

논문 2025-3-10 <http://dx.doi.org/10.29056/jsav.2025.09.10>

4족 보행 로봇에서 WebRTC와 ROS2 기반 자율 순찰 및 위험 탐지 모듈 구현

황동하*, 김재웅*†, 김동현**

Implementation of an Autonomous Patrol and Risk Detection Module based on WebRTC and ROS2 for a Quadruped Robot

Dong-Ha Hwang*, Jae-Woong Kim*†, Dong-Hyun Kim**

요 약

본 논문은 4족 보행 로봇 Unitree GO2를 활용하여 복잡하고 비정형적인 환경에서 통합 운용이 가능한 차세대 지능형 자율 순찰 시스템을 구현하였다. 이는 기존의 단편적인 모듈 구현이 아닌 통합 시스템으로서 첫째, WebRTC와 ROS2를 통합하여 200ms 이내의 저지연 영상 스트리밍과 원격 제어를 가능하게 하였으며 둘째, ROS2 기반 SLAM과 Nav2를 이용하여 능동적으로 지정된 순찰 경로를 따라 이동하며 장애물을 회피하여 임무를 완수하도록 하였다. 셋째, 순찰시 위험(화재, 연기, 쓰러진 사람 등)을 실시간으로 탐지하기 위해, 객체 탐지 모델을 탑재하였으며, 로봇의 위치 보조와 출입자의 인식표에 있는 ArUco marker를 인식하도록 하였다. 넷째, 원격 관제 대시보드 기능과 Discord API를 이용하여 이벤트 발생 및 로봇 상태 조회 등의 메시지 서비스를 구축하여 보안-안전 시나리오에 필요한 엔드-투-엔드 파이프라인을 보유했고 있는 것이 특징이다.

Abstract

This paper implemented a next-generation intelligent autonomous patrol system that can be integrated in complex and unstructured environments using the four-legged walking robot Unitree GO2. This is not an existing fragmentary module implementation, but an integrated system. First, WebRTC and ROS2 were integrated to enable low-latency video streaming and remote control within 200 ms. Second, ROS2-based SLAM and Nav2 were used to actively move along designated patrol routes and avoid obstacles to complete tasks. Third, an object detection model was installed to detect threats (fire, smoke, fallen people, etc.) in real-time during patrol, and the robot's location assistance and ArUco marker on the visitor's recognition table were recognized. Fourth, it is characterized by the construction of message services such as event occurrence and robot status inquiry using the remote control dashboard function and the Discord API to have the end-to-end functions required for security-safety scenarios.

한글키워드 : 4족보행로봇, SLAM, Lidar, YOLO, WebRTC, Discord

keywords : Four-legged Robot, SLAM, Lidar, YOLO, WebRTC, Discord

* 공주대학교 컴퓨터공학과

접수일자: 2025.08.29. 심사완료: 2025.09.12.

** 나사렛대학교 IT인공지능학부

게재확정: 2025.09.20.

† 교신저자: 김재웅(email: jykim@kongju.ac.kr)

1. 서론

1.1 연구 배경 및 필요성

고도화되어 가고 있는 현대 사회에서는 안전 및 보안에 대한 요구는 날로 증대되고 있지만, 인력 기반의 전통적인 순찰과 고정식 CCTV는 고비용과 감시 인력의 오류, 사각지대 발생과 같은 한계가 있다[1]. 이러한 문제들을 해결하기 위해 로봇 기반의 자율 순찰 로봇 시스템이 부상하고 있다. 자율 순찰 로봇은 시스템에서 정한 구역을 24시간 순찰하고 보안 공백을 최소화하며, 이상 상황 발생 시 사람을 대신하여 능동적으로 대응할 수 있는 잠재력을 가진다[2,3]. 특히 계단과 같은 비정형 지형과 같은 환경에서 탁월한 기동성을 보이는 4족 보행 로봇은 기존 바퀴형 로봇의 이동 한계를[4] 극복할 수 있어 차세대 순찰 플랫폼으로 주목받고 있다[5]. 기존 연구들은 바퀴형 이동체 위에서 ROS1/ROS2와 RTSP/RTMP 계열 스트리밍을 결합한 경우가 대부분이며 이는 비정형 지형에서 이동성의 제한과 원격 관제에서 수백 밀리초 이하의 중단 간 지연을 지속 보장하려는 개선의 여지를 가지고 있었다[6]. 본 논문에서는 이러한 한계를 극복하고자 Unitree.com 사의 최신 4족 보행 로봇인 GO2 EDU를 플랫폼을 기반으로 한 비정형 지형에 특화된 지능형 자율 순찰 및 위험 탐지 모듈을 개발하는 것을 목표로 한다. 이를 위해 로봇의 이동 능력과 주변 환경에 대한 인지능력을 극대화하여 다양한 환경에서 자율 순찰이 가능하도록 시스템을 구축한다. 핵심적으로 상호작용과 원격 관제 상황에서의 낮은 지연시간을 개선하기 위한 WebRTC와 ROS2를 통합, 자율 순찰 파이프라인 구축, 순찰 중 발생하는 이상 감지에 대해서 상황을 스스로 인식하고 위험 요소를 식별하여 신속하게 대응하는 시스템을 통해 향후 로봇의 다양한 활용성을 찾고자 한다.

1.2 연구 목표 및 내용

본 논문에서는 Unitree사의 GO2 EDU 로봇을 플랫폼으로 사용하였으며 첫 번째 연구 목표로 WebRTC와 ROS2를 통합하여 로봇-관제 간 통신의 핵심으로 적용하여 200ms 이내 저지연의 실시간 스트리밍과 양방향 제어를 결합한 원격 관제 시스템을 제안하고자 한다. 이는 물리적 로봇에 대한 접근 방식 중 기존 방식 대비 개선된 최소의 지연시간을 갖게 하며, High Level 제어가 가능하게 하여 펌웨어 및 Low Level 제어에 대한 부담을 대폭 줄일 수 있게 하고자 한다. 다음의 표 1은 이러한 플랫폼이 되는 Unitree Go2 Edu 로봇의 사양을 정리한 것이다.

