https://doi.org/10.7840/kics.2022.47.11.1884

SLAM 및 OCR을 이용한 자율 주행 기반 라스트 마일 배송 시스템 설계 및 구현

박 건 형◆, 조 성 윤*, 신 재 성*, 박 종 건*, 양 현 규*, 김 태 운°

Design and Implementation of the Self-Driving Based Last Mile Delivery System Using SLAM and OCR

Geon-hyeong Park◆, Seong-yun Cho*, Jae-seong Shin*, Jong-gun Park*,
Hyun-kyu Yang*, Taewoon Kim°

요 약

최근 배송 산업의 규모가 증가함에 따라 물류 자동화 기술 개발이 가속화되고 있다. 그중에서도 상품이 소비자에 전달되는 마지막 단계인 라스트 마일 배송이 주목받고 있으며, 라스트 마일 배송의 효율성 개선을 위한 연구와 개발이 활발히 진행되고 있다. 본 논문은 SLAM(Simultaneous Localization And Mapping) 및 딥 러닝을 사용하여 이동 로봇 기반의 라스트 마일 배송 자동화 기술을 구현하였다. 제안하는 기술은 딥 러닝 비전 기반의 OCR(Optical Character Recognition) 기법을 활용하여 택배 송장에서 수취인의 주소 및 목적지 텍스트 정보를 추출하고 SLAM 기술을 활용한 로봇 자율주행을 통해 택배를 수취인의 문 앞까지 배달한다. 또한, 배달 기사의 사용 편의성을 개선하기 위한 전용 애플리케이션을 개발했다. 실험 결과 제안하는 로봇은 배송 중에 발생할 수 있는 장애물 등에 동적으로 대처하여 정확한 목적지로 물건을 배송할 수 있음을 확인했고, 인력에 의존하는 라스트 마일 배송을 효과적으로 자동화할 수 있음을 보여준다.

키워드 : 물류 자동화 기술, 라스트 마일 배송, 동시적 위치추정 및 지도작성, 자율주행, 딥러닝
Key Words : Logistics Automation Technology, Last Mile Delivery, SLAM, Self Driving, Deep Learning

ABSTRACT

Recently, as the size of the delivery service market increases, the pace of the logistics automation technology is accelerating. Among others, the last step in a product's journey to customer doorstep called Last Mile Delivery has gained much attention, and thus, research and development is actively underway to enhance the efficiency of last mile delivery. This paper implements the mobile robot-based last mile delivery automation system using SLAM (Simultaneous Localization And Mapping) and deep learning. The proposed system extracts the text that contains the receiver's address and delivery destination from the invoice using deep learning vision based OCR (Optical Character Recognition), and then delivers a package to the receiver's front door by self driving using SLAM. Additionally, we have developed an Android application to improve

※ 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2021R1F1A1059109)
◆ First Author : Hallym University, Division of Software, po05105@gmail.com, 학생회원
° Corresponding Author : Pusan National University, School of Computer Science and Engineering, taewoon@pusan.ac.kr, 정회원
* Hallym University, Division of Software, josy1102@naver.com; rornfl916@gmail.com; ooostus@gmail.com; yhk974@gmail.com
논문번호 : 202207-127-D-RN, Received July 5, 2022; Revised August 17, 2022; Accepted August 17, 2022

the courier's usability. By experiment we have verified that the automated robot can accurately deliver a package to the receiver's front door while coping with obstacles dynamically during self driving. In other words, the proposed system can effectively automate last mile delivery that depends on manpower.

Ⅰ. 서 론

시간 및 장소에 구애받지 않으면서 편리하고 신속하게 물건을 구매할 수 있는 전자상거래 시장이 성장하면서[1] 택배·물류 시장의 규모도 커지고 있다.[2]

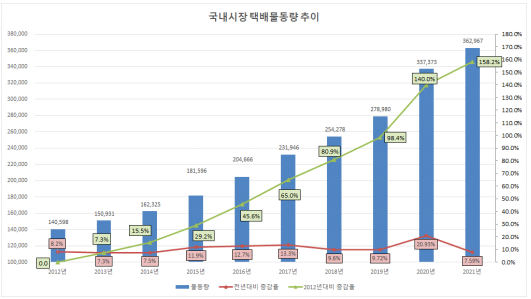
국가물류통합정보센터에 따르면 최근 10년간 생활물류 분야의 택배물동량과 택배 매출액, 국민 1인당 택배 이용 횟수는 꾸준히 증가하고 있다. 그중 국내에서 최초로 코로나 19가 발생한 2020년 같은 경우 물동량은 전년도 대비 20.93%가 증가하였으며 택배 시장 매출액은 18.4%가 증가하였다(참고: 그림 1).

현재 물류 산업은 로봇을 통한 자동화 기술이 도입된 물류 센터[3] 또는 물류 아웃소싱[4] 등을 기반으로 물류 체계 효율화를 달성하고 있다. 하지만, 상품이 소비자에게 전달되는 마지막 과정을 의미하는 라스트 마일 배송 영역은 여전히 인력에 의존하고 있으며, 일정 수준의 자동화를 달성한 다른 물류 단계와 다르게 많은 인력과 비용이 필요하다.[5] 2020년 기준 라스트 마일 배송 시장 규모는 7.5조 원으로 추산된다[6]. 라스트 마일은 단순한 서비스가 아닌 하나의 시장으로 구분할 수 있을 만큼 성장했으며 향후 계속 성장할 것[7]으로 예측된다.

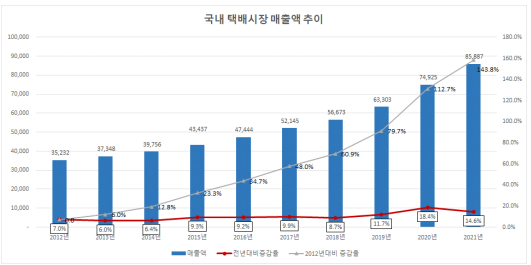
다양한 글로벌 기업들이 라스트 마일 배송 시장을 선점하기 위해 자동화 기술을 연구·개발하고 있으며, 그중 미국의 최대 전자상거래 업체인 '아마존'과 중국의 '징둥', '알리바바'는 ICT 기술을 활용한 배송 로봇 개발을 진행하고 있다[8]. 이에 따라, 글로벌 배송 로봇 시장 또한 크게 성장할 것으로 전망되며[9], 국내외에서도 다양한 관련 연구[10-13]가 활발히 이루어지고 있다.

