Journal of the KNST

2024; Vol.7, No.2; pp.093-106

ISSN: 2635-4926

https://doi.org/10.31818/JKNST.2024.6.7.2.93

Check for updates

Received: 2024/04/08
Revised: 2024/04/19
Accepted: 2024/05/13
Published: 2024/06/30

Corresponding Author:

Kyeongmo Kang

Dept. of Mechanical System Engineering, Republic of Korea Naval Academy
1 Jungwon-ro, Jinhae-gu, Changwon-si, Gyungsangnam-do, 51704, Republic of Korea

Tel: +82-55-907-5316

E-mail: kmkang@navy.ac.kr

Indoor Autonomous Surveillance Robot Based on RGB-D Sensor

RGB-D 센서 기반 실내 자율 감시 로봇

Kyeongmo Kang[1*], Won-jong Kim[2]

[1]LT, ROK Navy/Instructor, Department of Mechanical System Engineering, Republic of Korea Naval Academy
[2]Associate professor, Dept. of Mechanical Engineering, Texas A&M University

강경모[1*], 김원종[2]

[1]해군 대위/해군사관학교 기계시스템공학과 교관
[2]텍사스 A&M 대학교 기계공학과 부교수

Abstract

This article presents an autonomous surveillance robot with an Red-Green-Blue-Depth (RGB-D) sensor. The robot incorporates Simultaneous Localization and Mapping (SLAM), autonomous patrol, face recognition, and human tracking. Based on mathematical modeling, the control system of the robot is designed with proportional-integral-differential (PID) controllers. Autonomous patrol is achieved through the control system and Robot Operating System (ROS) Navigation Stack. A Convolutional Neural Network (CNN) model is employed for face recognition. For human tracking, a position-control system is developed based on skeleton tracking. The integration of these functions into a single system results in a low-cost surveillance robot, which is tested in real-life environments.

본 논문에서는 저가 RGB-D 센서 기반의 실내 자율 감시 로봇을 소개한다. 로봇에는 SLAM, 자율정찰, 얼굴인식, 사람추적 기능이 통합되어 탑재되었으며, 제어시스템은 수학적 모델링과 PID 제어기를 기반으로 설계되었다. ROS Navigation Stack을 활용하여 자율정찰 기능을 개발했으며, CNN 모델을 통해 얼굴인식 기능을 구현했다. 또한, 골격추적 기반의 위치제어시스템을 개발하여 사람추적 기능을 구현했다. 기능들을 단일 시스템으로 통합하여 저비용 감시 로봇을 개발했고, 실제 생활 환경에서 실험했다.

Keywords

자율정찰(Autonomous Patrol),
디지털제어(Digital Control),
RGB-D 센서(RGB-D Sensor), 로봇운영체계(ROS),
감시로봇(Surveillance Robot)

Acknowledgement

본 논문은 해군사관학교 해양연구소 학술연구과제 연구비의 지원으로 수행된 연구임.

1. Introduction

Today, the use of security devices to enhance safety and prevent crime is widespread and a surveillance camera is the most popular device. However, fixed ceiling-mounted cameras often suffer from blind spots and limitations in person recognition. Surveillance robots provide an alternative solution, offering various advantages such as comprehensive area coverage, optimal viewing angles, and the ability to incorporate additional functionalities.

In past decades, surveillance robots were developed by combining surveillance systems and mobile robots. Advancements in autonomous navigation technology enabled the development of surveillance robots for autonomous patrol, building mapping, intruder tracking, and suspicious behavior recognition[1,2]. These robots significantly reduce personnel risks in dangerous environments.

The disadvantage of surveillance robots is the high cost. Surveillance robots generally employ various sensors tailored to their purpose, including a Red-Green-Blue (RGB) camera, RGB-D sensor, Light Detection and Ranging (LiDAR), Inertial Measurement Unit (IMU), and Global Positioning System (GPS). The more these expensive sensors are used, the higher the cost required for robot development. To reduce the cost, we studied developing a compact surveillance robot by replacing various sensors with only one sensor.

Among various sensors, we focused on RGB-D sensors. They provide RGB images along with pixel distance information using infrared. Due to the advantage of providing RGB images and distances simultaneously, it can be utilized in various research areas. RTAB-Map uses an RGB-D sensor for 3D SLAM and visual odometry[3,4]. RGB-D sensors were used for face recognition[5,6]. In [7,8], they were used to detect humans and recognize motion in 3D. Microsoft developed an RGB-D sensor for motion-sensing games[9]. Almeida et al. used an RGB-D sensor to recognize sign language[10].