표 1. Unitree Go2 Edu 로봇 사양
Table 1. Unitree Go2 Edu robot Specification

Item	Specifications
Physical	70x30x40Cm Weight 15 kg 최대 보행속도 3.7 m/Sec Payload 8 ~ 12kg
Joint 및 동작 각도	12 Dof Max Torque 45N.m Body : -48° ~ 48° Thigh : -200° ~ 90° Shank : -156° ~ 48°
Computing Module	NVIDIA Jetson Orin (optional) (40~100Topscomputing power)
System	Battery 15000mAh(2~4h) 4D LiDAR : 360~90° Ultra wide angle scanning allows RGB카메라: 1280*720p, 30fps Depth 카메라 : Intel RealSense 발바닥압력센서 : 4점/각발

두 번째로 구현하려고 하는 자율 순찰 파이프라인은 SLAM-Nav2로 구현되며, 이는 Unitree Go2의 4D Lidar를 이용하여 주변 환경을 스캔하고 수집된 3D 점들의 집합(PointCloud2) 형태로

주변 환경의 형태와 구조를 입체적으로 맵핑하고, 이를 SLAM 패키지를 통해 3D 지도를 작성한다. 로봇이 그 지도 안에서 로봇의 위치를 정확히 추정하도록 하고 작성된 지도를 기반으로 Nav2 자율주행을 구현하는 것을 목표로 한다. 세 번째로 AI 기반 이상 감지 구현은 자율 순찰 시스템에서 카메라로부터 수집된 영상 데이터에 딥러닝 기반의 객체 탐지 모델을 적용하여 구현되는 기능으로, coco_dataset으로 학습된 기본 모델과 커스텀 데이터 세트로 학습한 특정 모델을 함께 사용하였다. 이는 순찰 중 미리 정의된 이상 조건 예를 들어 제한 구역 내에 'person' 객체가 감지되거나 학습된 모델에서 특정하는 쓰러진 사람, 연기와 불꽃과 같은 이상 현상과 출입이 허용된 사람의 신원 확인과 로봇의 순찰 경로를 보조하는 ArUco Marker의 인식을 포함하고 있다. 이렇게 감지된 상황은 카메라 스트리밍과 웹 기반 로봇 제어, Discord를 통한 이벤트 메시징 서비스까지 구현한 엔드-투-엔드 워크플로우가 완성된다[7]. 이러한 결과는 4족 보행 로봇이라는 플랫폼 특성과 ROS2 - WebRTC - YOLOv11 - ArUco - Discord의 통합에 의해 달성한 것으로, 웹 애플리케이션으로 구현되어 AWS를 통해 배포하여, 통합 관제 시스템 개발을 위한 기본 구조를 구축하였다. 이는 기존 연구의 단편적 모듈 구현을 넘어 현장에 적용할 수 있는 시스템 수준의 실증 차원의 연구이다.

2. 관련 연구

2.1 원격 제어를 위한 통신 및 프레임워크

로봇의 원격 제어와 관련해서 지연시간의 문제는 매우 중요한 부분이다. 그동안 주로 사용했던 기술들인 RTSP, HLS 등은 수 초의 지연시간이 발생하게 되어 실시간 상호작용에 부적합하다. 본 논문에서 사용하고 있는 WebRTC (Web

Real-Time Communication)는 Web Browser와 Native Application에서 추가로 설치해야 되는 플러그인 없이 매우 짧은 지연시간의 데이터를 송수신할 수 있게 하는 오픈소스 기술이다. 또한, 클라이언트간 직접 연결을 가능하게 하는 P2P(Peer-to-Peer) 연결을 통해 수백 밀리초 이내의 초저지연 통신을 가능하게 한다[8,9]. 이는 원격지의 관리자가 로봇의 시점에서 상황을 즉각적으로 파악하고 정밀하게 제어할 수 있게 하는 핵심 기술이다. 이러한 복잡한 로봇 시스템의 각 소프트웨어 모듈(SLAM, 인식, 제어)을 개발하고 통합하기 위해서는 표준화된 프레임워크가 필요하다. ROS(Robot Operating System)는 로봇 응용 프로그램 개발을 위한 사실상의 표준이며, 본 논문에서는 실시간성, 보안, 다중 로봇 지원이 강화된 ROS2[10]를 시스템의 미들웨어로 채택하여 모듈성과 확장성을 확보하였다.

2.2 자율 주행을 위한 SLAM과 Nav2

로봇의 자율주행에 있어서 LiDAR를 통해 점들의 집합정보를 수집하고 수집된 정보는 SLAM (Simultaneous Localization and Mapping) 패키지를 통해 3D 맵핑 과정을 거친다. 이렇게 작성된 3D 지도는 로봇의 자율주행을 위한 위치를 제공하는 역할을 한다. Nav2는 이러한 SLAM기반의 맵에서 Unitree Go2 로봇이 위치한 환경을 이해하고, 자율적으로 움직이는데 이용되는 로봇 내비게이션 핵심 기술이다. Nav2는 ROS2 환경에서 로봇의 자율주행을 위한 포괄적인 프레임워크를 제공한다. 이는 로봇의 위치를 추정하고, 목적지까지의 경로를 계획하며, 로봇의 이동 명령을 실행한다. Nav2의 특징 중에 행동 트리(Behavior Tree)는 목표에 도달하기 위해 어떤 순서로 행동을 해야 하는지 시각적으로 쉽게 설계하고 디버깅 할 수 있게 하며, 비동기 분산처리가 가능한 특징을 가지고 있다. SLAM의 초창

기 연구는 확장 칼만 필터(EKF)와 같은 확률론적 접근법에 집중하였으나, 계산 복잡도와 비선형성 문제로 인해 대규모 환경에서는 한계를 보였다. 이후, SLAM 문제를 비선형 최적화 문제로 정의하는 그래프 기반(Graph-based) SLAM이 주류로 자리 잡았다[11]. 이 방법은 로봇의 자세(pose)를 노드(node)로, 센서 측정값을 두 노드 간의 제약(constraint) 또는 엣지(edge)로 표현하는 팩터 그래프(Factor Graph)[12]를 구성하고, 전체 그래프의 오차를 줄여 최소화함으로써 일관성 있는 지도와 궤적을 추정하였다. 구글의 카토그래퍼(Cartographer)[13]는 LiDAR의 수집 데이터를 활용한 대표적인 자율주행 2D/3D Graph-SLAM 시스템이다. 4족 보행 로봇의 SLAM을 구현하기 위해서 단일 센서의 한계를 극복하려는 다중 센서 융합 연구가 활발히 진행되고 있다. 카메라와 IMU를 활용한 VIO(Visual-Inertial Odometry) 기술 중에서 VINS-Mono[14]는 시각 정보와 관성 정보를 상호 보완하여 자율주행에 대한 신뢰도를 높였다. 최근의 기술동향은 LiDAR, IMU, 카메라를 모두 융합하고 있다[15]. 최근 로봇의 고유한 신체 정보를 활용하는 방향으로 연구가 발전하고 있다. 이는 다리 관절의 운동학을 통해 로봇의 움직임을 추정하는 Leg Odometry를 SLAM 프레임워크에 통합하고자 하는 것이다. FAST-LIO2[16]는 LiDAR와 IMU를 높은 결합성(tightly-coupled) 방식으로 융합하여 높은 성능을 보이고 있는 대표적인 연구로 두각을 나타내고 있다.