본 논문은 물류 시스템의 효율성을 개선하기 위해 반드시 해결해야 하는 라스트 마일 배송 구간을 ICT 기술을 활용하여 자동화하는 기술을 구현한다. 제안하는 자동화된 라스트 마일 배송 기술은, 확장성이 뛰어나고 오픈 소스에 기반한 Robotis사의 TurtleBot3 범용 로봇 플랫폼[14]을 사용한다. 로봇에 탑재된 LiDAR 센서로 주변 환경을 인지하고 측정된 거리 값을 기반으로 실내 지도 작성 및 실시간 위치 추정을 수행한다. 완성된 환경 지도를 기반으로 로봇은 자율적으로 주행하며, 자체 개발한 모바일 애플리케이션에서 수취인의 정보가 담겨있는 송장을 촬영하면 Google Vision API를 활용한 딥 러닝 기반 OCR[15]을 통해 배송 목적지 인식에 필요한 정보를 자동으로 추출하고, 로봇은 해당 목적지에 물품을 배송한다.

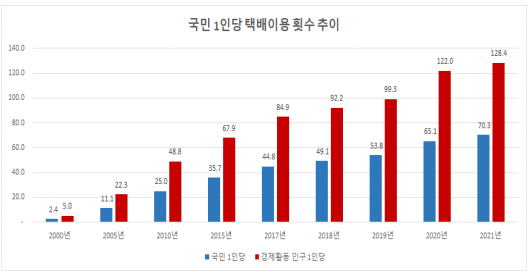
제안하는 기술은 라스트 마일 배송 과정에서 물건



(a) 최근 10년간 국내시장 택배물동량 추이
(a) Delivery quantity in the domestic market for the recent ten years



(b) 최근 10년간 국내 택배시장 매출액 추이
(b) Sales in the domestic delivery market for the recent ten years



(c) 국민 1인당 택배이용 횟수 추이
(c) Average number of delivery service uses per each citizen

그림 1. 연도별 생활물류통계
Fig. 1. Consumer logistics statistics by year

을 배송하는 사람이 직접 수취인의 문 앞까지 이동하지 않고 실내 자율주행이 가능한 로봇과 딥 러닝 기반의 사용자 애플리케이션을 통해 택배를 목적지까지 배송한다. 자동화된 라스트 마일 배송 시스템은 배송 인력이 직접 문 앞까지 이동하는 기존의 방식보다 더 많은 물품을 신속하게 배송할 수 있고, 이를 통해 배송 산업의 효율성과 수익성 증가를 기대할 수 있다.

논문의 구성은 다음과 같다. II. 본론에서는 제안하는 기술의 구현 방법을 로봇, 애플리케이션, 서버 부분으로 나누어 상세히 설명한다. III. 실험에서는 실환경에서 제안하는 로봇을 동작시켜 애플리케이션과 연동한 다음, 물건을 로봇에 적재해 원하는 위치로 이동시키고 하역하는 과정과 실험의 결과를 설명한다. IV. 결론에서는 본 논문의 결론, 활용 방안 및 기대효과, 향후 계획을 설명한다.

Ⅱ. 본 론

2.1 제안하는 라스트 마일 배송 시스템 개요

라스트 마일 배송 시스템 자동화를 위해 자율주행 로봇, 모바일 애플리케이션, 서버 시스템을 개발하였다. 제안하는 라스트 마일 배송 자동화 시스템의 주요 특징은 다음과 같다.

1. 딥 러닝 비전 기반의 OCR 기술을 사용하여 수취인의 목적지 정보 자동 추출
2. SLAM 기반의 실내 자율주행 및 택배 배송 자동화
3. 택배 기사 및 수취인의 사용 편의성을 위한 모바일 애플리케이션 개발
4. 로봇·서버·애플리케이션 연동을 통한 목적지 정보 전송 및 배송 상황 모니터링

본 논문에서 제안하는 시스템 구성 및 동작 방식은 그림 2와 같다.

제안하는 로봇 상단부에 배송할 물품을 적재하고, (1) 전용 애플리케이션으로 송장을 촬영하면 목적지 정보가 자동으로 추출된다. (2) 추출된 목적지 정보는 서버로 전송되고 이와 동시에 데이터베이스로부터 목적지 주소를 기반으로 수취인의 정보를 획득한다. (3) 로봇 서버는 수신한 정보를 기반으로 목적지 좌표 값을 획득하고 Navigation(자율 주행)을 명령한다. (4) 로봇은 해당 좌표까지 자율 주행하여 물품을 수취인의 문 앞까지 배달한다. 목적지에 도착하면 로봇은 컨베이어 벨트를 동작시켜 물건을 내려놓고 (5), (6) 배달 완료 메시지를 로봇 서버를 거쳐 애플리케이션으로 보낸 뒤 최초 위치로 복귀한다. 이후 (7) 애플리케이션은 데이터베이스로부터 받은 수취인 번호를 통해 배송 완료 SMS 메시지를 보내는 것으로 배송이 완료된다.

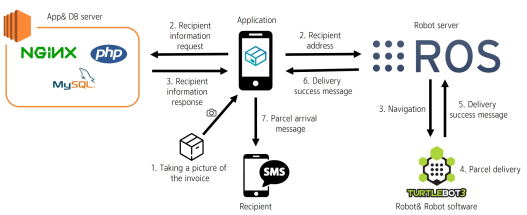


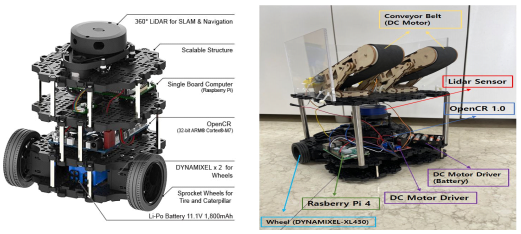
그림 2. 시스템 구성도
Fig. 2. System configuration

2.2 제안하는 라스트 마일 배송 시스템 구성

2.2.1 로봇 하드웨어 구성

그림 3(a)는 본 연구에서 참조 모델로 활용한 TurtleBot3 Burger 로봇의 원형이며, 그림 3(b)는 본 논문에서 제안하는 개선된 모델이다. 기존 모델은 로봇의 단면이 좁아, 물건을 싣기 적당하지 않고 배송 중 로봇이 쓰러지는 등의 문제가 발생하였다. 이를 개선하기 위해 플레이트와 샤프트를 추가하여 안정적으로 물건을 적재하고 배송할 수 있도록 로봇의 차체를 넓고 견고하게 설계하였다.

