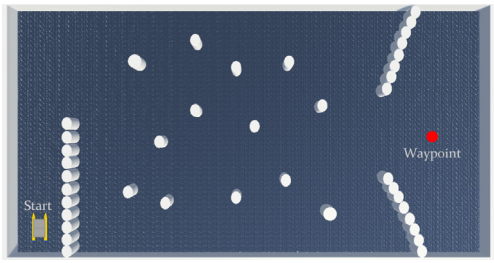
검증의 효율성을 높이기 위해 선박의 현재 상황과 과거 경로를 확인할 수 있는 GUI(Graphical User Interface)를 제작하여 검증을 수행하였다.

4.1 시나리오 계획

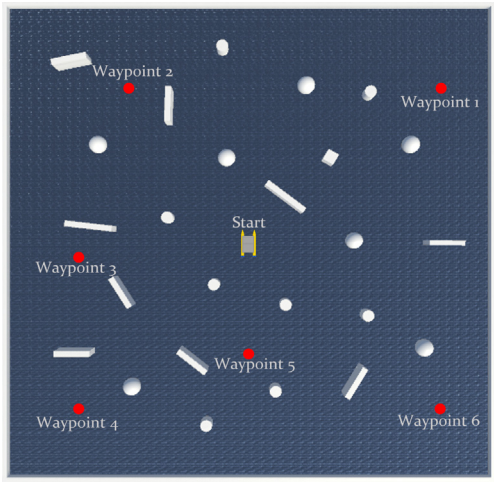
제안된 충돌 회피 알고리즘을 포함하는 자율운항시스템의 검증을 위해 Unity 기반의 3D 모델링 환경과 ROS 기반의 통신환경을 구축하고 통신을 통해 정보를 전송하여 실제 해양환경을 모사하였다. 충돌 회피와 경로 추종 성능을 확인하기 위해 두 가지 시나리오를 작성하고 이를 시뮬레이션 환경에서 구현하였다.

첫 번째 시나리오는 충돌 회피와 경로 추종 성능의 검증을 위해 하나의 목표점을 설정하고 그 과정에 다양한 형태의 장애물 환경을 두었다. 장애물을 회피하고 경로를 효과적으로 추종하는지에 중점을 두어 검증 시나리오를 구성하였다.

두 번째 시나리오는 다양한 장애물이 존재하는 상황에서 다수개의 목표점을 통과하는 경로를 추종하는 성능을 평가하였으며, 첫 번째 시나리오에 비해 넓은 환경에서 6개의 경유점과 최종 목표점을 추종하는 시뮬레이션 환경을 구성하고 자율운항 알고리즘 성능을 검증하였다 (Fig. 7 참조).



(a)



(b)

Fig. 7 The Scenarios setting for system verification (a) Scenario 1 and (b) Scenario 2.

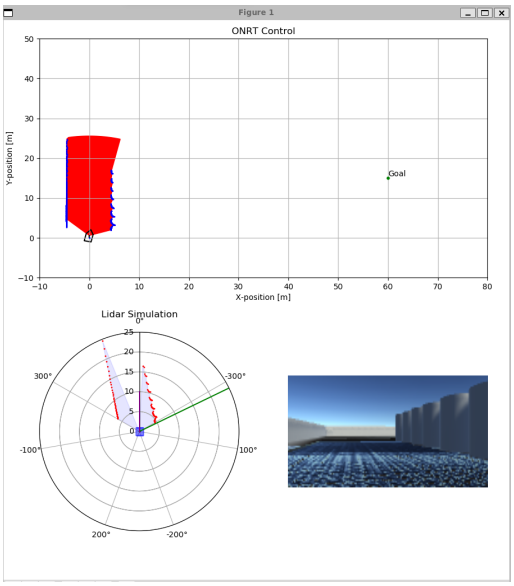


Fig. 8 GUI for simulation

4.2 시뮬레이션 성능 평가를 위한 GUI

검증과정을 운용자가 이해하여 시뮬레이션의 효용성을 높이기 위해, Unity 상에 LiDAR와 카메라를 구현하여 지도를 작성하고 다양한 관점에서 선박의 충돌 여부를 모니터링 가능한 환경을 구축하였다. Fig. 8과 같은 GUI를 구성하여 선박의 현재 상황을 실시간으로 모니터링할 수 있도록 하였고, 다양한 각도에서 선박을 촬영하는 카메라를 구현하여 선박의 충돌 여부를 관찰하였다.