2.3 이상 감지를 위한 딥러닝 기술

딥러닝, 특히 합성곱 신경망(CNN)의 발전은 객체 탐지 분야에 혁신을 가져왔다. 로봇 분야에서는 실시간 처리가 중요한 이슈이며, 이로 인해 YOLO(You Only Look Once)[17,18]와 같은 추

론 속도가 빠른 1-stage detector를 선호하고 있다. YOLO는 속도와 정확도가 균형을 맞추며 지속적으로 발전해왔고, 본 연구에서 사용하고 있는 YOLOv7[19]은 당시 실시간 객체 탐지 분야에서 최고의 성능을 보인 모델이었으며, 현재는 YOLOv11까지 발표되어 있다. 이러한 고성능 모델을 로봇의 제한된 자원에 탑재하기 위해서는 경량화 과정이 많은 제약이 따를 수밖에 없다. 이를 위해 MobileNets[20]와 같이 효율적인 네트워크 아키텍처를 설계하는 연구와 더불어, 학습된 모델을 양자화(quantization)하거나 하드웨어에 최적화하는 후처리 기술이 중요하다.

2.4 ArUco Maker

로봇 내비게이션에서 정확한 위치 추정을 위해 마커를 이용한 위치 보정 기술은 일반적으로 사용되고 있는 자율 주행의 보조 기술이다. Aruco 마커는 로봇 분야에서 카메라를 통해 3차원 위치 및 자세(pose)를 정확하게 추정할 수 있는 이점이 있어 활용도가 높다. ArUco 마커가 기존의 마커 시스템에 비해 높은 신뢰성과 감지 정확도를 제공하고 마커 간 혼동을 최소화하도록 설계되었다[21]. Aruco 마커가 제조, 물류, 품질 관리 등 다양한 산업 분야에서 로봇의 정밀한 위치 파악에 기여 하여 생산 효율성을 높인다고 강조하였다[22]. 이러한 연구들은 ArUco 마커가 로봇의 안정적인 내비게이션 시스템 구축에 효과적인 도구임을 입증한다.

3. 시스템 설계 및 구현

본 장에서는 연구 목표를 달성하기 위해 4족 보행 로봇 기반 자율 순찰 시스템 구현을 위한 구체적인 설계 내용과 구현 방법을 상세히 기술한다. 전체 시스템은 ROS 2(Robot Operating System 2) 프레임워크를 기반으로 각 기능 모듈

이 유기적으로 연동되도록 설계되었다. 본 장의 구성은 세 가지 핵심 연구 목표를 순차적으로 구현한다. 첫째 WebRTC 통신 프로토콜 연결의 구현하고 둘째 Unitree Go2로봇의 ROS2 통합으로 객체 탐지와 Discord를 통한 메시징 서비스를 구현하고 셋째 WebRTC 기반 원격 관제 인터페이스 구축을 구축하여 통합 시스템 성능 검증을 할 수 있도록 한다.

3.1 시스템 아키텍처

다음의 그림 1은 총 4개 계층으로 구성된 시스템을 나타낸 것이다.

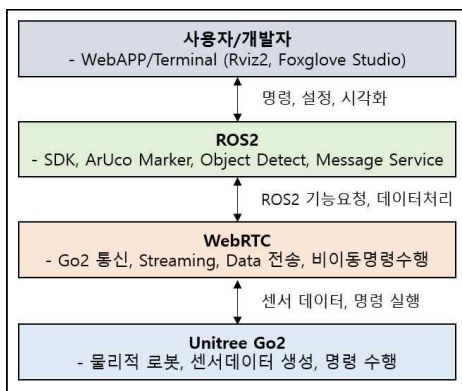


그림 1. 시스템 계층 구조도

Fig. 1. System Layer Architecture Diagram

첫째, 사용자/개발자 계층은 사용자가 로봇을 조작하거나 개발자가 명령을 내리는 인터페이스이다. 이는 사용자를 위해 웹앱으로 구현하였으며, 개발자는 터미널과 프로그래밍 툴을 이용하여 로봇을 제어할 수 있다. 또한, ROS2에서 전달되는 영상, 음향, 센서 데이터를 시각화하며 로봇에게 명령을 전달하고, 로봇의 상태를 모니터링하며 각종 설정을 변경할 수 있다. 둘째, ROS2 계층은 사용자 계층 바로 아래 위치하고 있는 로봇의 핵심 두뇌 역할을 한다. 이는 분산 처리 환경을 제공하는 프레임워크로 본 논문에서는 Go2 SDK연동과 ArUco 마커를 이용한 출입자 판별,

Object Detection을 이용한 화재 감지, 이러한 내용을 Discord를 이용하여 Message Service를 구현하였다. 셋째, WebRTC계층으로 ROS2와 Unitree Go2 로봇을 연결하는 중간 역할을 한다. 이는 실질적으로 웹브라우저 간의 실시간 통신을 하는 기술이나, 여기서는 로봇과 제어 시스템 사이의 실시간 통신을 위해서 사용되었다. 여기서 주요 역할은 Go2와의 직접적인 통신을 담당하고, 로봇의 센서데이터 중 영상 데이터와 라이다 데이터를 스트리밍한다. 또한, 제어 명령이나 기타 데이터를 양방향으로 전송한다. 마지막으로 ROS2로부터 받은 이동 명령을 실제로 로봇에게 전달하는 역할을 담당한다. 넷째, Unitree Go2 계층은 가장 아래에 있는 물리적인 로봇 자체이다. 이 계층에서 로봇은 실제로 움직이고 주변 환경을 감지한다. 이러한 계층을 통해 ROS2를 활용해 고수준의 복잡한 로봇 제어 로직을 구현할 수 있고, SLAM과 Nav2 기반의 자율 순찰 로봇 시스템을 구현하였다.

3.2 시스템 프로세스

다음의 그림 2는 시스템 전체의 프로세스를 나타낸 것이다.