표 1은 로봇 구현에 사용된 모듈 목록이다. 로봇이 활동하는 지역의 지도를 형성하고 배송 목적지로 자율 주행하기 위한 핵심 부품인 라이더 센서는 기존 LDS-02 제품을 YDLIDAR사의 X4 LiDAR Sensor 제품[16]으로 교체하였다. 기존 제품은 최대 3m까지 인식이 가능하지만, 교체한 제품은 최대 10m까지 인식 가능하여 더 신속하고 정확하게 지도를 그릴 수 있으며 이동 중에 발생하는 장애물 및 지형변화에 대해서도 신속한 대처가 가능하다.



(a) 터틀봇3
(a) Turtlebot3

(b) 제안하는 로봇 구성
(b) Proposed robot configuration

그림 3. 로봇 하드웨어 구성도
Fig. 3. Robot harware configuration

표 1. 주요 부품 목록
Table 1. List of main parts

Part Name	Specification	Function
Robotis TurtleBot3 Burger	LDS-02, OpenCR 1.0 Raspberry Pi 3, Wheel, DYNAMIXEL (XL430), Li-Po Battery 11.1V 1800mAh	Robot Body
YDLIDAR X4 LiDAR Sensor	Range Radius(m): 0.12-10	Robot localization
Raspberry Pi 4 Model B	Gigabit Ethernet port, USB2.0/3.0 ports, Micro HDMI ports, Micro SD card slot, GPIO 40-pin header, WIFI 802.11 b/g/n/ac	Receive Message from Robot Server, Control Robot
OpenCR 1.0 Board	USB2.0 ports, TTL ports, RS485, UART ports	Control wheel (move)
Conveyor Belt	DC Motor(3V-6V) Wooden plywood	Box Carrier, Drop Box
DC Motor Driver 2A L298N Board	Input ports, Output ports	Control Conveyor Belt

로봇 서버와 통신하고 명령을 전달하는 단일 보드 컴퓨터(SBC, Single Board Computer)는 처리속도 개선 및 동작 안정성 개선을 위해 기존 Raspberry Pi 3 B+를 Raspberry Pi 4 B 모델로 교체하였고 ROS Noetic 버전이 설치된 Ubuntu 20.04 이미지[17]를 사용하였다. OpenCR 1.0 보드는 Raspbeery Pi로부터 명령을 받아 로봇의 바퀴 즉, 움직임을 제어하고 Li-Po 배터리에서 전원을 공급받아 Raspberry Pi로 전력을 공급하는 역할을 수행한다. Conveyor Belt는 물건을 하역하는 데 사용하며, 로봇이 목적지에 도착하면 모터를 동작시킨다. DC Motor Driver 2A L298N 보드는 Raspberry Pi를 통해 Conveyor Belt의 Motor를 제어하기 위한 드라이버이다.

2.2.2 로봇 서버 구성

로봇 서버 환경 구축을 위해 Ubuntu 20.04가 설치된 PC에 ROS Noetic 버전의 로봇 운영체제[18]를 설치하고 ROS 관련 패키지 및 TurtleBot3 관련 패키지[19]를 설치했다. 지도를 기반으로, 사용자가 지정한 특정 위치로 자율적으로 이동하는 것을 Navigation이라고 하며 Android 애플리케이션과의 연동을 통해 자동으로 동작시키기 위해 Navigation 자동화 프로그램을 개발했다[20]. Python으로 개발한 자동화 프로그램은 ROS에서 지원하는 rospy[21], 로봇의 움직임을 제어하는 패키지 (move_base_msgs, actionlib, geometry_msgs) 및 소켓 통신을 활용하여 개발했다. 애플리케이션으로부터 배송 요청 메시지를 받으면 목적지를 설정하고 자동으로 Navigation을 수행한 뒤 목적지에 도착하면 Conveyor Belt 동작을 제어하고, 물건의 하역이 완료되면 출발 위치로 돌아오도록 프로그램을 구현했다.

2.2.3 로봇 소프트웨어 구성

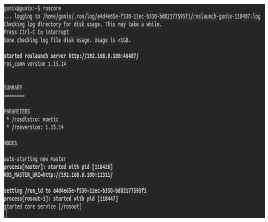
로봇 소프트웨어 환경 구축을 위해 Raspberry Pi에 ROS Noetic 버전의 로봇 운영체제가 설치된 Ubuntu 20.04 이미지를[17] 사용했다. Raspberry Pi는 고정 IP를 사용하도록 설정했으며, 별도의 환경 설정 파일을 통해 서버의 IP 주소를 등록했다[22,23]. LiDAR Sensor에서 수집하는 정보를 수신하기 위해 Raspberry Pi에 YDLIDAR사의 라이더 센서 드라이버 및 YDLIDAR-SDK[24,25]를 설치했다. 다음으로, Conveyor Belt를 제어하기 위한 프로그램을 개발했다[26]. 해당 프로그램은 Conveyor Belt 실행 메시지를 수신하면 벨트를 동작시키는 기능을 수행하며, 로봇이 목적지에 도달하면 로봇 서버로 도착 메시지를 보내고 로봇 서버는 로봇으로 Conveyor Belt 실행 메시지를 보낸다. 이때 해당 메시지를 받는 프로그램은 Python의 Socket과 모터를 제어하기 위한 gpiozero.Motor 패키지를 활용하여 개발했다.

