물류 운송을 위한 차량 군집 추적 로봇 “KEEP GOing”
YOLOv7-tiny 및 Jetson Orin Nano를 이용한 로봇 구현

노윤지 조민지
서울과학기술대학교 인공지능응용학과
지도 교수: 박종열

Abstract

The study focuses on developing a cluster tracking robot for logistics purposes, aiming to resolve the existing issues of high sensor costs and limited object recognition accuracy in traditional tracking robots. In this study, we use YOLOv7-Tiny, an optimized and lightweight deep learning model for the Jetson Orin Nano, to enhance object recognition accuracy. By controlling the robot's direction and speed solely using depth information obtained from Intel's RealSense camera, we address the high-cost issues associated with mass production of these robots and aim to solve traffic accident problems involving logistics vehicles.

I. 서론

군집 추적 로봇이란 여러 대의 로봇을 한 번에 움직이는 기술을 의미한다. 이 기술은 값비싼 로봇 1대를 사용하는 것이 아니라 저렴하고 간단한 로봇을 많이 생산하여 움직이게 만듦으로써, 적은 에너지와 비용으로 같은 작업을 여러 곳에서 수행할 수 있다는 장점을 가지고 있다.

군집 추적 기술은 현재 제조업 분야에서 주로 사용되고 있으며, 최근 자율 주행 차량에 대한 관심이 집중되면서 해당 기술을 접목한 자율 주행 차량 연구가 진행되고 있다. 경찰청 교통사고 통계에 의하면[1] 지난 5년간 (2019~2023년) 졸음운전으로 인한 교통사고가 총 1만 765건으로 하루 평균 5.9건이 발생했다. 특히, 고속도로 차량을 기준으로 차량 10만대당 졸음운전 사고는 특수차가 13.6건, 승합차는 11.2건, 화물차는 10.6건으로 화물 운송 등의 업무 목적으로 운행하는 차량이 졸음운전에 취약했다. 화물차 운전자의 하루 운행 시간이 수입과 비례하기 때문에 피로한 상태에서 주행하여 졸음운전 발생 확률이 높은 것으로 밝혀졌다. 그러나, 기존의 추적 로봇은 주로 LiDAR 및 RGB-D 카메라 등의 센서를 사용하여 Visual SLAM 알고리즘을 진행하기 때문에 높은 객체 인식 정확도를 보이나 많은 센서 사용으로

높은 비용이 청구된다. 더불어, 센서를 사용하지 않는 경우에는 주로 단순 HSV 색상 검출 알고리즘을 사용하기 때문에, 추적 타겟과 비슷한 장애물이 많은 곳에서는 정확한 타겟 추적이 불가능하다. 하드웨어(HW) 측면에서는 RAM 사용량에 따라 사용할 수 있는 소프트웨어(SW) 알고리즘의 성능이 크게 달라진다.

따라서, 본 연구는 군집 추적 기술을 사용하여 화물 차량 교통사고를 예방하는 것을 목표로 하여 NVIDIA의 Jetson Orin Nano를 활용한 객체 추적 자율 운전 차량을 구현하고자 한다. 추적하고자 하는 대상인 R/C카를 촬영 후 라벨링(Labelling) 데이터셋을 구축하였으며, 3가지 객체 인식 모델을 비교하였고, 실시간 추적이 가능한 모델 중에서도 하드웨어 성능 및 객체 인식 정확도, FPS 등을 고려하여 가장 최적의 딥러닝 모델로 YOLOv7-tiny를 최종 선택하였다.

해당 모델을 사용하여 타겟 인식을 진행하였고, Intel의 RealSense 카메라를 통해 얻은 깊이 정보를 기반으로 하드웨어의 종방향 및 횡방향 제어하였다. 최종적으로 구현한 차량이 실시간으로 R/C카를 인식하여 이를 추종하는 차량 로봇을 구현하였다. 해당 로봇은 높은 비용문제와 객체 인식 정확도를 개선함으로써, 화물 운송 차량 교통사고 예방 이외에도 군사 목적 등 다양한 곳에서 범용적으로 사용되는 효과를 기대할 수 있다.

II. 관련 연구

기존 연구[2]는 군집 로봇의 추적을 위하여 LiDAR 및 카메라 등의 센서를 사용하여 SLAM (simultaneous Localization And Mapping), DWA(Dynamic Window Approach) 등의 방법으로 해당 공간에 대한 지도를 그려 로봇 자신의 위치를 파악하고, 침입자를 탐색한다. 이후 A* 알고리즘 및 BFS(Brute-Force-Search) 알고리즘을 사용하여 로봇이 순찰하는 중에 카메라에 침입자를 인식하면 침입자의 Bounding Box 좌표를 기반으로 해당 2가지 알고리즘을 활용하여 도망치는 침입자의 경로를 추정하고, 침입자가 갈 수 있는 구역 내에서 침입자와 가장 최단 거리에 있는 로봇에게 순찰을 진행할 수 있도록 하였다.