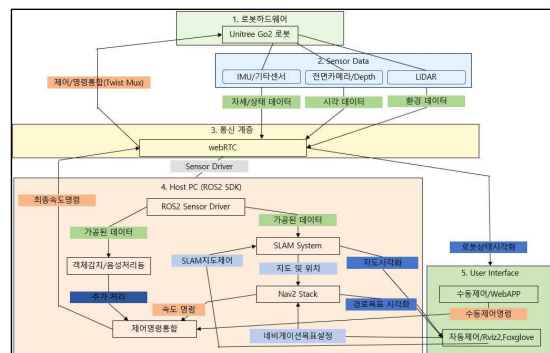


그림 2. 시스템 프로세스

Fig. 2. System Process Diagram

로봇 하드웨어와 User Interface 간의 데이터와 제어 흐름을 보여주고 있으며, 이 시스템은

자율 순찰과 원격 개입을 동시에 만족하는 구조로 설계되었으며 통신 계층은 WebRTC를 이용하여 200ms 이하의 최적의 지연시간으로 제어 및 개입에 있어 이질감을 최소화하였다. Ubuntu 22.04 OS 위에 ROS2를 적용하여 사물 인식, 센서 데이터의 수집, SLAM 데이터의 수집 등 다양한 Node를 수용할 수 있게 하였다. 각종 데이터와 처리 과정을 통해 도출된 자동 제어 명령과 원격 개입에 의한 수동 제어 명령이 User Interface 계층에서 함께 처리될 수 있도록 하여 로봇 제어의 일관성을 갖고 있으며, ROS2는 이러한 명령을 통합하여 통신 계층을 거쳐 로봇 하드웨어에 전달하도록 하였다. 센서 데이터는 전면 카메라와 LiDAR 정보를 즉시 전달하여 ROS2 node에 있는 객체 감지/음성처리 모듈에서 처리하도록 하였으며, SLAM 모듈에서 지도 만들어진 Map 정보를 Nav2 Stack에 전달하였고, User Interface 계층에서 구성한 순찰 경로에 따라 주행할 수 있도록 구현하였다.

3.3 WebRTC 구현

다음의 그림 3은 WebRTC를 구현하기 위한 단계별 과정을 나타낸 것이다.

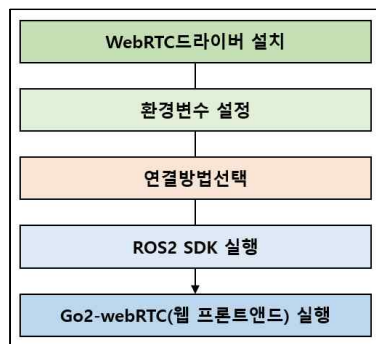


그림 3. WebRTC 구현 프로세스
Fig. 3. WebRTC Implementation Process

다음의 그림 4는 WebRTC 드라이버를 단독으로 사용하기 위한 설치 명령어이다.

```

cd ~
sudo apt update
sudo apt install python3-pip
sudo apt install portaudio19-dev
git clone --recurse-submodules https://github.com/legion1581/go2_webrtc_connect.git
cd go2_webrtc_connect
pip install -e .
  
```

그림 4. WebRTC 드라이버 설치
Fig. 4. Driver installation

이 드라이버를 통하여 Unitree Go2 로봇에 연결하기 위한 WebRTC Python 개발 환경의 구축이 가능하다. ROS2 SDK를 실행하기 전에 연결 유형을 WebRTC로 명시하고 로봇의 IP 주소를 함께 설정하는 과정이다. WebRTC는 다음과 같이 3가지 연결 모드를 지원한다. 첫째, AP 모드로서 Go2가 AP 모드 상태이고 WebRTC 클라이언트가 로봇에 직접 연결되는 방식으로 개발 단계에서는 사용하게 되지만 실제 서비스에서는 다른 방식을 사용하여야 한다. 둘째, STA-L 모드로서 Local STA Mode를 의미하고, Go2와 WebRTC 클라이언트가 동일한 로컬 네트워크에 있을 때 IP 주소 또는 일련번호를 사용하여 연결하게 된다. 셋째, 가장 많이 사용하게 되는 STA-T(Remote STA Mode)로 Unitree Turn Server를 통한 원격 연결로 다른 네트워크에 있더라도 Go2를 제어할 수 있다. 이때는 Unitree 계정의 사용자 이름과 비밀번호가 필요하다.

3.4 SLAM-Nav2기반 자율 주행 시스템

다음의 그림 5는 ROS2에 몇 층의 레이어로 구현된 SLAM-Nav2 자율주행 시스템의 프로세스이다. 센서레이어는 LiDAR를 이용해서 3D 포인트 클라우드 데이터를 수집하며, 카메라를 이용해서 시각적 정보 및 객체 인식을 담당하도록 하고, IMU는 관성측정 데이터를 수집하고 차량 움직임을 추적하는 오도메트리로 구성되어 있다. SLAM 영역은 지도 생성 모듈로 센서 데이터를 기반으로 환경 지도를 작성하고 위치 추정 모듈로 차량의 현재 위치를 실시간으로 추정할 수 있게 한다. 인식/의사 결정 레이어는 센서 융합을

통한 환경이해를 담당하는 인식 시스템과 주행 전략 및 행동 결정을 담당하는 의사 결정 모듈로 구성되어 있다.

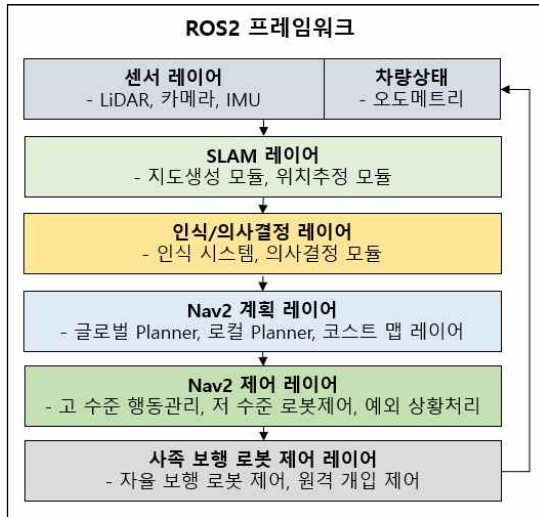


그림 5. SLAM-Nav2 기반 자율주행 프로세스
Fig. 5. SLAM-Nav2 Based Autonomous Driving System Process Diagram

Nav2 계획 레이어는 전체 경로계획과 실시간 지역 경로 생성을 담당하는 플래너와 장애물 및 코스트를 계산하는 코스트맵 레이어로 구성되었으며, Nav2 제어 레이어는 고수준 행동 관리와 저수준 차량 제어 및 복구 행동을 담당하는 예외 상황 처리로 구성되어 있다. 마지막으로 차량 제어 레이어는 자율 주행과 원격 개입에 따른 수동 제어를 통합하는 차량 제어 인터페이스를 제공하고 WebRTC를 이용한 실시간 영상 전송 및 데이터 전송과 원격 개입이 가능하게 하는 안전 모니터링으로 구성되어 있다.