2.2.4 로봇 환경 테스트

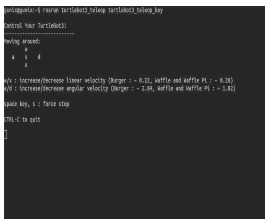
앞선 로봇 소프트웨어 환경 구성이 정상적으로 설정되었는지를 확인하기 위한 테스트를 수행했다. 그림 4(a)는 'roscore' 명령어를 실행한 화면으로, roscore는 ROS에서 모든 노드들의 메시지를 전달하고 관리하는 중앙노드이며 ROS 기반으로 작성된 프로그램은 roscore를 통해 다른 노드로 메시지를 전달한다. 그림 4(b)는 'rosrun turtlebot3_teleop turtlebot3_teleop_key' 명령어를 입력하여 원격 로봇 이동 제어 프로그램을 실행한 화면이다. 그림 4(c)는 SSH (Secure Shell)를 이용해 로봇에 접속한 후 'roslaunch turtlebot3_bringup turtlebot3_robot.launch' 명령어를 입력한 결과 화면으로, TurtleBot3 기반 패키지 노드와의 통신을 가능케 한다.

2.2.5 로봇 지도 구성

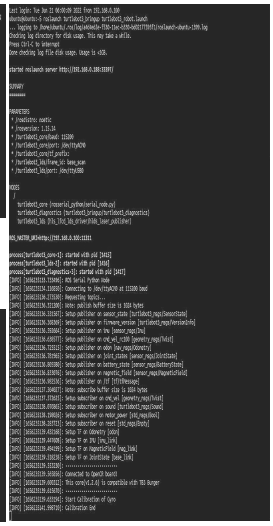
로봇이 자율주행하기 위해서 운용지역의 지도가 필요하다. 라이더 센서로 임의의 공간에서 지도를 생성



(a) 코어 모듈
(a) Core module



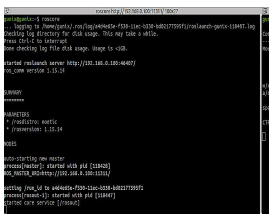
(b) 원격제어 모듈
(b) Remote control module



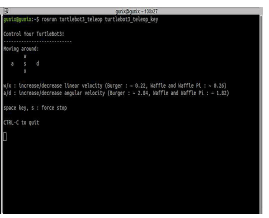
(c) 로봇 실행 모듈
(c) Robot operation module

그림 4. 로봇 동작 테스트 화면
Fig. 4. Robot operation test screenshots

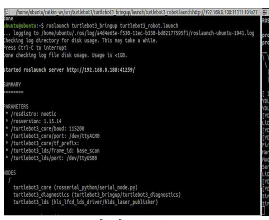
하는 기법을 Mapping이라고 하며 Mapping을 통해 로봇을 운용할 지역의 지도를 형성한다. ROS에서 지원하는 시각화 도구인 RViz[27]와 TurtleBot3 패키지들을 사용하여 지도를 구성하였다. 이를 위해, 그림 5와 같이 총 4개의 Terminal을 실행했고, 그림 5(d)는 'roslaunch ydlidar_ros_driver X4.launch' 명령어를 통해 YDLIDAR 사의 X4 모델 LiDAR 드라이버를 실행한 화면이다. 추가적으로 'roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods



(a) 코어 모듈



(b) 원격제어 모듈



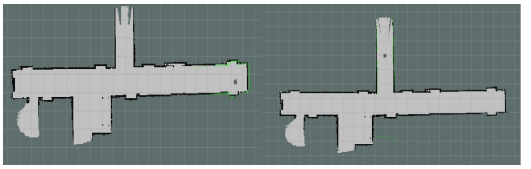
(c) 로봇 실행 모듈

(d) 라이더센서 드라이버

그림 5. 로봇 지도 매핑 구성
Fig. 5. Robot mapping configuration

(a) 지도 형성 초기
(a) Mapping: early stage

(b) 지도 형성 진행
(b) Mapping in progress



(c) 지도 형성 후반부
(c) Mapping: later stage

(d) 지도 형성 완료
(d) Mapping completed

그림 6. 로봇 지도 형성 과정
Fig. 6. Robot map configuration

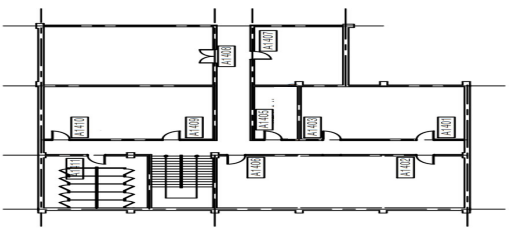


그림 7. 공학관 4층 평면도
Fig. 7. Floor plan of the 4th floor, Engr. Bldg.

:= gmapping' 명령어로 RViz 기반의 지도 Mapping 프로그램을 실행시킨다.

이후, 그림 6(a)와 같이 주변 지형지물을 기반으로 라이더 센서가 파악한 정보를 시각화할 수 있으며, 그림 6(a)를 시작으로 로봇의 이동에 따라 그림 6(b), 그림 6(c), 그림 6(d) 순으로 지도가 생성된다. 그림 7에 표시된 실내 환경에서 테스트 및 실험을 진행했고, 그림 5(b) 터미널을 통해 키보드 입력으로 로봇을 조종하여 지도를 획득했다. 최종 완성된 지도는 그림 8과 같다.

2.3 사용자 애플리케이션 및 서버 구성

본 논문에서 제안하는 라스트 마일 배송 시스템에서는 물건을 배송하는 사람이 원격으로 로봇에게 배송 명령을 전달할 수 있고, 이를 위해 안드로이드 애플리케이션을 개발하였다. 애플리케이션은 Android Studio 4.1 환경에서 Java 언어로 구현하였다.