GUI는 3개의 영역으로 구성된다. 첫 번째 영역은 경유점과 목표점을 표시하고 LiDAR를 통해 관측되는 거리정보를 중첩하여 지도를 작성한 결과를 시각화하였다. 작성된 지도에는 LiDAR의 관측범위, 선박의 과거 진행경로, 현재 선수 방향을 표기하여 충돌 회피와 경로 추종 여부를 시각적으로 확인할 수 있게 하였다. 두 번째 영역은 LiDAR를 통해 관측되는 장애물과 목표점의 정보 및 선수각의 방향을 극좌표계로 변환하여 표시함으로써 선박의 목표점에 대한 추종과 장애물에 대한 회피 판단 근거를 확인할 수 있게 하였다. 마지막 영역은 선박의 전면에 카메라를 설치하여 실시간으로 선박과 장애물과의 충돌 여부를 시각적으로 확인할 수 있도록 하였다.

4.3 성능 분석 및 결과

계획된 첫 번째 시나리오인 단일 목표점을 설정하고 다양한 형태의 장애물이 존재하는 상황을 가정한 상황에서 ONRT와 Otter모델의 시뮬레이션을 진행한 결과, 두 선박 모두 장애물에 충돌하지 않고 목표점에 도달한 것을 확인할 수 있었다. Fig. 9, 10의 (a)는 LiDAR 센서에서 관측된 거리정보를 기반으로 관측값을 중첩하여 Unity 상에서 작성된 환경의 지도를 그리고, 이를 기반으로 경로 추종과 충돌 회피를 수행한 선박의 경로이고, Fig. 9, 10 (b)는 각각의 시간에서 LiDAR에 의해 측정된 장애물들까

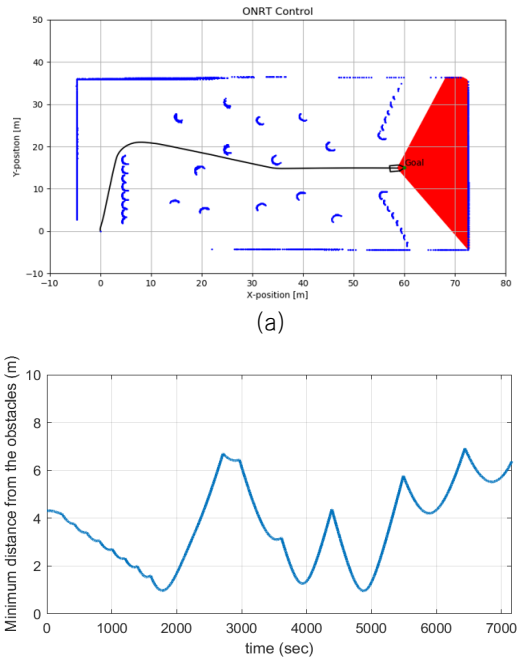


Fig. 9 The (a) trajectory and (b) minimum distance from obstacles of ONRT in Scenario 1.

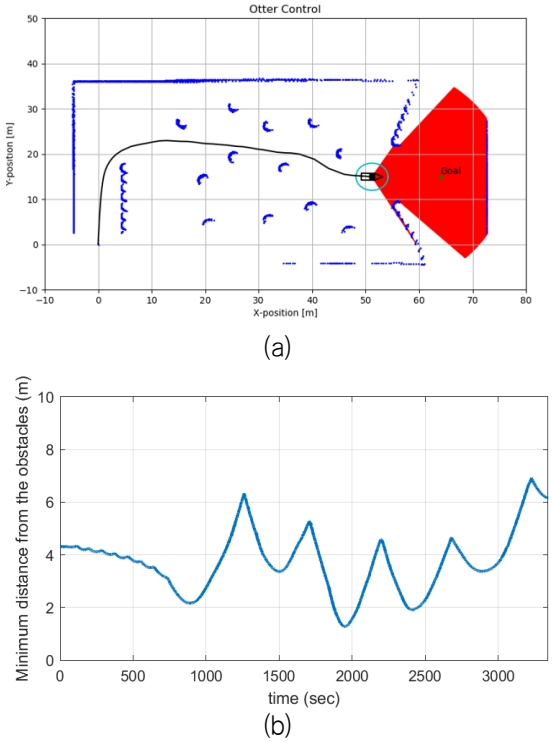


Fig. 10 The (a) trajectory and (b) minimum distance from obstacles of Otter in Scenario 1.