3.5 AI 기반 이상 감지

본 모듈의 목표는 순찰 임무 중 발생할 수 있는 주요 이상 상황을 로봇 스스로 실시간 탐지하여 즉각적인 대응을 가능하게 하는 것이다. 이를 위해 고성능 객체 탐지 모델을 선정하고, 데이터

세트를 준비하여 전이 학습으로 커스텀 모델을 만들어 적용하였으며, ArUco 마커를 인식하도록 하여 로봇의 현재 위치와 출입자의 신원을 식별할 수 있도록 하였다. 이는 Host PC에서 자율 순찰 기능을 수행하기 위한 최소한의 기능을 구현한 것이다. 로봇의 제한된 온보드 환경에서 효율적으로 동작하도록 최적화하였다. 다음의 그림 6은 이상감지 모듈 내의 커스텀 모델에 의한 이상 감지와 ArUco Maker 인식 모듈에 의한 위치 확인과 출입자 확인 모듈의 구성을 나타낸 것이다.

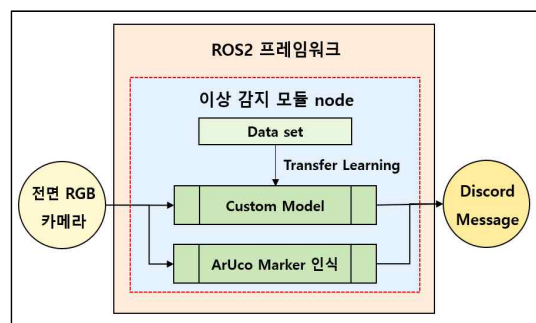


그림 6. ROS2 기반 이상 감지 모듈
Fig. 6. Anomaly detection module based on ROS2

고품질의 AI 모델을 학습시키기 위해 가장 중요한 것은 양질의 데이터로서, 보안 및 안전 분야에 특화된 자체 커스텀 데이터셋을 구축하였다. 또한, 목표 클래스: **/화재/연기(Fire/Smoke)', '쓰러진 사람(Fallen Person)', '무단 침입자(Intruder)**의 세 가지 핵심 클래스를 정의하였다. 데이터 수집은 공개된 안전과 관련된 화재 이미지 데이터셋과 실제 환경에서 직접 촬영 및 연출한 이미지를 혼합하여 약 5,000장의 초기 데이터셋을 구성하였다. 데이터 증강(Data Augmentation)을 위해 수집된 이미지에 대해 회전, 좌우 반전, 밝기 및 채도 조절, 무작위 노이즈 추가 등 다양한 증강 기법을 적용하여 학습 데이터의 양과 다양성을 확보하였다. 실시간성과 정확도를 동시에 고려해야 하는 로봇 환경에 적합

한 모델로, 1-stage detector인 YOLOv11을 선정하였다. 전이 학습으로 대규모 데이터셋(COCO)으로 사전 학습된 YOLOv11 모델을 기반으로, 구축한 커스텀 데이터셋을 이용해 미세 조정(Fine-tuning)을 진행했다. 이를 통해 상대적으로 적은 양의 데이터로도 특정 도메인에 대한 높은 탐지 성능을 효율적으로 달성할 수 있다. 다음의 그림 7은 이상 감지 모듈을 실행하기 위해 필요한 패키지를 나열한 것으로, webrtc_producer 모듈이 반드시 포함되어야 한다.

```
1 import cv2
2 import time
3 from flask import Flask, Response, render_template, request, jsonify
4 from multiprocessing import Queue
5 from webrtc_producer import start_webrtc, send_command, ensure_normal_mode_once
6 import threading
7 from ultralytics import YOLO # YOLO 모델 임포트
8 import logging
```

그림 7. 모듈의 패키지
Fig. 7. Package of Module

다음의 그림 8은 이상 감지 모듈의 주요 함수를 구현한 것이다. Flask 기반의 웹앱으로 실행되어 결과의 확인이 가능하다.

```
10 logging.basicConfig(level=logging.INFO)
11
12 app = Flask(__name__, template_folder='templates')
13 frame_queue = Queue(maxsize=10)
14 command_queue = Queue(maxsize=10)
15
16 # YOLO 모델 로드
17 yolo_model = YOLO('templates/yolo11n.pt') # 모델 파일 경로
18 # WebRTC 프레임 수신 시작 (명령 큐도 전달)
19 start_webrtc(frame_queue, command_queue)
20
21 > def generate():...
22
23 @app.route('/video_feed')
24 def video_feed():
25     return Response(generate(),
26                     mimetype='multipart/x-mixed-replace; boundary=frame')
27
28 @app.route('/')
29 def index():
30     return render_template('index.html')
31
32 @app.route('/move', methods=['POST'])
33 def move():
34     data = request.get_json()
35     direction = data.get('direction')
36     send_command(command_queue, direction)
37     return jsonify({'status': 'ok', 'direction': direction})
38
39 @app.route('/joystick', methods=['POST'])
40 def joystick():...
41
42 @app.route('/start_control', methods=['POST'])
43 def start_control():...
44
45 if __name__ == '__main__':
46     app.run(host='0.0.0.0', port=5010, debug=False)
```

그림 8. 모듈의 주요 함수
Fig. 8. Main function of Module

다음의 그림 9는 카메라의 영상을 판별하고 그 결과를 생성하여 리턴하는 generate() 함수를 나타낸 것이다.

```
21 def generate():
22     last_detect_time = 0
23     last_boxes = []
24     while True:
25         if not frame_queue.empty():
26             img = frame_queue.get()
27             now = time.time()
28             # 1초에 한 번만 YOLO 추론
29             if now - last_detect_time > 1.0:
30                 results = yolo_model(img)
31                 last_boxes = []
32                 for result in results:
33                     boxes = result.boxes
34                     for box in boxes:
35                         cls = int(box.cls[0])
36                         label = yolo_model.names[cls]
37                         if label == "person":
38                             x1, y1, x2, y2 = map(int, box.xyxy[0])
39                             last_boxes.append((x1, y1, x2, y2))
40                 last_detect_time = now
41             # 이전 결과(박스)만 영상에 표시
42             for (x1, y1, x2, y2) in last_boxes:
43                 cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
44                 cv2.putText(img, 'person', (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
45             ret, jpeg = cv2.imencode('.jpg', img)
46             if not ret:
47                 continue
48             yield (b'--frame\r\n'
49                  + b'Content-Type: image/jpeg\r\n\r\n' + jpeg.tobytes() + b'\r\n\r\n')
50         else:
51             time.sleep(0.01)
```