애플리케이션에서 운송장 사진을 촬영하면 딥 러닝 기반 OCR을 사용해 사진으로부터 주소 데이터를 추출하고 해당 정보를 로봇 서버로 전송하여 배송 명령

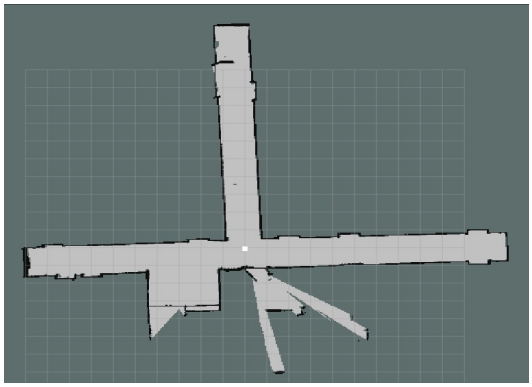


그림 8. 완성된 지도
Fig. 8. Completed map

을 내릴 수 있도록 구현하였다. 또한, 배송이 완료되면 수취인에게 배송 완료 메시지가 전송될 수 있도록 구현하였다. 애플리케이션의 전체 흐름도는 그림 9와 같다.

그림 10은 애플리케이션을 구동하는 화면을 캡처한 것이다. 처음 애플리케이션을 실행하면 그림 10(a) 화면을 거쳐 그림 10(b) 화면으로 이동한다. 그림 10(b) 화면에서는 운송장 촬영을 위한 간단한 설명이 사용자에게 제공된다. 그림 10(b)의 좌측 상단에 위치한 메뉴 버튼을 통해 로봇 서버의 IP 주소를 지정할 수 있고 그림 10(b) 하단의 “사진 촬영 하기” 버튼을 통해 애플리케이션을 실행한 단말기 내 저장소에 저장된 사진을 선택하거나 단말기의 카메라 기능을 통

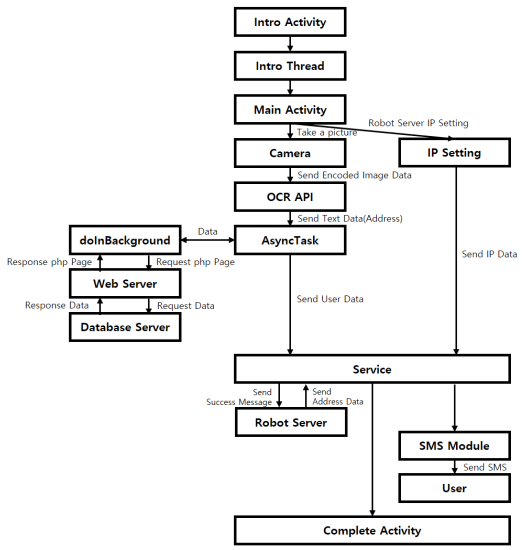
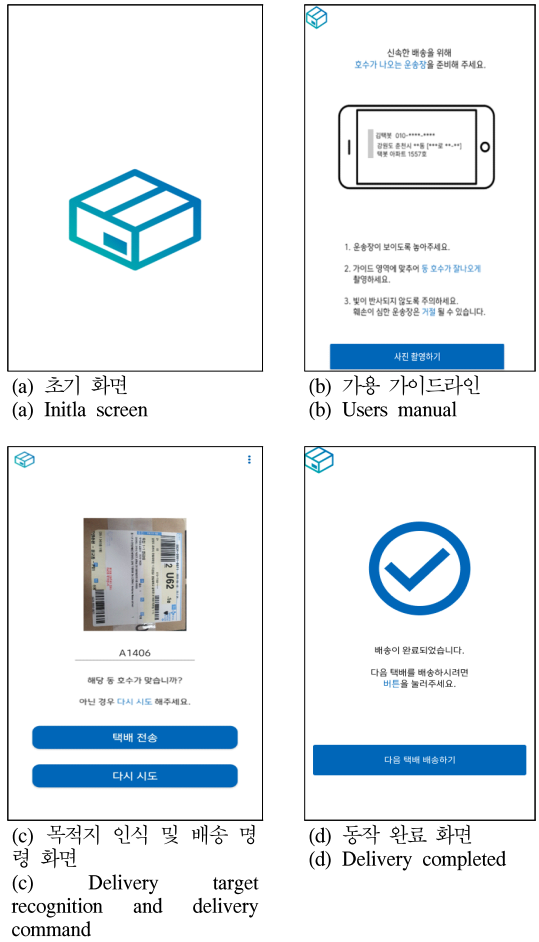


그림 9. 애플리케이션 전체 흐름도
Fig. 9. Application flow chart



(a) 초기 화면
(a) Initla screen

(b) 가용 가이드라인
(b) Users manual

(c) 목적지 인식 및 배송 명령 화면
(c) Delivery target recognition and delivery command

(d) 동작 완료 화면
(d) Delivery completed

그림 10. 애플리케이션 화면
Fig. 10. Application screen

해 송장 사진을 촬영할 수 있다.

운송장 사진을 선택/촬영하면 그림 10(c) 화면으로 전환된다. 해당화면은 사진에서 추출한 주소 데이터와 간단한 설명, 택배 전송 버튼과 다시 시도 버튼으로 구성된다. 화면에서 보여지는 주소 데이터에 오류가 있을 경우 직접 수정이 가능하도록 Android Studio에서 EditText[28] 기능을 사용해 구현했다. 그림 10(c)에서 ‘택배 전송’ 버튼을 누르면 추출한 주소 데이터를 로봇 서버로 전송한다. 그 다음, 주소 데이터를 통해 애플리케이션 데이터베이스 서버에 접근해 송장에서 확인할 수 없는 수취인의 이름과 수취인의 전화 번호를 획득하고, 로봇의 자율주행을 통해 물건이 정상적으로 수취인에게 도착한 경우 수취인에게 배송 완료 메시지를 전송한다.

2.3.1 딥러닝 기반 OCR을 사용한 이미지 텍스트 추출

그림 10(b)에서 '사진 촬영 하기' 버튼을 통해 획득한 이미지 파일로부터의 텍스트 인식 및 추출은 Google Cloud Vision AI를 사용했다.[15] 제공되는 API를 사용하면 해당 이미지에 포함된 문자를 텍스트로 추출하여 JSON 포맷으로 반환한다. 이미지에서 추출한 JSON 포맷 데이터를 String 포맷 데이터로 변경 후 배송에 필요한 주소 문자열 데이터만 추출한다.