해당 연구는 기존의 추적 로봇과 같이 LiDAR 및 카메라를 사용하여 추적을 진행하고 있으며, SLAM을 사용하여 해당 공간에서의 순찰 로봇의 위치를 파악하고, 기존에 인식되지 않았던 로봇의 위치가 인식되었을 경우 침입자로 간주하여 순찰을 진행하는 것을 통해 기존에 일상생활에서 주로 접할 수 있는 자율 이동 로봇의 시스템과 크게 다른 점을 찾기 어렵다. 다만, 객체 인식 정확도 측면에서는 좋은 결과를 나타낼 수 있고, 단순 알고리즘을 사용하여 침입자를 인식하고, 침입자가 도망칠 경로를 예상하여 해당 퇴로에서 가까운 순찰 로봇을 배치할 수 있도록 하는 시스템이기 때문에 하드웨어의 사양이 다소 낮더라도 로봇을

구현하는 데에 해당 문제가 크게 작용하지 않는다. 단순 알고리즘을 사용하여 추적을 진행하지만, 여전히 많은 센서를 사용하고 있기에 여러 대의 로봇을 구현하기에는 높은 비용 문제가 발생한다. 비용 문제를 해결하기 위해 또 다른 연구에서는 스테레오 카메라만을 사용하여 자율 이동 로봇 시스템을 구현하였다[3]. 스테레오 카메라를 통해서 추적하고자 하는 객체의 좌표 및 객체와의 거리 깊이 정보를 추출하여 실시간으로 이동하는 보행자를 검출할 수 있도록 한다. 이동 로봇과 보행 시간의 거리와 위치 좌표 및 다른 물체 간의 상대 거리를 산출하여 보행자가 이동할 때 이동 로봇이 최적의 거리를 유지하며 추적을 진행하고, 더불어 곳곳에 놓여 있는 장애물을 평균 약 2%의 오차 이내로 정확하게 회피하여 자율 주행을 진행할 수 있게 한다. 해당 연구에서는 YCbCr 컬러 모델을 사용하여 사람의 피부색과 배경 영역을 각각 검출한 후 촬영된 영상을 기반으로 차분 필터(Difference Filter) 및 논리곱 연산(AND operation)을 사용하여 추적을 진행한다.

위의 소개된 2가지 연구는 본 연구에서 진행하고자 하는 추적 모델 로봇 구현을 주제로 하였으나 모두 단순 알고리즘 혹은 필터만을 사용하여 추적을 진행하여 높은 비용을 수반하거나 혹은 객체 인식 정확도가 비교적 낮다. 따라서, 본 연구에서는 오로지 RGB-D 카메라를 활용한 딥러닝 모델 기반의 객체 추적 로봇을 구현하고자 한다.

III. 연구 방법

1. 커스텀 데이터 구축

객체 추적의 핵심은 '추적하고자 하는 대상'과 일치하는가?'이다. 일반적으로 클래스 단위로 분류를 진행하는 객체 판별의 정확도를 높이기 위해서는 진행하기 동일 클래스 내에서의 '추적하고자 하는 대상'의 특징점을 잘 찾아내는 것이 중요하다.

본 연구에서 추적의 대상은 빨간색의 R/C카이므로, 해당 객체를 RealSense 카메라를 사용하여 여러 각도 및 거리에서 촬영했으며, 물체가 카메라 화면 내에서 벗어났을 때 인식되지 못하는 폐색 현상(Occlusion) 발생을 방지하기 위하여 물체를 절반 정도 가려진 상황을 연출한 이미지를 포함하였다. 직접 RealSense 카메라로 찍은 영상을 프레임 단위로 나누어서 총 675장의 이미지를 얻었고, Labelme 프로그램을 사용하여 'Car' 클래스로 정의하여 이미지 라벨링을 진행하였다. 이후, 해당 이미지 데이터를 7:3의 비율로 나누어 훈련 및 검증 데이터로 분류하였다.