그림 9. Generate 함수
Fig. 9. Generate function in module

3.6 Discord 메시지 서비스 구축

다음의 그림 10은 로봇의 상태 등을 알리기 기능을 사용하기 위한 Discord Bot을 이용한 메시지 서비스의 코드를 나타낸 것이다.

```
1 try:
2     if DISCORD_SINKS_AVAILABLE:
3         # Discord.py 2.x+ sinks 사용
4         enhanced_sink = EnhancedWaveSink(self)
5         voice_client.start_recording(enhanced_sink, self._recording_finished_callback)
6         print("Discord sinks를 이용한 녹음 시작")
7         self.is_recording = True
8         return True
9
10    elif AUDIO_LIBS_AVAILABLE:
11        # PyAudio 대안 사용
12        return self._start_pyaudio_capture()
13
14    else:
15        print("❌ 사용 가능한 오디오 캡처 방법이 없습니다")
16        return False
17
18 except Exception as e:
```

그림 10. Discord 메시지 서비스 구현 코드
Fig. 10. Discord Message Service Python Code

4. 구현 및 실험 결과

본 장에서는 제3장에서 구현한 코드를 종합하여 최종 사용자가 볼 수 있는 Interface를 구현하고 이를 바탕으로 설계한 시스템의 실제 구현 내

용과 그 성능을 검증하기 위한 체계적인 실험 결과를 상세히 기술한다. 먼저 시스템 구현에 사용된 하드웨어 및 소프트웨어 환경을 소개하고 구현된 시스템의 실제 작동 모습과 50회의 반복 실험을 통해 도출된 정량적/정성적 결과를 분석하여 제안 시스템의 우수성과 실용성을 확인하였다.

4.1 구현 환경

본 연구의 통합 시스템은 다음의 그림 11과 같이 로봇 플랫폼과 AWS에 구현된 원격 관제 WebAPP으로 구성되어 있다.

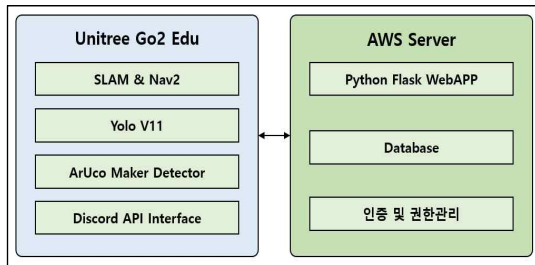


그림 11. 구현 환경 구성도

Fig. 11. Implementation Environment Diagram

로봇 시스템에 있는 HostPC에 해당하는 하드웨어(Hardware)는 다음의 표 2와 같은 구성으로 되어 있다.

표 2. 하드웨어 구성

Table 2. Hardware Configuration

구분	구성
로봇 플랫폼	Unitree GO2 EDU
Host PC	NVIDIA Jetson Orin NX
주요 센서	4D LiDAR, RGB카메라, Depth카메라, IMU
Web APP Server	Amazon_AWS
Network	Wifi 6(802.11ax)

Host PC의 소프트웨어(Software) 구성은 다음의 표 3과 같은 구성으로 되어 있다.

표 3. 소프트웨어 구성

Table 3. Software Configuration

구분	구성
OS	Ubuntu 22.04
ROS	ROS 2 Humble
Language	Python
Major Packages	abizovnuralem/go2_ros2_sdk legion1581/go2_webrtc_connect RoboSense-LiDAR/rslidar_sdk realsense2_camera ros2_go2_video slam_toolbox nav2_bringup rviz2, discord.py[voice]

4.2 구현 결과

본 연구에서 다중 센서 융합 SLAM을 이용하여 실험 환경(대학 건물 3층)을 맵핑 하면서, 지정된 순차 경로를 따라 자율 주행 하도록 하였다. 주행 구간에는 장애물을 배치하여 통과 하도록 하였으며 로봇은 이동 중에 사람을 판별하고 ArUco 마커를 인식하여 출입자를 판별하였으며 SLAM-Nav2 없이 LiDAR 센서 기반으로 장애물을 통과하는 것을 확인하였다. Discord 메시지를 이용하여 로봇의 현재 상태를 묻고 응답을 들 수 있었으며 이상감지를 하였거나 출입자를 판별 하였을 때 결과를 즉시 통보 하였다. 이러한 자율주행로봇의 관제 시스템을 위해서 WebAPP으로 UI를 구현하였다.

다음의 그림 12는 원격제어와 관제 모니터링을 위해서 제작한 WebAPP의 비디오 스트리밍과 로봇 제어 부분이다.

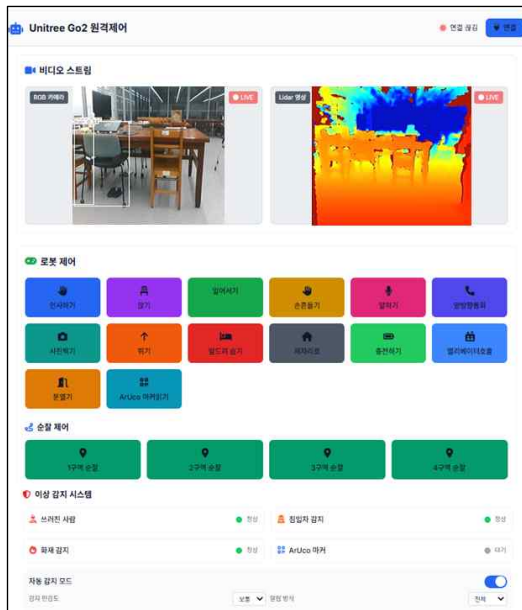


그림 12. WebAPP 원격 제어부
Fig. 12. Remote Control WebAPP

다음의 그림 13은 WebAPP의 Telemetry 부분으로 로봇의 상태와 시스템 정보 네트워크 상태 및 시스템 로그 기록을 확인할 수 있다. 이 밖에 구현된 시스템에는 로봇의 수동 방향 제어를 위한 조이스틱 UI가 포함되어 있다.

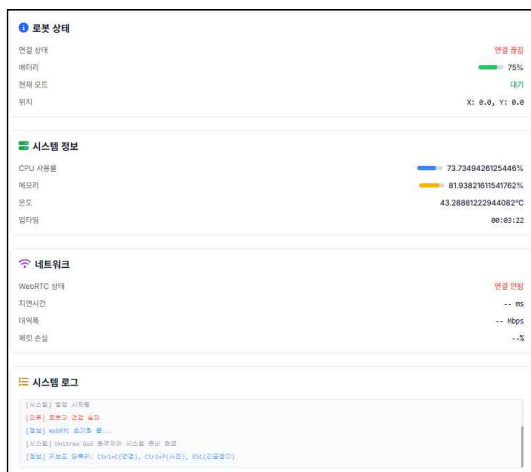


그림 13. WebAPP Telemetry
Fig. 13. Telemetry of WebAPP

4.3 실험(Experimental)

구현된 시스템의 성능을 객관적으로 평가하기 위해, 세 가지 핵심 기능에 대해 각각 50회의 반복 실험을 수행하였다.