In this research, we use Kinect V2 sensor, a low-cost RGB-D sensor developed by Microsoft, to develop an autonomous surveillance robot. The advantage of using the sensor is that multiple functions can be implemented without the need for additional sensors. Unlike existing surveillance robots that rely on combinations of high-cost sensors, this article presents a novel autonomous surveillance robot integrating SLAM, autonomous patrol, face recognition, and human tracking functions with a single Kinect V2 sensor.

We made a comparison between 2D SLAM of Gmapping[11] and 3D SLAM of RTAB-Map[3,4] to find the suitable SLAM technique for the RGB-D sensor-based surveillance robot. To implement autonomous patrol, we developed the robot's control system based on mathematical modeling and used ROS Navigation Stack[12] with Dijkstra's algorithm[13] and Dynamic Window Approach (DWA)[14]. We utilized a CNN model[15] to recognize faces from RGB images. For human tracking, we designed a position control system of the robot using depth image based on skeleton tracking.

This approach makes it possible to develop low-cost compact a surveillance robot by replacing multiple sensors with a single Kinect V2 sensor. We verify the performance of the robot in a real-life environment and present the process of developing a new system by integrating various techniques.

This paper consists of six sections. Section 2 introduces the hardware design and components of the surveillance robot. In Section 3, the robot's mathematical model is analyzed and its control system is presented based on the robot's hardware design. Section 4 covers the software design to implement SLAM, autonomous patrol, human tracking, and face recognition in the robot as a single system. In Section 5, the experimental results of each function in a real-life environment are given. Section 6 presents the conclusions of this study.

2. Hardware design

As a mobile platform, a differential-drive robot was developed, as shown in Fig. 1. The robot has two wheels with independent actuators at the front and a caster wheel at the rear. The robot moves based on the velocity difference between the front wheels. Using their velocity difference, it can change the direction easily, even in place.

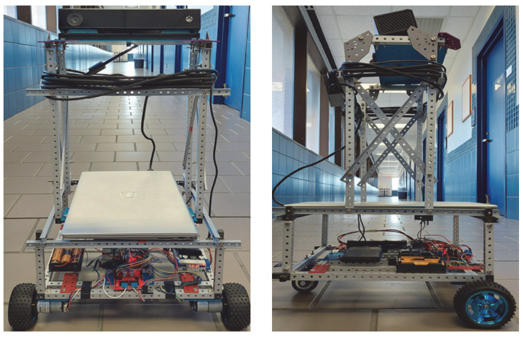


Fig. 1. Differential-drive robot proposed in this study

Fig. 2 shows the hardware connection of the robot. Three batteries power the Kinect sensor, Raspberry Pi, and DC motors. Voltage boosters ensure a stable power supply for the Kinect sensor and motor driver. Wireless Local Area Network (WLAN) facilitates communication between the laptop and Raspberry Pi. The Raspberry Pi controls the motors through the General-Purpose Input/Output (GPIO) pins connected to the motor driver.

Fig. 3 shows the hardware setup and communications. The laptop runs the main software code for SLAM, autonomous patrol, face recognition, and human tracking. The Raspberry Pi calculates and controls the robot's velocity. In the user interface part, the remote computer monitors the robot and sends a navigation goal. The user interface provides live RGB-D images, the robot's current position/orientation, skeleton tracking results, and face recognition results.