지의 거리 중 최소값을 나타낸 것이다. 방향타를 사용하여 조향을 수행하는 ONRT가 두 개의 모터의 출력차를 사용하여 조향하는 Otter에 비해 작은 움직임으로 장애물을 회피하는 섬세한 조

종 결과를 보였다. 더 자세한 비교를 위해 두 선박의 희망 선수각과 실제 선수각을 Fig. 11에 나타내었다. ONRT의 조종 내용에서 상대적으로 급격한 변화가 없이 완만한 변화를 보이는 희망 선수각과 이를 잘 추종하는 실제 선수각이 얻어지는데 반해, Otter의 경우 장애물과 조우시 희망 선수각이 급격하게 변화하며 실제 선수각이 변화하는 것을 확인 할 수 있다. 이러한 조종 특성의 차이로 인하여 ONRT의 경우 상대적으로 장애물에 근접한 경로로 회피하기 때문에 안전성은 낮지만 운항 효율은 높은 특성을 보인다. 반면 Otter의 경우 급격한 기동으로 인하여 에너지 효율이 낮아지지만, 장애물 회피에 관련한 부분에 한정하여 궤적 및 근접한 장애물까지의 최소 거리를 확인한 결과 ONRT에 비해 여유를 크게 갖는 안전한 반경으로 장애물을 회피하는 기동을 보이는 것을 확인할 수 있다.

두 번째 시나리오는 다수의 목표점을 설정하고 복합적인 장애물이 존재하는 환경에서의 운항 시뮬레이션이다. 복합적인 장애물이 존재하여 경로를 식별하기 복잡한 상황에서도 두 개의 선박 모두 충돌 없이 모든 경유점을 경유하여 목적지에 도착하는 것을 확인할 수 있었다. Fig. 12는 ONRT와 Otter모델이 시뮬레이션에서 운행한 궤적을 LiDAR로 관측된 데이터를 중첩하여 작성한 지도에 표시한 것이다. ONRT의 경우 방향타를 사용하여 조향이 Otter에 비해 쉬우므로 장애물 회피의 안전성보다 최소한의 기동으로 장애물을 회피하고 목표점에 도달하는 경향을 확인할 수 있고 Otter는 큰 선회각을 가지면서 장애물 주변을 큰 반경을 가지고 회피하는 기동을 보여주었다. Fig 13은 ONRT와 Otter모델이 시뮬레이션에서 장애물을 회피하는 과정에서 근접한 장애물까지의 최소 거리를 나타내고 있다. ONRT의 경우 조종을 위한 제어명령의 응답속도가 Otter에 비해 빠르기 때문에 장애물까지의 최소 거리의 변화 폭이 더 큰 것을 확인할 수 있다.

희망 선수각과 실제 선수각의 비교에서도 비슷한 경향은 있으나 시나리오 1에 비해 넓은 공간으로 인하여 Otter 모델이 시나

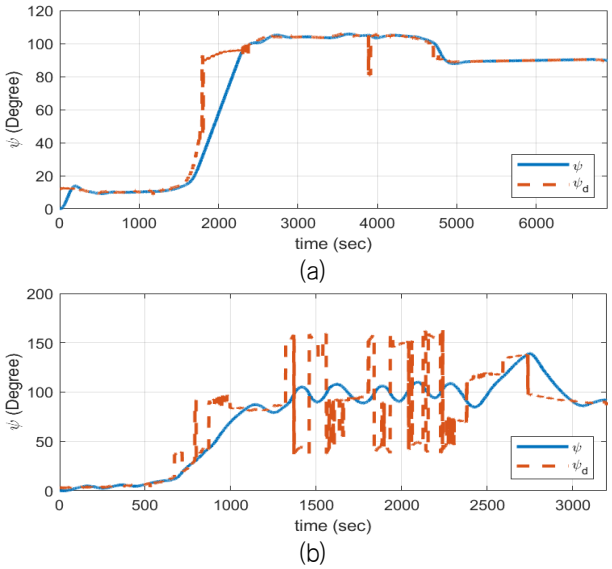


Fig. 11 Actual heading and desired heading of (a) ONRT and (b) Otter in Scenario 1.

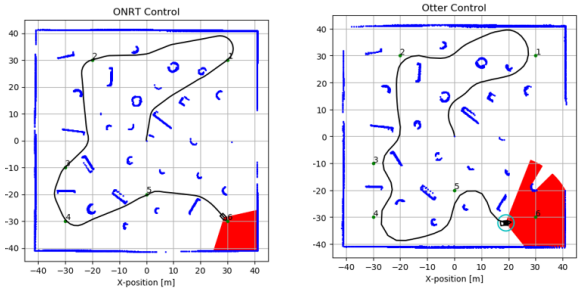


Fig. 12 The trajectory of (a) ONRT and (b) Otter in Scenario 2.