4.3.1 자율주행 중 장애물 회피 돌파 실험

다음의 그림 14는 구간 자율주행 중에 장애물을 회피하고 있는 모습을 나타낸 것이다. 이 실험은 총 50회 실시하여 50회 모두 통과하여 100%의 성공률을 기록했다.

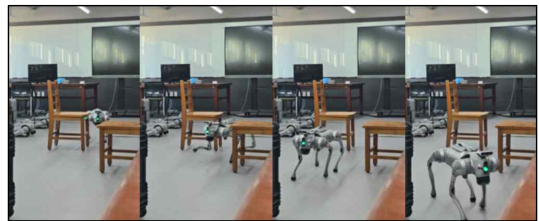


그림 14. 장애물 회피
Fig. 14. Obstacle avoidance

4.3.2 이상 감지 및 ArUco Marker 탐지 실험

다음의 표 4는 이상감지 및 ArUco Marker에 대한 탐지 실험으로 각각 50회의 시도와 결과를 정리한 것이다.

표 4 이상감지 및 ArUco Marker 탐지

Table 4 Object and ArUco Marker Detection

탐지 종류	시도	성공	(%)
쓰러진 사람	25	25	100
불꽃	25	23	92
장소 ArUco	25	25	100
신분증 ArUco	25	25	100

‘쓰러진 사람’의 경우 명확한 형태와 충분한 데이터 세트에 100% 탐지에 성공했으며, ‘연기’나 ‘불꽃’과 같이 형태가 불분명하고 계속 변하는

경우 92%의 비교적 높은 탐지 성공률을 기록했다. 이는 커스텀 모델을 이용한 탐지 임무의 성공률은 96%로 나타났다. 평균 추론 속도는 약 26.3 FPS로, 자율주행 중에도 탐지가 충분히 가능하다는 것을 확인하였다. ArUco Marker 탐지는 50회 모두 성공하여 100%의 탐지 성공률을 보여주었다.

4.3.3 로봇 제어 명령 및 원격 모니터링

관제 시스템의 지연시간 및 신뢰도 측정 결과는, 영상의 평균 지연시간은 약 145ms, 제어 명령의 평균 지연시간은 약 112ms로, 모두 200ms 이내의 매우 낮은 수준을 유지하여 관리자가 실시간에 가깝게 상황을 인지하고 제어할 수 있음을 보여 주었다. 명령 수행의 결과는 총 50회의 제어 명령 중 단 1회의 유실(loss)만 발생하여 98%의 매우 높은 전달 신뢰도를 기록했다.

다음의 그림 15는 스마트폰에서 수신한 Discord Bot이 이상 감지 또는 출입자를 식별했을 때 보내주는 메시지와 !info 명령으로 조회한 로봇의 상태를 나타낸 것이다.

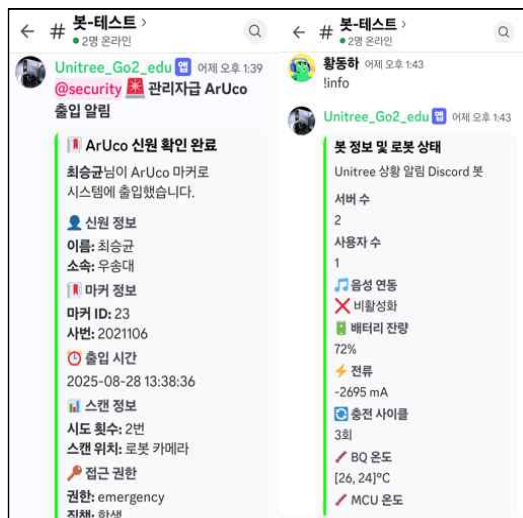


그림 15. Discord 메시지
Fig. 15. Discord Message

이상의 실험 결과들은 본 논문에서 제안하고 구현한 통합 자율 순찰 시스템은 자율 주행, 실시간 지능형 인지, 저지연 원격 제어 측면에서 모두 95% 이상의 높은 신뢰도와 우수한 성능을 보임을 명확하게 입증하였다.

5. 결론 및 향후 연구

본 논문에서 플랫폼으로 사용한 4족 보행 로봇은 휴머노이드 로봇에 비해 주행 안정성 측면과 경제성 측면에서 좀 더 빠르게 보급될 수 있는 로봇 플랫폼으로 판단이 되었다. 이를 활용하기 위한 자율주행과 로봇 제어를 모두 충족시키는 4계층 아키텍처를 통해 WebRTC 프로토콜을 ROS2와 통합하여 실시간 수준의 저지연을 구현하여 자율 순찰 중 원격 개입(Tele-intervention)이 필요한 보안-안전 시나리오에서 사용자가 지각 가능한 지연을 최소화하고 조작 신뢰도를 확보할 수 있음을 확인하였다. ROS2 기반에 자율주행과 딥러닝을 이용한 이상 감지 및 로봇의 위치 정확도와 출입자의 신분 확인을 위한 ArUco 마커의 인식을 구현하고 로봇의 상태 조회와 이벤트 발생에 대한 알림을 SNS 툴인 Discord에서 구현하였다. 최종적인 UI를 WebAPP 기반의 원격제어 모니터링 시스템을 구현하여 향후 소방, 순찰, 등 위험 구간의 유해 물질이 유출된 지역을 별도의 특정 센서를 장착하고 현장을 정찰, 탐색할 수 있는 기초를 마련하였다. 또한, 개별 기술의 구현에 머물지 않고, 종합적이고 유기적인 시스템으로 통합하였다. 본 논문은 기존 연구와의 차별성으로 다음과 같이 정리할 수 있다. 첫째, 바퀴형 위주의 이동 로봇 대비 비정형 지형에서의 기동성과 장애물 극복 능력이 탁월한 4족 보행 플랫폼을 채택하여 시스템을 구현하였다. 둘째, ROS2-Nav2-SLAM-WebRTC를 단일 파이프라인으로 유기적으로 통합하여 200ms(실

제 영상 145ms 제어 112ms)이하의 지연을 구현하였다. 셋째, YOLOv11기반 경량 탐지와 맞춤형 데이터셋으로 연기, 쓰러진 사람에 대한 탐지 성능과 실시간성을 동시에 확보하였다. 넷째, Aruco-Discord를 통한 신원 확인, 이벤트 알람까지 포함한 엔드-투-엔드 보안 운용 시나리오를 구현하였다. 향후 연구 방향으로 환경적 요인, 지형적 요인을 극복할 수 있는지 실험 방법을 확장하고, SLAM 데이터를 직접 작성하여 작성된 지도를 기반으로 NAV2를 이용한 AI기반 순찰 경로 및 스케줄 작성, 특정 상황에 따라 로봇 플랫폼에 교체 가능한 센서 키트를 장착할 수 있는 스테이션 개발 연구가 계속되어야 할 것이다.