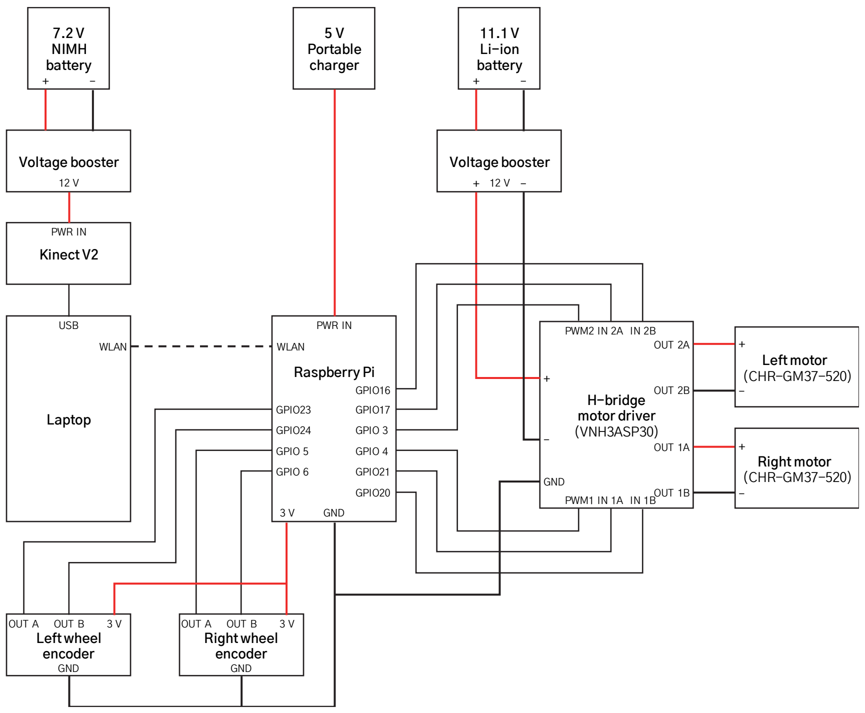

7.2 V NIMH battery
5 V Portable charger
11.1 V Li-ion battery
Voltage booster 12 V
Voltage booster 12 V
PWR IN Kinect V2
USB WLAN Laptop
PWR IN WLAN Raspberry Pi
GPIO16 GPIO23 GPIO17 GPIO24 GPIO 3 GPIO 5 GPIO 4 GPIO 6 GPIO21 GPIO20 3 V GND
PWM2 IN 2A IN 2B OUT 2A OUT 2B OUT 1A OUT 1B GND PWM1 IN 1A IN 1B
H-bridge motor driver (VNH3ASP30)
Left motor (CHR-GM37-520)
Right motor (CHR-GM37-520)
OUT A OUT B 3 V Left wheel encoder GND
OUT A OUT B 3 V Right wheel encoder GND

Fig. 2. Hardware circuit diagram

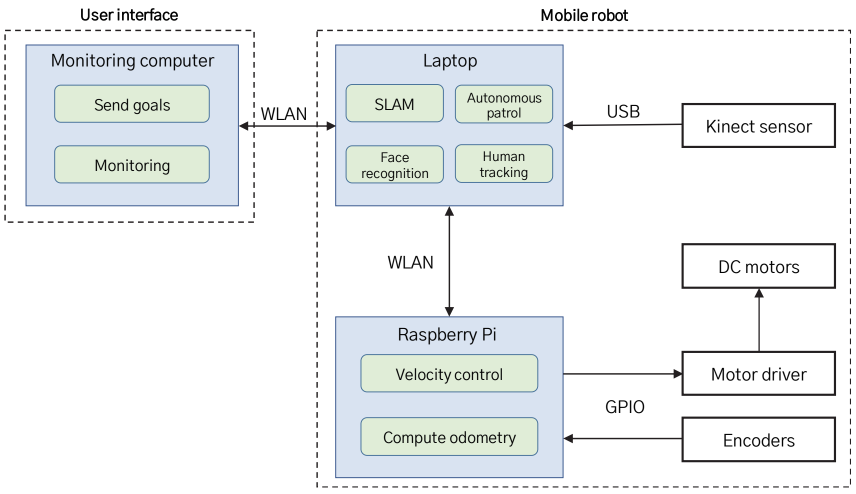


Fig. 3. Hardware setup and communications

3. Mathematical modeling and control system

3.1. Kinematic modeling

In this research, we developed the robot's kinematic model in a global frame (X, Y) and a base-fixed frame (x, y) as shown in Fig. 4. The global frame is fixed to decide the absolute position of the mobile robot. On the other hand, the base-fixed frame is located at the center of the robot's front wheels and moves with the robot.