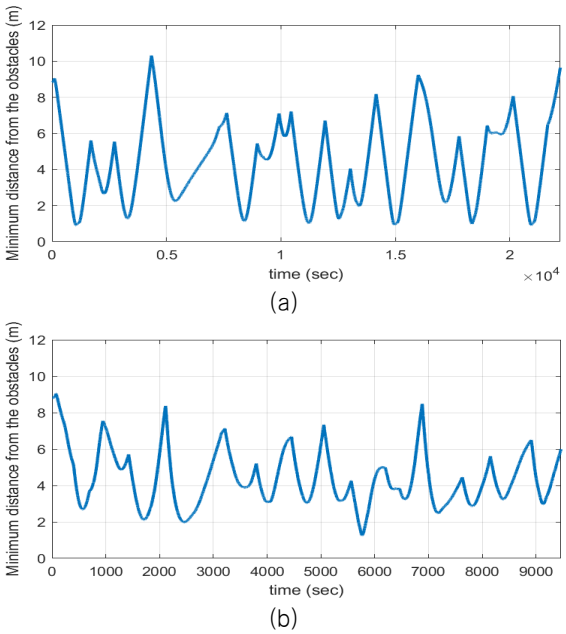


Fig. 13 The minimum distance from obstacles of (a) ONRT and (b) Otter in Scenario 2.

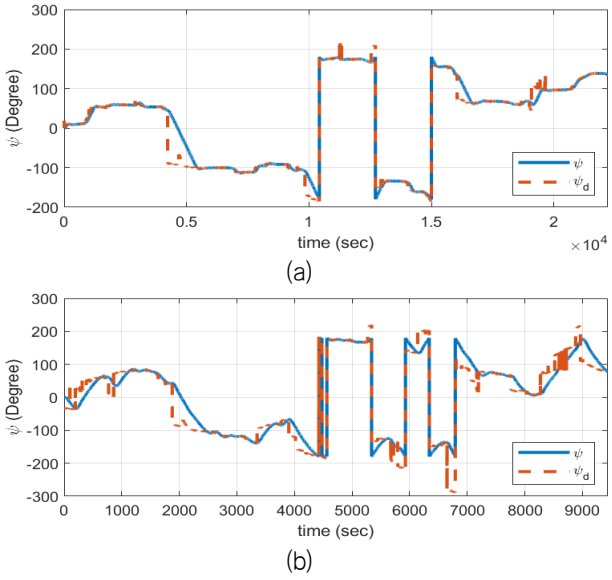


Fig. 14 Actual heading and desired heading of (a) ONRT and (b) Otter in Scenario 2.

리오 1에서 보이던 희망 선수각과 실제 선수각이 급격한 차이를 보이며 장애물을 회피하는 기동을 보이는 현상이 완화된 것을 확인할 수 있다(Fig. 14 참조).

5. 결 론

본 연구에서는 Unity 시뮬레이터 기반 선박 동역학 모델을 구성하고 ROS 통신을 활용하여 선박의 자율운항 알고리즘을 개발하고 검증하는 데 중점을 두었다. Unity에서 제공하는 물리 함수를 사용하여 개발한 가상 LiDAR와 제어 방식이 다른 두 가지의 선박 동역학 모델을 통해 실제 실험 환경과 유사한 모의 시뮬레이션을 수행하였으며 이를 통해 충돌 회피 알고리즘의 신뢰성을 확보하였다.

ONRT와 Otter의 시뮬레이션에서 볼 수 있듯이 선박의 조향 방식과 형태 등에 따라 선박의 기동 경향은 다양하게 발생할 수 있다. 그러나 실제 선박을 제작하여 이러한 기동방식을 실험하고 실용화할 수 있는 알고리즘을 구현하고 최적화하기 위해서는 많은 실험과 데이터의 수집이 필요하다. 본 논문에서는 이러한 실험의 효율성을 확보하고자 Unity를 사용하여 실제 선박의 기동환경을 고려한 시뮬레이션 환경을 개발하고 이를 사용하여 충돌 회피와 목표점 추종을 포함하는 자율운항 시스템을 검증하였다. 이를 통해 장애물까지의 최소 거리와 헤딩에 대한 정보를 기반으로 ONRT모델의 조종에 대한 제어 응답이 Otter모델에 비하여 빠르다는 것을 시뮬레이션을 통해 확인할 수 있었고 이러한 차이가 두 선박의 충돌회피 특성에 영향을 주는 것을 확인할 수 있었다.