참 고 문 헌

- [1] Armat, C., et al., "A review of the security of the Internet of Things", In 2017 13th International Conference on Semantics, Knowledge and Grids (SKG), IEEE, pp.187-190, 2017.
DOI:10.1109/SKG.2017.00035
- [2] LEE, Bryce Kenji Tim-Sung, "Design of a humanoid robot for disaster response", 2014. PhD Thesis, Virginia Tech.
<http://hdl.handle.net/10919/47492>
- [3] Raibert, M., et al., "Dynamically stable legged locomotion", MIT Press. (Note: For a representative paper: "Legged robots that balance," IEEE Transactions on Robotics and Automation, Vol.5, No.2, pp. 129-141, 1986. DOI: 10.1109/70.24021
- [4] Nagatani, K., Ishigami, G., & Okada, Y., Modeling and control of robots on rough terrain. In Springer handbook of robotics Cham: Springer International Publishing. pp. 1267-1284, 2016.
DOI: 10.1007/978-3-319-32552-1_50
- [5] Chetoui, M., Akhloufi, M. A., "Fire and Smoke Detection Using Fine-Tuned YOLOv8 and YOLOv7 Deep Models", Fire, Vol.7, No.4 pp.135. 2024.
DOI: 10.3390/fire7040135
- [6] Sozol, M. S. S., Mondal, M. R. H., & Thamrin, A. H., "Indoor fire and smoke detection based on optimized YOLOv5. PLoS One", Vol.20, No.4, e0322052, 2025.
DOI: 10.1371/journal.pone.0322052
- [7] Johnson, B., Salter, M., "Discord's Design Encourages "Third Place" Social Media Experiences", arXiv preprint arXiv: 2501.09951.
DOI: 10.48550/arXiv.2501.09951v1
- [8] Siddiqui, Z., et al., "Low-latency tele operation of a humanoid robot using WebRTC. In 2019 IEEE-RAS 19th International Conference on Humanoid Robots(Humanoids), IEEE, pp.42-49, 2019.
DOI: 10.1109/Humanoids43949.2019.9035043
- [9] M. Xia, M. Shirazipour, Y. Zhang, H. Green and A. Takacs, "Optical service chaining for network function virtualization", in IEEE Communications Magazine, vol.53, no.4, pp.152-158, April 2015. DOI: 10.1109/MCOM.2015.7081089.
- [10] Macenski, S., et al., "Robot Operating System 2: Design, architecture, and uses in the wild", Science Robotics, Vol.7, No.66, eabm6074, 2022.
DOI: 10.1126/scirobotics.abm6074
- [11] Grisetti, G., et al. "A tutorial on graph-based SLAM", IEEE Intelligent Transportation Systems Magazine, Vol.2, No.4, pp.31-43. 2010.
DOI: 10.1109/MITS.2010.939925
- [12] Kümmerle, R., et al., "g2o: A general framework for graph optimization", 2011 IEEE International Conference on Robotics and Automation, pp.3607-3613, 2011. DOI: 10.1109/ICRA.2011.5979949
- [13] Hess, W., et al., "Real-time loop closure in 2D LIDAR SLAM", In 2016 IEEE International Conference on Robotics and

- Automation (ICRA), IEEE, pp.1271-1278, 2016. DOI: 10.1109/ICRA.2016.7487258
- [14] Qin, T., et al. "VINS-Mono: A robust and versatile monocular visual-inertial state estimator", IEEE Transactions on Robotics, Vol.34, No.4, pp.1004-1020, 2018. DOI: 10.1109/TRO.2018.2853729
- [15] Wisth, D., et al., "Unified multi-modal landmark tracking for Tightly-coupled Lidar-Visual-Inertial odometry", In 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 11952-11958, 2021. DOI: 10.1109/ICRA48506.2021.9561005
- [16] Xu, W., et al., "FAST-LIO2: Fast, robust, and versatile LiDAR-inertial odometry", IEEE Transactions on Robotics, Vol.38, No.5, pp. 2613-2630, 2022. DOI: 10.1109/TRO.2022.3141876
- [17] Redmon, J., et al., "You only look once: Unified, real-time object detection", In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 779-788, 2016. DOI: 10.1109/CVPR.2016.91
- [18] KK. U. Dujia, I. A. Khan and M. Alsuhaibani, "Video Surveillance Anomaly Detection: A Review on Deep Learning Benchmarks", in IEEE Access, vol.12, pp. 164811-164842, 2024. DOI: 10.1109/ACCESS.2024.3491868.
- [19] Wang, C. Y., et al., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", In 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp.7464-7475, 2023. DOI: 10.1109/CVPR52729.2023.00721
- [20] Howard, A. G., Zhu, M., et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. 2017. DOI: 10.48550/arXiv.1704.04861
- [21] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez., "Automatic generation and detection of highly reliable fiducial markers under occlusion", Pattern Recognition, Vol.47, No.6, pp.2280-2292, 2014. DOI: 10.1016/j.patcog.2014.01.005
- [22] Seo, K. W., Hwang, D. Y., "Re-estimate the Robot Position by Estimating the Location of Unknown ArUco Markers with Feature Velocity Aid", The Journal of the Korean Institute of Electrical Engineers, Vol.73, No.4, pp. 711-719, 2024. DOI: 10.5370/KIEE.2024.73.4.711

저 자 소 개



황동하(Dong-Ha Hwang)

2011.2 한국방송통신대학교 법학과 학사
2022.8 우송대학교 IT융합 석사
2024.3 공주대학교 컴퓨터공학과 박사과정
2023.3-현재 우송대학교 IT융합 겸임교수
<주관심분야> 휴머노이드 로봇, 인공지능,
인공지능 에이전트



김재웅(Jae-Woong Kim)

1983.2 중앙대학교 전자계산학과 학사
1988.2 중앙대학교 컴퓨터공학과 석사
2002.2 대전대학교 컴퓨터공학과 박사
1992.8-현재 국립공주대학교 천안공과대학
컴퓨터공학부 교수
<주관심분야> 소프트웨어공학, 인공지능
시스템, 멀티미디어공학, 빅데이터



김동현(Dong-Hyun Kim)

1986.2 중앙대학교 전기공학과 학사
2005.2 공주대학교 컴퓨터멀티미디어공학과
석사
2010.2 공주대학교 컴퓨터공학과 박사
2021.9-현재 나사렛대학교 IT인공지능학부
교수
<주관심분야> 인공지능, 로봇 제어, 멀티미
디어 시스템