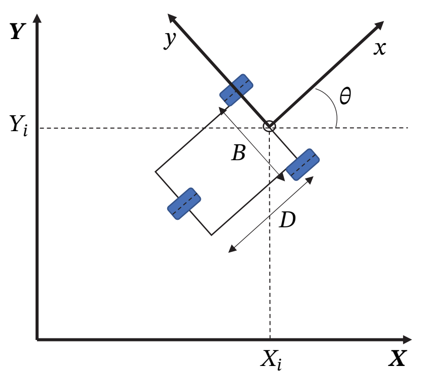


Fig. 4. Robot's global (X, Y) and base-fixed (x, y) frames

In the base-fixed frame, the robot's velocities can be obtained from the two front wheels' angular velocities.

$$\dot{x} = \frac{r}{2}(w_r + w_l), \quad \dot{y} = 0, \quad \dot{\theta} = \frac{r}{B}(w_r - w_l) \tag{1}$$

where $\dot{x}$ is the linear velocity in the longitudinal direction, $\dot{y}$ is the linear velocity in the lateral direction, $\dot{\theta}$ is the angular velocity, B is the distance between the two front wheels, r is the front wheels' radius, and w_r and w_l are their angular velocities. The differential-drive robot cannot move in the y-direction. Moving in the y-direction indicates that it slides in the lateral direction.

The robot's velocity in the global coordinates can be obtained from the velocity in the base-fixed coordinates.

$$\dot{X} = \dot{x}\cos\theta = \frac{r}{2}(w_r + w_l)\cos\theta \tag{2}$$

$$\dot{Y} = \dot{x}\sin\theta = \frac{r}{2}(w_r + w_l)\sin\theta \tag{3}$$

where $\dot{X}$ and $\dot{Y}$ are the robot's linear velocities in the global coordinates.

3.2. Dynamic modeling

In dynamic modeling, we focused on getting the robot's velocity according to the two front motors' torques in the base-fixed frame. The dynamic model of the robot can be analyzed by dividing it into linear and angular motions.

The dynamic equation in linear motion was developed.

$$\frac{d^2x}{dt^2} = \frac{1}{M}\left((f_r - f_{Frf}) + (f_l - f_{Flf}) - f_{Rf}\right) \tag{4}$$

$$V_x(s) = \frac{1}{Ms}\left((F_r - F_{Frf}) + (F_l - F_{Flf}) - F_{Rf}\right) \tag{5}$$

where M is the robot's mass, f_r and f_l are the forces from the front-right and front-left motors, f_{Frf}, f_{Flf}, and f_{Rf} are the friction forces from front-right, front-left, and rear wheels, and $V_x(s)$ is the robot's linear velocity in the Laplace domain. By substituting torques for the forces, the linear velocity according to the motors' torques is obtained.

$$V_x(s) = \frac{1}{Mrs}(T_r + T_l) - F_{xf} \tag{6}$$

$$F_{xf} = \frac{1}{Ms}\left(F_{Frf} + F_{Flf} + F_{Rf}\right) \tag{7}$$

where T_r and T_l represent the torques of front-right and front-left motors, and F_{xf} is the friction term in linear motion.

The dynamic equation of the robot in angular motion was developed.

$$\frac{d^2\theta}{dt^2} = \frac{B}{2I}\left(\left(f_r - f_{Frf}\right) - \left(f_l + f_{Flf}\right)\right) - \frac{Df_{Rf}}{I} \quad (8)$$

$$V_\theta(s) = \frac{B}{2Is}\left(\left(F_r - F_{Frf}\right) - \left(F_l + F_{Flf}\right)\right) - \frac{Df_{Rf}}{Is} \quad (9)$$

where D is the distance between the center of the front wheels and the rear wheel, and $V_\theta(s)$ is the robot's angular velocity. The angular velocity according to the torques is obtained by replacing the forces with torques.

$$V_\theta(s) = \frac{B}{2Irs}(T_r - T_l) - F_{\theta f} \quad (10)$$

$$F_{\theta f} = \frac{1}{2Is}\left(BF_{Frf} + BF_{Flf} + 2DF_{Rf}\right) \quad (11)$$

where $F_{\theta f}$ is the friction term in angular motion.

3.3. Control system design

Based on the developed mathematical models, we designed a velocity-control system of the robot as shown in Fig. 5. D_x and D_θ represent the linear-velocity and angular-velocities controllers. G_{Lx} and G_{Rx} are the transfer functions of the front-left and front-right motors in linear motion. $G_{L\theta}$ and $G_{R\theta}$ are the transfer functions of the front-left and front-right motors in angular motion.

From Eq. (6), the linear velocity of the robot can be calculated with the Pulse Width Modulation (PWM) torque constant K_P, the right motor PWM input P_R, and the left motor PWM input P_L.

$$V_x(s) = \frac{K_P(P_R + P_L)}{Mrs} - F_{xf} \quad (12)$$

The parameter K_P describes the linear relationship between the motor's torque and the PWM duty ratio, given that the input current is proportional to it.