2.3.2 데이터베이스 서버 구축 및 연동

운송장에서 주소 데이터를 추출한 후, 이를 통해 추가적인 수취인 정보를 얻기 위해 수취인의 이름, 전화번호 등과 같이 송장에서 확인할 수 없는 정보는 별도의 데이터베이스를 통해 저장 및 관리한다. 확장성 있는 데이터베이스 서버를 구축하기 위해 AWS (Amazon Web Services)의 EC2 (Elastic Compute Cloud)[29]를 사용하여 Ubuntu Server 20.04 버전의 운영체제로 서버를 구축하였고 데이터베이스는 MySQL 8.0.29 버전을 사용했다. 수취인의 정보를 저장하기 위한 User Table을 그림 11과 같이 설계하였다.

실제 애플리케이션에서 데이터베이스에 접근하는 동작은 비동기(Asynchronous) 작업으로 수행되며 이를 위해 AsyncTask[30]를 사용했다. Main Thread가 생성된 직후, 데이터베이스 접근을 위해 새로운 Thread를 생성하고 해당 Thread에서 백그라운드 작업(doinBackground)을 통해 데이터베이스에 접근한다. 백그라운드 작업을 통해 웹 서버에 php 페이지를 요청하고 데이터베이스 접근 결과를 수신하여 이를 Main Thread로 반환한다. 이 과정을 통해 애플리케이션은 운송장에 표기되지 않은 수취인의 정보를 얻게 된다.

```
mysql> SELECT * FROM USER;
+----+-----------+---------+---------------+
| id | name      | address | number        |
+----+-----------+---------+---------------+
|  1 | 홍길동    | A1111   | 010-1111-2222 |
|  2 | 김철수    | A2222   | 010-3333-4444 |
|  3 | 김영희    | A1401   | 010-1234-1234 |
|  4 | 홍수림    | A1402   | 010-0000-0000 |
|  5 | 박상철    | A1403   | 010-5555-5555 |
|  6 | 박종건    | A1406   | 010-6666-6666 |
|  7 | 이현아    | A1407   | 010-7777-7777 |
+----+-----------+---------+---------------+
7 rows in set (0.00 sec)
```

그림 11. 데이터베이스에 정의한 'User' 테이블
Fig. 11. Defined 'User' table

2.3.3 로봇 서버와의 통신 및 SMS 전송

그림 10(c)에서 '배송 하기' 버튼을 선택하면 Main Thread는 Service 로직을 실행하게 되고(참고: 그림 9) 추출한 주소 데이터와 2.3.2절에서 얻은 수신자 정보를 Service에 전달한다. 여기서, 서버의 Service 로직은 로봇 서버와의 통신뿐 아니라 수취인에게 배송 완료 SMS를 전송하는 작업을 수행한다. 로봇 서버와의 통신은 소켓 통신을 통해 구현했다. 그림 10(b) 화면에서 미리 설정한 로봇 서버의 IP주소와 Port 번호를 통해 Socket을 생성하고 연결 요청을 보낸 후 연결이 수립되면 수취인 주소 문자열 데이터를 전송하고 배송 완료를 의미하는 성공 메시지가 수신될 때까지 대기한다. 로봇이 배송을 성공적으로 마치고 로봇 서버가 성공 메시지를 애플리케이션으로 전송해 애플리케이션이 이를 수신하게 되면 SmsManager[31] 객체를 통해 수취인 전화번호로 배송 완료 SMS를 발신한다.

위의 모든 작업이 수행되면 그림 10(d) 화면으로 전환된다. 해당화면은 수취인에게 배송 완료 SMS를 전송함과 동시에 물건이 성공적으로 배송을 완료했음을 알린다. 추가로 그림 10(d) 하단의 '다음 택배 배송하기' 버튼을 클릭하면 그림 10(b) 화면으로 돌아오게 되고 새로운 배송을 시작할 수 있다.

Ⅲ. 실 험

본 논문에서 제시하는 시스템의 검증을 위해, 본교 공학관 4층 일부를 실험 환경으로 사용하였다. 배송 물품은 16cm x 12cm x 9cm 크기의 상자이고, 무게는 약 2kg이다. 배송 기사의 휴대폰은 SAMSUNG Galaxy S10 5G (Android 12)를 사용했고 로봇 서버는 ROS Noetic 버전이 설치되어있는 Ubuntu 20.04 환경의 노트북, 애플리케이션의 데이터베이스 서버는 Amazon AWS EC2를 사용했다. 그림 12는 본교 공학관 A1407호를 배송지로 하여 택배를 전달하는 전체적인 과정을 나타낸다.

그림 12(a)는 사용자가 애플리케이션을 이용해 운송장을 촬영하는 과정을 나타낸다. 촬영한 사진을 딥러닝 기반 OCR을 사용해 배송 목적지 주소를 자동으로 추출하고, 추출된 정보는 Wi-Fi 또는 이동 통신망 연결을 통해 로봇 서버로 전송된다.

그림 12(b)와 12(c)는 로봇이 이동하는 과정을 나타낸다. 로봇은 SLAM 기술을 이용하여 완성된 지도를 기반으로 지도의 특정 좌표를 목적지로 설정하여 자율적으로 이동한다. 그림 12(b)와 같이 로봇의 진행 경로에 장애물 또는 복잡한 지형지물이 존재하는 경

(a) 송장 촬영 (b) 배송 중 코너 감지 및 회전
(a) Taking a photograph of the invoice (b) Corner detection and turning during delivery

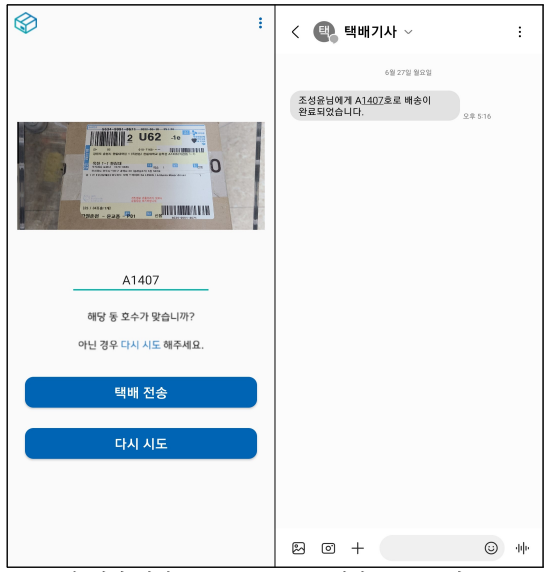
(c) 목적지 도착 및 하역을 위한 차체 회전 (d) 컨베이어 벨트 구동을 통한 물품 하역
(c) Arrived at the target address and rotating to drop the parcel (d) Drop the parcel by operating the conveyor belt

그림 12. 실험의 전체적인 과정
Fig. 12. Complete process of an experiment

우, 로봇의 Lidar 센서에서 수집한 정보를 바탕으로 안전하게 회피 또는 방향을 전환하여 진행 가능한 경로를 계산한 뒤 계속 이동한다. 로봇은 목적지에 도착한 이후 배송을 위해 그림 12(c)와 같이 로봇의 후면이 배송지의 문을 바라보도록 로봇의 차체를 회전시킨 후 정지한다.