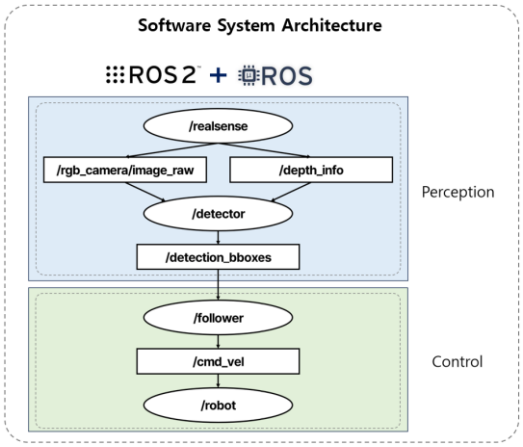
본 연구에서 사용하는 YOLO 모델에 적합한 데이터 형식이 txt이므로, json 파일로 저장된 라벨링 이미지를 모두 txt로 변환하여 NVIDA RTX A6000을 통해 모델 학습을 진행하였다.

Class	Data for detection	
	Training	Validation
Car	472	203

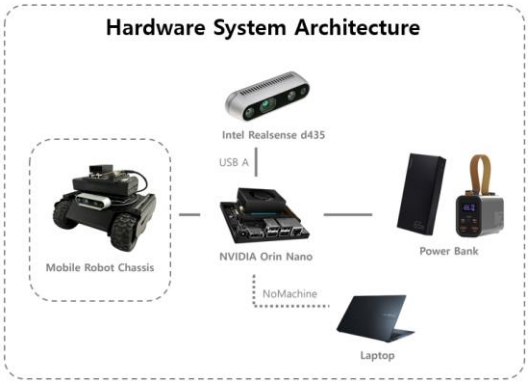
〈표1〉 객체 인식을 위한 커스텀 데이터셋 구축

2. 모바일 로봇 시스템 설계

본 연구에서 제안하는 ROS 기반의 모바일 로봇 시스템의 핵심 요소 기술은 인지와 제어이다. 이에 대한 구체적인 내용을 SW 및 HW시스템으로 나누어 설명하고자 한다. 각각의 다이어그램은 〈그림 1, 2〉와 같다.



〈그림1〉 소프트웨어 시스템 다이어그램



〈그림2〉 하드웨어 시스템 다이어그램

2.1 소프트웨어 시스템

2.1.1 ROS 시스템

개발한 SW시스템의 기반이 된 ROS2 (Robot Operating System 2)는 자율주행 로봇 및 로봇 응용 프로그램 개발에 많이 사용되는 오픈 소스 프레임워크로 실시간 처리, 보안, 분산 시스템 등의 기능을 강화하여 로봇 응용 프로그램을 효과적으로 개발할 수 있도록 지원한다[4]. ROS2와 ESP32 보드 간의 연동을 위해 사용한 Micro-ROS는 ROS의 주요 컨셉을 담고 있는 마이크로-컨트롤러 최적용 클라이언트 API로, Micro-ROS로 작성된 MCU를 노드로 만든 Micro-ROS Agent를 실행하여 ROS2 시스템에 연결을 자연스럽게 만들어준다. 따라서, 사용자는 마이크로-컨트롤러의 기능들을 일반 노드처럼 다룰 수 있고, 다른 ROS2 도구들을 활용할 수 있게 해준다[5]. 제안한 ROS 시스템은 특정 기능을 수행하는 노드(Node) 4개와 노드 간의 송수신 메시지인 토픽(Topic) 6개로 구성되어 있으며 노드와 토픽에 대한 자세한 설명은 아래와 같다.

■ **노드**

1) */realsense*

이미 구현되어 있는 RealSense 카메라 노드로써, RGB와 Depth에 관한 정보를 publish하며, 본 연구에서는 아래의 3가지 노드 토픽을 택하였다.

>>> */camera/color/image_raw*

>>> */camera/depth/image_rect_raw*

>>> */camera/aligned_depth_to_color/camera_info*

2) */detector*

직접 구현한 노드로, */realsense* 노드가 publish한 토픽을 subscribe하여 객체 인식 모델이 탐지한 객체의 Bounding box (bbox)에 관한 정보를 아래의 토픽으로 publish하도록 구현되었다.

>>> */yolo_detection/detector/detection_bboxes*

3) */follower*

직접 구현한 노드로, */detector* 노드가 publish한 토픽을 subscribe하여 제안한 상위 제어 로직에 따라 계산한 선속도와 각속도를 */cmd_vel* 토픽으로 publish하도록 구현되었다.

4) */robot*

Micro-ROS Agent 노드 실행 시, 생성되는 노드로, */follower* 노드가 publish한 토픽을 subscribe하여 제안한 하위 제어 로직에 따라 계산한 DC 모터의 PWM만큼 모터가 구동되는 기능이 구현되어 있다.