Unity와 ROS를 연동한 시뮬레이터 기반의 자율운항 시스템에서 선박과 그 주변 환경을 구성하는 물리적 상황은 Unity 기반의 물리 모델링을 사용하여 구성할 수 있어 장애물의 형태를 다양화하거나, 다수의 장애물이 있는 환경, 파고가 높은 환경 등 다양한 환경을 구현하여 실험할 수 있다는 장점이 있다. 또한, 선박의 실제 제어에 대한 명령의 전달을 위해 ROS를 사용하여 향후 선박이 실제로 하드웨어에서 구현되는 과정과 동일한 데이터와 명령에 대한 전달 시스템을 구축하여 시뮬레이션 이후 실제 현장 시험의 적용을 위한 시스템 수정 과정을 효율화할 수 있다는 장점이 있다.

향후 연구에서는 Unity에서 자율운항 알고리즘의 검증을 위해 사용된 카메라를 추가하여 주변 환경을 인식할 수 있는 알고리즘에 대한 개발을 진행하여 다양한 환경에도 적용 가능한 자율운항 알고리즘의 개발을 수행하고 딥러닝이나 영상처리, 강화학습 등의 기법을 적용하여 종합적인 자율운항 시스템의 구축을 수행하고자 한다.

후 기

이 논문은 2023년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (P0017006, 2023년 산업혁신인재성장지원사업)

References

Cho, Y.H., Kim, J.H. and Kim, J.W., 2021. Automatic ship collision avoidance in narrow channels through curvilinear coordinate transformation. *Journal of the Society of Naval Architects of Korea,* 58(3), pp.191-197.

Fossen, Thor I. *Handbook of marine craft hydrodynamics and motion control*. John Wiley and Sons, 2011.

Hong, S.K., 2015. Analysis of factors influencing ship collision avoidance judgment of maritime officers. *Journal of the Korean Institute of Plant Engineering*, 20(1), 3-10.

Hu, Y. and Meng, W. 2016. ROSUnitySim: development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning. *Simulation : Transactions of the Society for Modeling and Simulation International*, 92(10), pp.931-944.

Hussein, A., Garc´ıa. F. and Olaverri-Monreal, C., 2018. ROS and unity based framework for intelligent vehicles control and simulation. In ICVES *International Conference on Vehicular Electronics and Safety*, Spain, Madrid, 12-14, September, 2018.

Jo, H.J., Kim, J.H., Kim, S.R., Woo, J.H. and Park, J.Y., 2021. Development of autonomous algorithm for boat using robot operating system. *Journal of the Society of Naval Architects of Korea*, 58(2), pp.121-128.

Jo, Y.S., Jung, H.J., Lee, S.H. and Jeong, D.W., 2022. LiDAR sensor and pure pursuit algorithm-based autonomous navigation system for small ships. *Journal of Korean Institute of Information Technology*, 20(4), pp.1-11.

Kim, K.W., Kim, H.Y., Choi, S.E., Na, K.I., Lee, H. and Seo, J.W., 2022. Development of ship dynamics model by free-running model tests and regression. *Journal of the Society of Naval Architects of Korea*, 59(3), pp.173-182.

Lee, E.-J., Ruy, W.-S. and Seo, J., 2020. Application of reinforcement learning to fire suppression system of an autonomous ship in irregular waves. *International Journal of Naval Architecture and Ocean Engineering*, 12, pp.910-917.

Matsumoto, N. and Suemitsu, K., 1983, Interference effects between hull, propeller and rudder of a hydrodynamic mathematical model in maneuvering motion. *Journal of Kansai Society of Naval Architects*, 93, pp.45-62.

Shin, D.S., Park, B.C., Lim, C.O., Oh, S.J., Kim, G.Y. and Shin, S.C., 2020. Pipe routing using reinforcement learning on initial design stage. *Journal of the Society of Naval Architects of Korea*, 57(4), pp.191-197.

Son, D.G. and Kim, D.H., 2021. Autonomous navigation of unmanned surface vehicle based on fuzzy control considering COLREG. *Journal of Korean Institute of Intelligent Systems*, 31(1), pp.11-20.

Yang, W.J. and Ko, J.Y., 2004. A study on the risk control measures of ship's collision. *Journal of the Society of Naval Architects of Korea*, 41(3), pp.41-48.

김 기 원 | 방 현 태 | 서 정 화 | 윤 원 근