그림 12.(d)는 로봇의 물품 보관함에서 배송지로 물품을 하역하는 과정을 나타낸다. 로봇은 목적지 도착 후 컨베이어 벨트를 이용해 배송 물품을 캐리어에서 떨어뜨린다. 모든 배송 진행 과정이 완료된 이후, 로봇은 출발지로 복귀하고 다음 배송을 위해 준비상태로 전환한다.

그림 13(a)는 배송 정보 추출이 완료된 애플리케이션 화면을 나타낸다. 딥 러닝 기반 OCR을 이용해 배송 목적지 정보를 추출하는데, 추출한 정보가 정확하지 않을 경우 화면 중앙의 텍스트 영역을 통한 정보 수정이 가능하다. 목적지 정보 확인 후, '택배 전송' 버튼을 클릭하면 추출된 주소 정보를 로봇 서버로 전



(a) 송장 인식 화면 (b) 수취인 SMS 수신
(a) Recognizing the address written on the invoice (b) Receiver is notified of the delivery by SMS

그림 13. 사용자 애플리케이션과 수취인 메시지 수신 결과
Fig. 13. Application screen and receiver's screen

송한다. 주소 정보를 수신한 로봇 서버는 Navigation 동작을 명령하고 로봇은 목적지로 자율 운행한다.

그림 13(b)는 배송 완료 후 수취인이 수신한 SMS 메시지를 나타낸다. 배송이 정상적으로 완료된 이후 로봇에서 배달 성공 메시지를 로봇 서버를 통해 애플리케이션에 전송하게 되고 애플리케이션에서 데이터베이스 서버에 저장되어있는 사용자 정보를 통해 안드로이드에서 제공하는 SmsManager 라이브러리를 이용하여 수취인에게 배송 완료 메시지를 그림 13(b)와 같이 전송한다.

Ⅳ. 결 론

본 논문에서는 SLAM 기술과 딥 러닝 비전 기반의 OCR을 사용한 라스트 마일 배송 자동화 시스템을 제안 및 구현 결과를 소개한다. TurtleBot3, Raspberry Pi 4, YDLIDAR X4, OpenCR 및 DC Motor 기반의 Conveyor Belt로 하드웨어를 구성하고 ROS와 Python을 이용해 로봇 서버와 각종 제어기능을 구현했다. 애플리케이션의 데이터베이스 서버는 AWS EC2를 통한 Ubuntu Server와 MySQL, nginx, php를 사용해 구축했고 Google Vision API를 활용한 딥 러닝 기반 OCR과 Android Studio, Java를 통해 제작된 애플리케이션으로 로봇의 동작을 제어했다. 제안하는

시스템은 애플리케이션에서 촬영된 실제 송장 사진으로부터 배송 목적지 정보를 추출하고 로봇 서버를 통해 로봇으로 배송을 명령한다. 로봇은 수신한 정보를 기반으로 자율 주행을 통해 물건을 배송하고 배송이 완료되면 애플리케이션에서 데이터베이스 서버에 저장된 사용자 정보를 통해 도착 확인 메시지를 성공적으로 전송하는 것을 확인했다.

논문에서 제안하는 시스템은 배송하는 사람이 직접 수취인의 문 앞까지 이동할 필요 없이, 정해진 장소에 위치한 로봇에 물건을 적재하면 애플리케이션을 통해 자율적으로 목적지까지 배송하기 때문에 인력 의존도가 높은 기존의 라스트 마일 배송 방식에 비해 시간·비용 효율적인 자동화된 배송이 가능하다. 향후, Raspberry Pi에 카메라 모듈을 탑재하여 수취인에게 배송 완료 메시지를 보낼 때 물건이 놓여 있는 수취인의 문 앞 사진과 함께 전송하는 기능, 무선 충전 모듈을 이용하여 배송 완료 후 스스로 배터리를 충전하는 기능, 동시에 다수의 물품 배송 명령을 처리하는 기능, 배송 중 장애 발생 시 자가 진단을 통해 정상 상태로의 자동화된 복귀 기능 등 라스트 마일 배송 시스템의 완전 자동화를 위한 추가적인 연구를 진행할 계획이다.

References

[1] Statistics Korea, *Transaction value of online shopping mall*(2022), Retrieved Jun. 24, 2022, from https://kosis.kr/statHtml/statHtml.do?orgId=101&tblId=DT_1KE10071

[2] National Logistics Information Center, *Household logistics statistics*(2021), Retrieved Jun. 24, 2022, from https://nlic.go.kr/nlic/parcelServiceLogist ics.action

[3] B. Paik, Coopang logistics center, Robot and cooperation... release of the automation technology(2022), Retrieved Jun. 24, 2022, from https://zdnet.co.kr/view/?no=20220208104838

[4] S. Lee, CJ Logistics, Contracted all fields in e-commerce logistics(2017), Retrieved Jun. 23, 2022, from http://www.techholic.co.kr/news/articleView.html?idxno=171031#rs

[5] C. Jung, Delivery robots... 'Last Mile' revolution(2021), Retrieved Jun. 23, 2022, from https:// magazine.hankyung.com/business/article/202105142612b

[6] I. Shin, Part 1. Continual growth of 'Last Mile' market(2021), Retrieved Jun. 23, 2022, from https://www.klnews.co.kr/news/articleView.html?idxno=122698

[7] G. Song, 'Last Mile'... Naver-Kakao made a full-fledged project(2022), Retrieved Jun. 24, 2022, from https://www.sisaweek.com/news/articleView.html?idxno=151519

[8] H. Lee, 'Last Mile'... Global enterprises' heated competition in delivery robots(2021), Retrieved Jun. 24, 2022, from https://zdnet.co.kr/view/?no=20210603160016

[9] Innopolis, *Delivery robot and logistics robot market*(2020), Retrieved Jun. 24, 2022, from https://www.innopolis.or.kr

[10] M.-C. Park, K.-H. Kim, and H.-S. Jeon, "Apartment-type self-driving courier delivery robot," in *Proc. Korean Soc. Comput. Inf. Conf.*, vol. 30, no. 1, pp. 301-302, Jan. 2022.