■ **토픽**

1) */rgb_camera/image_raw*

/realsense 노드에서 publish하는 토픽으로, sensor_msgs.Image 타입의 메시지이다. 객체 탐지를 위해 RealSense 카메라 센서가 받은 이미지 프레임을 publish하기 위함이다.

2) */depth_info*

/realsense 노드에서 publish하는 토픽으로, sensor_msgs.CameraInfo 타입의 메시지이다. Target과 카메라 간의 거리 추출을 위한 Depth 카메라의 메타 정보를 publish하기 위함이다.

3) */detectoin_bboxes*

/detector 노드에서 publish하는 토픽으로, car_tracking_interfaces.BoundingBoxArray 타입의 메시지이다. 직접 구현한 bbox 메세지를 요소로 갖는 배열이며, 이미지 프레임 별로 탐지된 객체의 bbox 리스트를 publish하기 위함이다. bbox 메세지의 구성 요소는 〈표2〉와 같다.

Data Type	Variable	Note
float64	conf	Confidence Score
int64	xmin	Minimum value of x coordinate of the target bounding box
int64	ymin	Minimum value of y coordinate of the target bounding box
int64	xmax	Maximum value of x coordinate of the target bounding box
int64	ymax	Maximum value of y coordinate of the target bounding box
int64	center_x	X coordinate of the centerpoint of the target bounding box
int64	center_y	Y coordinate of the centerpoint of the target bounding box
int16	class_idx	Class index of the target in class array
string	class_name	Class of the target
float64	distance	Distance extracted at the centerpoint of the target bounding box

〈표2〉 bbox 메시지의 구성 요소

4) */cmd_vel*

/follower 노드에서 publish하는 토픽으로, geometry_msgs.Twist 타입의 메시지이다. Target의 bbox와 frame 중심이 이루는 각도 그리고 추출된 거리 값을 사용하여 선속도인 vcmd와 각속도인 wcmd를 계산한 후, 각각을 메시지의 linear.x 그리고 angular.z 변수에 저장하여 publish하기 위함이다.

2.1.2 객체 인식 모델

한편, 소프트웨어 시스템에서 사용할 객체 인식 모델을 선정하고자 본 연구에서는 객체 인식 모델인 YOLO-FastestV2, YOLOv7-Tiny, YOLOv7-Tiny&DeepSORT 이 3개의 모델을 후보로 그 성능을 비교 분석하였다. 앞의 3개의 모델에 대한 자세한 설명은 아래와 같다.

■ YOLO-FastestV2

Yolo-FastestV2는 널리 사용되는 경량 객체 인식 모델이다. 모바일 장치에서 실시간으로 탐지하기 위하여 Dog-qiuqiu [6]이 제안하였다. 이 모델은 YOLOv5의 백본 네트워크를 ShuffleNetv2로 대체하는 동시에 FPN(Feature

Pyramid Network) 구조를 경량화하기 위한 것이며, 파라미터 크기는 약 250kB에 불과하다[7]. 본 연구에서 사용한 컴퓨터 사양에 적합한 경량화 모델이라 판단하여 후보로 선택하게 되었다.

- YOLOv7-Tiny

YOLOv7-tiny [8]는 Edge GPU용으로 설계된 네트워크 모델인 YOLOv7을 기반으로 구조를 단순화한 모델이다. 이 모델은 어느 정도 정확도를 희생하지만 YOLOv7에 비해 속도와 무게 측면에서 장점이 있다[9]. 최신 YOLO 모델이고, YOLOv4-Tiny보다 성능이 좋으며, 사용한 컴퓨터에서 실행 가능하여 후보로 선택하게 되었다.

- YOLOv7-Tiny & DeepSORT

객체 인식 모델과 객체 추적 모델을 함께 사용하여 여러 개의 객체가 인식되어도 특정 id의 객체만을 추적하고자 하였다. DeepSORT (Deep Simple Online and Real-time Tracking)는 SORT의 정확도를 개선한 다중 객체 추적 기술로써, SORT에서 연속된 프레임 간의 객체들의 관계를 판별할 때 CNN(Convolution Neural Network)의 특징 맵(Feature map)을 이용한다는 특징이 있다. 이러한 특징은 정확도를 비약적으로 개선시킨다는 장점이 있지만, 필요한 추가 컴퓨팅 연산의 양이 많아 FPS(Frame Per Second)를 떨어뜨린다는 단점이 있다. 따라서 자율주행 자동차와 같은 실시간 시스템의 경우, DeepSORT보다는 SORT가 더욱 적합하다고 간주되어 왔다[10]. 그러나, 구현하고자 하였던 객체 추적 기능의 추적 성능을 높이기 위해 딥러닝 객체 추적 모델의 사용이 불가피하다고 판단하여 후보로 선택하게 되었다.