[11] Z. Dilkashbek and S. Choi, "Lidar based SLAM and navigation of indoor mobile robot," *2021 Autumn Annu. Conf. IEIE*, pp. 1080-1083, Nov. 2021.

[12] M. Milford and G. Wyeth, "Hybrid robot control and SLAM for persistent navigation and mapping," *Robotics and Autonomous Syst.*, vol 58, no. 9, pp. 1096-1104, Sep. 2010. (https://doi.org/10.1016/j.robot.2010.05.004)

[13] S. Saeedi, et al., "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality," in *Proc. IEEE*, vol. 106, no. 11, Nov. 2018. (https://doi.org/10.1109/JPROC.2018.2856739)

[14] *Robotis Turtlebot3*, Retrieved Feb. 25, 2022, from https://www.robotis.com/shop/item.php?it_id=901-0118-202

[15] *Google Cloud Vision API*, Retrieved Apr. 15, 2022, from https://cloud.google.com/vision/

[16] *YDLIDAR X4*, Retrieved Apr. 20, 2022, from https://www.ydlidar.com/products/view/5.html

[17] *Turtlebot3 e-Manual*, Retrieved, Mar. 5, 2022, from https://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#download-turtlebot3-sbc-image-2

[18] Qboticslabs, *ROS Noetic*(2022), Retrieved May

30, from http://wiki.ros.org/noetic/Installation/Ubuntu
[19] *Turtlebot3 PC Setup*, Retrieved Mar. 5, 2022, from https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/#pc-setup
[20] GEONheong, *Listen_appServer.py*(2022), Retrieved May 16, 2022, from https://github.com/DeliveryBotCapstone/DeliveryBot/blob/main/src/Robot_Server/delivery_action/listen_appServer.py
[21] GvdHoorn, *Rospy*(2017), Retrieved Apr. 15, 2022, from http://wiki.ros.org/rospy
[22] *Turtlebot3 SBC Setup*, Retrieved Mar. 5, 2022, from https:// emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#configure-the-wifi-network-setting-1
[23] *Turtlebot3 SBC Setup*, Retrieved Mar. 5, 2022, from https://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#ros-network-configuration-1
[24] Eaibot, *How_to_build_and_install.md*(2022), Retrieved Apr. 24, 2022, from https://github.com/YDLIDAR/YDLidar-SDK/blob/master/doc/howto/how_to_build_and_install.md
[25] Zhanyiaini, *README.md*(2022), Retrieved Apr. 24, 2022, from https://github.com/YDLIDAR/ydlidar_ros_driver/blob/master/README.md
[26] GEONheong, *Listen_formortor.py*(2022), Retrieved May 16, 2022, from https://github.com/DeliveryBotCapstone/DeliveryBot/blob/main/src/Robot_SW/listen_formotor.py
[27] WilliamWoodall, *Rviz*(2018), Retrieved Apr. 20, 2022, from http://wiki.ros.org/rviz
[28] *Android Developers EditText API*, Retrieved Apr. 15, 2022, from https://developer.android.com/reference/android/widget/EditText
[29] *Amazon Web Service*, Retrieved Apr. 10, 2022, from https://aws.amazon.com/ko/ec2/
[30] *Android Developers AsyncTask API*, Retrieved Apr. 17, 2022, from https://developer.android.com/reference/android/os/AsyncTask
[31] *Android Developers SmsManager API*, Retrieved Apr. 20, 2022, from https://developer.android.com/reference/android/telephony/SmsManager

박 건 형 (Geon-hyeong Park)

2017년~현재 : 한림대학교 정보과학대학 소프트웨어학부 스마트IoT 전공
<관심분야> 정보통신, 로봇 제어, 딥러닝, 강화학습

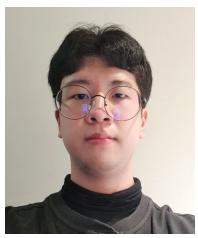
조 성 윤 (Seong-yun Cho)

2017년~현재 : 한림대학교 정보과학대학 소프트웨어학부 빅데이터 전공
<관심분야> 정보통신, 네트워크, 딥러닝, 자율주행

신 재 성 (Jae-seong Shin)

2017년~현재 : 한림대학교 공과대학 컴퓨터공학과
<관심분야> 정보통신, 클라우드 컴퓨팅, 딥러닝, 자율주행

박 종 건 (Jong-gun Park)



2017년~현재 : 한림대학교 정보과학대학 소프트웨어학부 빅데이터 전공
<관심분야> 정보통신, 웹 백엔드, 딥러닝, 자율주행

양 현 규 (Hyun-kyu Yang)

2017년~현재 : 한림대학교 정보과학대학 소프트웨어학부 빅데이터 전공

<관심분야> 정보통신, 자연어처리, 딥러닝, 자율주행

김 태 운 (Taewoon Kim)

2008년 : 부산대학교 정보컴퓨터공학과 학사

2010년 : 광주과학기술원 정보통신 공학과 석사

2018년 : 아이오와주립대학 컴퓨터 공학과 박사

2018년~2022년 : 한림대학교 소프트웨어학부 조교수

2022년~현재 : 부산대학교 컴퓨터공학부 조교수

<관심분야> 무선 통신 및 네트워크, 클라우드/엣지 컴퓨팅, 수치 최적화, 심층 강화학습 등

[ORCID:0000-0002-7811-5022]