Darknet, YOLOX 등의 경량 객체 인식 모델과 YOLOv7-tiny를 경량화한 모델을 사용하려는 시도를 하였으나, pytorch 모델을 경량 모델인 ONNX 및 TensorRT 모델로 변환하는 과정에서 발생한 문제가 해결되지 않아 후보에서 제외하였다.

2.2 하드웨어 시스템

2.2.1 하드웨어 구성

본 연구에서 개발한 모바일 로봇 “KEEP Going” 〈그림3〉의 주요 HW구성 부품은 〈표3〉에 나타내었다. 먼저, 메인 컴퓨터는 젝슨 오린 나노(NVIDIA Jetson Orin Nano)로, 젝슨 오린 나노 8GB 모듈과 참조 캐리어 보드로 구성되어 있는 개발자 키트를 사용하였다. 젝슨 오린 나노 8GB 모듈은 메모리가 8GB 128비트 LPDDR5이며, 1024개의 CUDA 코어, 32개의 3세대 Tensor Core, 6코어 Arm CPU가 포함된 NVIDIA Ampere 아키텍처 GPU를 갖추어 다수의 동시 AI 애플리케이션 파이프라인과 고성능 추론을 가능하게 한다[11]. 활용된 사례가 많아 참고 자료가 충분한 젝슨 나노(Jetson Nano) 개발자 키트가 아닌 젝슨 오린 나노 개발자 키트를 사용한 이유는 이전 세대 젝슨 나노의 80배의 성능으로 최신 AI 모델을 실행 가능하며, AI

성능이 크게 향상되었을 뿐만 아니라, 젝슨 나노에 비해 5.4배의 CUDA 컴퓨팅, 6.6배의 CPU 성능, 50배의 와트당 성능을 제공하기 때문이다[11]. 다음으로, 제어를 위한 드라이버 보드로 다중 주변기기 연결을 지원하는 Waveshare 사의 ESP32 기반의 드라이버 보드를 사용하였다. 컴퓨터의 낮은 메모리 사양으로 인하여 로봇과 target간의 거리 추정 모델을 사용하는 것이 어려운 관계로, 수월한 거리 추출과 객체 인식을 위하여 RGB-D 카메라인 Intel 사의 RealSense D435i를 사용하였다. 로봇 샤시로는 Waveshare 사의 6x4 Off-Road UGV를 사용하였다. 콘센트 선에 독립적인 주행과 컴퓨터 및 샤시의 전력 공급을 위하여 최대 출력이 65W인 파워뱅크 2개를 로봇에 적재하였다.

Component	Model
Computer	NVIDIA Jetson Orin Nano 8GB
Driver Board	ESP32
RGB-D Camera	Intel Realsense D435i
Robot Chassis	6x4 Off-Road UGV
Powerbank for Computer	NEXTU 30000mAh TYPE-C Max 65W
Powerbank for Robot Chassis	Morui 30000mAh TYPE-C Max 65W

〈표3〉 모바일 로봇의 주요 하드웨어 구성 부품

〈그림3〉 개발한 모바일 로봇 “KEEP GOing”

2.2.2 하드웨어 제어

RealSense 카메라를 통해 얻은 깊이 및 target으로 인식된 bbox 정보를 기반으로 로봇 제어를 진행하였다. 본 연구에서는 로봇 제어 시스템을 상위제어와 하위제어로 구분하였다. 상위제어 시스템은 ROS 시스템이며, 로봇 방향 제어 즉, 종방향 및 횡방향 제어가 진행된다. 하위제어 시스템은 하드웨어 시스템이며, 상위제어 시스템에서 구한 선속도와 각속도로 로봇 주행 제어가 진행된다. 주행 제어를 위하여 로봇의 DC모터에 전해질 PWM 값을 아래 식1을 통해서 구하였다. 선속도(vcmd)는 로봇이 직진하는 속도, 각속도(wcmd)는 로봇이 회전하는 속도이며, L은 로봇의 폭을 의미한다.

$$PWM\ of\ Motor\ A = vcmd + wcmd \times (L/2)$$

〈식1〉 왼쪽 모터(A) PWM 계산 공식

$$PWM\ of\ Motor\ B = vcmd - wcmd \times (L/2)$$

〈식2〉 오른쪽 모터(B) PWM 계산 공식

종방향 제어 값과 횡방향 제어 값은 서로 상호작용하여 영향을 끼치기 때문에 종방향 제어를 첫 번째로 진행하여 선속도의 계산 식의 파라미터를 최적화한 이후에, 각속도의 계산 식의 파라미터를 최적화하는 횡방향 제어를 진행하였다.

1) 종방향 제어

선속도는 PID 제어 시스템 (Proportional Integral Derivative Control System)을 기반으로 아래 〈식3〉를 통해서 구하였다.

$$vcmd = Gain \times (goal\_dist - Safe\_dist)$$

〈식3〉 PID 제어 기반의 선속도 계산 공식

Gain을 1로, 목표 거리와 안전 거리의 차이를 로봇과 target 위치와의 거리로 설정하였다. 또한, 주행 시 선속도의 영향력이 각속도보다 상대적으로 매우 커져 로봇이 회전하지 못하고 계속해서 직진하는 상황을 막기 위하여 선속도를 그것의 최댓값으로 제한하였다.

2) 횡방향 제어

각속도는 다음과 같은 〈식4〉을 통하여 구하였다. 로봇의 횡방향 제어를 위해 Pure Pursuit제어를 구현하였으며, 이는 로봇 차량의 뒷바퀴 중심이 경로 상의 목표 지점인 target까지 도달하기 위한 조향각(Steering angle)을 계산한다.

$$wcmd = tan^{-1}\frac{2Lsin\theta}{L_d}$$

〈식4〉 Pure Pursuit 각속도 계산 공식

L은 로봇의 축간 거리, θ는 카메라 프레임의 중심으로부터 target의 바운딩 박스가 이루는 각도이며, L_d (Look-ahead Distance)값은 전방주시거리로, 로봇이 현재 위치에서 일정 거리 앞의 목표 지점을 설정하는 기준 거리를 의미한다. L_d 값이 크면 로봇이 경로의 코너를 뭉뚝하게 지나치게 되는 Corner Cut-Off 문제가 발생하고, L_d 값이 작을 경우에는 목표지점을 너무 자주 바꾸게 되어 경로를 따라가는 과정에서 과도하게 흔들리거나 목표지점을 지나치게 되는 Overshoot 문제가 발생한다.

따라서, 로봇이 안정된 주행을 하고 선속도와 마찬가지로 각속도의 영향력이 커짐으로써 로봇이 직진하지 못하고 계속해서 회전하는 상황을 방지하도록 최적의 L_d를 찾았다.

V. 실험 결과

최종 Orin에서 사용한 훈련모델의 경우 및 ROS2 시스템에서 실행한 모델의 경우 각 2가지의 경우로 나누어 모델 비교를 진행하였다.

5.1 훈련 모델 in Orin

Orin에서 각 3가지 모델을 사용하여 모델 훈련을 진행하였다. 모두 YOLO모델을 사용하였으나, YOLOv7-tiny에 DeepSORT를 사용한 모델은 본 연구에서 사용된 하드웨어의 용량의 한계로 인하여 훈련이 불가하여 모델 정확도 비교 시에 제외하였다. FPS의 경우에는 일반적으로 30이상일 때 비로소 화면이 자연스럽게 움직이는 것처럼 해석되어 실시간 처리가 이루어졌다고 말한다. YOLOv7-tiny는 YOLO-FastestV2에 비해 전반적으로 FPS, mAP 정확도 및 추론(Inference)에서 높은 성능을 보였다. YOLOv7-tiny의 경우 FPS가 36.10로, YOLO-Fastest의 FPS 19.52보다 약 2배 높은 성능을 보였다. 더불어, YOLOv7-tiny의 추론 처리속도 또한 YOLO-Fastest에 비해 약 2배 정도

빠르다는 것을 확인하였다 〈표4〉.

결과적으로, YOLO-Fastest는 YOLOv7-tiny에 비해 파라미터의 수가 약 25배 정도 적기 때문에 전반적으로 정확도가 좋지 않았다.

MODEL	Parameters	mAP@0.5	FPS	Inference	Size
YOLO-FastestV2	238,496	0.583	19.52	29.93ms	352
YOLOv7-tiny	6,007,596	0.782	36.10	15.37ms	640
YOLOv7-tiny & DeepSORT	훈련 불가	훈련 불가	훈련 불가	훈련 불가	훈련 불가

〈표4〉 Orin에서의 훈련 모델 성능 비교

5.2 실행모델 in ROS2

실제 소프트웨어를 적용하여 로봇으로 구현하기 위해 ROS시스템에서 앞서 훈련시킨 모델을 가지고 객체 인식을 진행하였다. YOLOv7-tiny는 YOLO-FastestV2보다 정확도, FPS 및 추론 시간에서 전반적으로 높은 성능을 보였다. 훈련 모델에 비해서는 FPS가 약 13정도 낮아졌으나, 여전히 초당 20 프레임을 나타내어 다소 훈련 모델에 비해 부자연스럽게 보일 수 있으나, 객체 인식을 통한 실시간 추적이 여전히 잘 이루어지고 있다고 해석될 수 있다. 정확도는 0.936이며 Inference는 16임으로, 해당 모델이 빠르고 정확하게 실행되고 있음을 확인하였다 〈표5〉.

MODEL	Parameters	mAP@0.5	FPS	Inference	Size
YOLO-FastestV2	238,496	0.698	12.97	32.61ms	352
YOLOv7-tiny	6,007,596	0.936	23.98	16.75ms	640
YOLOv7-tiny & DeepSORT	실행 불가	실행 불가	실행 불가	실행 불가	실행 불가

〈표5〉 ROS2에서의 실행 모델 성능 비교

VI. 결론 및 향후 연구

6.1 결론

훈련 모델 및 실행 모델의 비교를 통해서 YOLOv7-tiny가 하드웨어의 RAM용량 등 종합적으로 고려하였을 때 가장 최적의 모델로써 해당 모델을 사용하여 로봇 구현을 진행하였다. 로봇 테스트를 진행한 결과 FPS는 약 29정도를 달성하였으며, target을 정확히 인식하여 추적을 진행하였다.

6.1.1 한계점

로봇 추적 테스트 진행 시, 테스트 환경에서 따라 객체가 잘못 인식되어 추적이 중단되거나 혹은 실행이 Delay되는 문제가 발생하였다.

첫째, target과 같은 색상의 객체를 taget으로 잘못 인식되는 문제가 발생하였다. 이는 적은 데이터로 인한 과적합(Overfitting)으로 인해 발생한 것으로 해석되며, 빛의 세기 혹은 굴절에 의해 target 인식에 영향을 받은 것으로 분석하였다.

둘째, 하드웨어 RAM용량 및 원격접속의 한계로 인한 Delay 및 모델 경량화 실패이다. 로봇이 target을 인식했음에도 불구하고 2~3초간의 delay가 발생하여 로봇이 즉각 반응하지 않는 문제가 발생하였다. 이는 NoMachine 애플리케이션(Application)을 통한 원격 접속으로 로봇의 실행 노드에 접근을 진행하였기 때문에, 해당 애플리케이션의 화면 그래픽이 차지하는 용량이 다소 높아 단순 터미널에서의 실행 때와 달리, 노드 실행 시에

컴퓨터 상에서 delay가 발생한 것으로 분석하였다. 더하여, 본 연구에서 사용한 RAM 용량은 총 8GB이며, 기본 시스템이 차지하는 용량을 제외하면, 6.3 GB를 사용할 수 있다.
이에 따라, 로봇을 실행하면 RAM 사용량이 4.8~5.2GB를 차지한다. 이 문제를 해결하고자 Swap memory를 사용하고자 하였으나, 로봇 실행이 차지하는 메모리가 전체 메모리의 80~90%를 차지하여 터미널이 freezing 되는 문제가 발생했다.

그러므로, RAM용량의 문제 해결을 위한 최적화된 모델 및 실시간 처리 속도 등 2가지를 종합적으로 고려하였을 때, 본 연구에서 가장 최적화와 경량화 된 모델이 YOLOv7-tiny이므로 최종 모델로 선택하여 추적 로봇 구현에 성공하였다.

6.2 기대 효과

물류 운송을 위한 군집 추적 로봇 “Keep GOing”을 사용하면 크게 3가지 효과를 기대할 수 있다.

첫째, 로봇 구현에 있어 비용 절감이 가능하여 다양한 산업분야에서 현실적으로 많은 사용이 가능하다. 정확한 target 인식을 위해 딥러닝 모델 YOLO를 사용함에 따라 오로지 카메라만을 사용하기 때문에 대량 생산 시에도 큰 비용 부담을 갖지 않는다.

둘째, 객체 추적 대상의 인식 정확도를 높여 안전한 물류 운송 자율 주행이 가능하다. 기존의 객체 추적 로봇은 장애물이 없는 곳에서 주로 사용되어 target이 잘못 인식되는 경우가 거의 없다. 그러나, 본 연구의 로봇은 딥러닝 모델을 사용하여 객체를 인식하기 때문에 다양한 환경에서 사용이 가능하여 공간적 제약의 문제를 해결함과 동시에 다양한 산업분야와의 접목이 가능하다. 즉, 공간적 제약의 한계를 극복한 안전한 물류 운송 자율주행이 가능하다.

셋째, 화물 운송 운전자의 졸음 운전 교통사고 피해를 방지할 수 있다. 선두 차량을 target으로 인식하여 적정거리를 유지하며 주행하기 때문에, 운전자의 졸음 혹은 건강의 문제에 따른 사고발생 확률이 감소하게 된다.

6.3 향후 연구 계획

본 연구의 문제점에 근거하여 빛의 밝기 혹은 굴절에 따라 타겟이 때때로 잘못 인식되는 문제를 해결하고자 한다. 특히, 훈련 데이터 과적합으로 인하여 발생한 잘못된 target 인식 추적 문제를 해결하여 다양한 환경에서 로봇을 실행했을 때에 정확하고 빠른 target 인식 및 추적이 되도록 본 연구보다 더 많은 커스텀 데이터를 구축하여 해당 문제를 해결하고자 한다.

더하여, 하드웨어 스펙 한계로 인해 소프트웨어 성능이 저하되는 문제와 로봇 제어 시, 값이 Delay되는 문제를 해결하고자 정확한 선속도 및 각속도를 구하기 위하여 여러번의 변수 값으로 테스트를 진행하여, 로봇과 target 간의 거리를 적절히 유지할 수 있도록 하여, 로봇 제어 노드 실행을 터미널로 진행할 수 있도록 하고자 한다. 이를 통해서, 화면 그래픽으로 인해

발생하는 컴퓨터 상의 delay 문제도 같이 해결하고자 한다.

본 연구에서 발생한 문제점 해결 이외에 객체가 카메라 frame에서 벗어났을 경우 혹은 동일한 클래스 중에서 특정 target을 추적하고자 하는 경우에 발생 가능한 Id-Switching 문제를 해결하여 보다 더 다양한 환경 및 다양한 객체를 추적할 수 있도록 추후에 이에 대해 연구를 진행할 예정이다.

VII. 참고문헌

[1] 김나성, "최근 5년간 졸음운전 교통사고 하루 평균 5.9건 발생, 토요일 최다 발생", 메디컬월드뉴스, 2024, https://medicalworldnews.co.kr/m/view.php?idx=1510960772

[2] 박종현, 안성은, & 조우현. (2020). ROS 기반 군집로봇의 지능형 추적 알고리즘 설계. *한국정보처리학회 학술대회논문집*, *27*(2), 545-547.

[3] 고정환, 김성일, & 김은수. (2006). 스테레오 카메라 기반의 적응적인 공간좌표 검출 기법을 이용한 자율 이동로봇 시스템. *한국통신학회 논문지*, *31*(1C), 26-35.

[4] Open robotics (Ed.). (2023, May 17). ROS 2 Documentation: Foxy. ROS 2 Documentation. https://docs.ros.org/en/foxy/index.html

[5] micro-ROS, 2024, https://micro.ros.org/docs/overview/features/

[6] dog-qiuqiu/Yolo-FastestV2: Based on Yolo's low-power, ultra-lightweight universal target detection algorithm, the parameter is only 250k, and the speed of the smart phone mobile terminal can reach ~300fps+ (github.com), 2022.

[7] Zhang, H., Xu, D., Cheng, D., Meng, X., Xu, G., Liu, W., & Wang, T. (2023). An improved lightweight yolo-fastest V2 for engineering vehicle recognition fusing location enhancement and adaptive label assignment. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *16*, 2450-2461.

[8] Li, B.; Chen, Y.; Xu, H.; Fei, Z. Fast Vehicle Detection Algorithm on Lightweight YOLOv7-Tiny. arXiv 2023, arXiv:2304.06002.

[9] Zhang, L., Xiong, N., Pan, X., Yue, X., Wu, P., & Guo, C. (2023). Improved object detection method utilizing yolov7-tiny for unmanned aerial vehicle photographic imagery. *Algorithms*, *16*(11), 520.

[10] 양수진, 정인화, 강동화, & 백형부. (2021). SORT 와 DeepSORT의 혼합을 이용한 실시간 다중객체 추적. *한국정보기술학회논문지*, *19*(10), 1-9.

[11] NVIDIA DEVELOPER, (2023, April 11), https://developer.nvidia.com/ko-kr/blog/develop-ai-powered-robots-smart-vision-systems-and-more-with-nvidia-jetson-orin-nano-developer-kit/