In linear-velocity control, the robot is considered to move straight without rotation. For straight motion, the PWM inputs to both motors should be equal, $P_R = P_L$. Then, Eq. (12) can be written using the common PWM input P_{in} and the motors' linear-motion transfer function G_{Lx} and G_{Rx}.

$$V_x(s) = P_{in}(G_{Rx} + G_{Lx}) - F_{xf} \quad (13)$$

$$G_{Rx} = G_{Lx} = \frac{K_P}{Mrs} \quad (14)$$

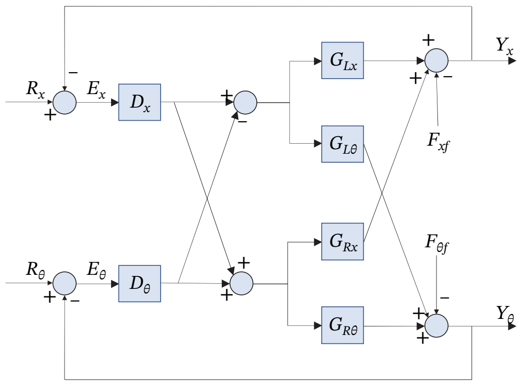


Fig. 5. Block diagram of the velocity control system

From Eq. (10), the angular velocity of the robot can be calculated with K_P, P_R, and P_L.

$$V_\theta(s) = \frac{BK_P(P_R - P_L)}{2Irs} - F_{\theta f} \quad (15)$$

In angular-velocity control, the robot is considered rotating about the center of the two front wheels without linear motion. That means the PWM inputs of the two front motors should have the same magnitude and opposite sign, $P_R = -P_L$. Then, Eq. (15) can be written with the common PWM input P_{in} and the motors' angular-motion transfer functions $G_{L\theta}$ and $G_{R\theta}$.

$$V_\theta(s) = P_{in}(G_{R\theta} - G_{L\theta}) - F_{\theta f} \quad (16)$$

$$G_{R\theta} = -G_{L\theta} = \frac{BK_P}{2Irs} \quad (17)$$

Based on the control system, we developed a PI controller D_x for linear-velocity control and a PID controller D_θ for angular-velocity control. The controllers were tuned using MATLAB Simulink.

$$D_x(s) = \frac{600s + 1200}{s} \tag{18}$$

$$D_\theta(s) = \frac{s^2 + 200s + 400}{s} \tag{19}$$

Tustin's method with a 30-Hz sampling frequency was used to implement them on the Raspberry Pi[16].

$$\begin{aligned} u_x[k] &= u_x[k-1] + 620e_x[k] \\ &\quad -580e_x[k-1] \end{aligned} \tag{20}$$

$$\begin{aligned} u_x[k] &= u_\theta[k-2] + 266e_\theta[k] \\ &\quad -106e_\theta[k-1] - 133e_\theta[k-2] \end{aligned} \tag{21}$$

where u_x and u_θ are the control inputs, and e_x and e_θ are the errors. Finally, the PWM inputs for each motor are calculated as Eq. (22).

$$\begin{aligned} P_R[k] &= u_x[k] + u_\theta[k], \\ P_L[k] &= u_x[k] - u_\theta[k] \end{aligned} \tag{22}$$

4. Software design

4.1 SLAM

The surveillance robot employs SLAM to draw indoor maps rather than using pre-made maps since digital maps for autonomous patrol do not exist in real-life applications. The robot uses the Kinect sensor to build an indoor map and updates it in a dynamic environment. Originally, the Kinect sensor was developed for Xbox video games as a motion-capture sensor, not for drawing a map. Therefore, it is important to find the best SLAM technique suitable for the Kinect sensor. In this paper, we compare 2D SLAM of Gmapping[11] with 3D SLAM of RTAB-Map[3,4] to find the best algorithm.

4.2 Autonomous patrol

We used ROS Navigation Stack[12] for path planning. SLAM and Navigation Stack were closely connected for autonomous patrol. Fig. 6 shows the overall software structure designed for autonomous patrol. Navigation Stack receives sensor data, odometry, map, and goal position/orientation. Based on the received data, it searches for the best path to the destination and sends velocity commands to the velocity controller.

Navigation Stack has four components, global costmap, local costmap, global planner, and local planner. The global costmap receives a global map from SLAM for global path planning. The local costmap creates a local map representing a small area near the robot by editing the global map using sensor data for local path planning.

The global planner receives a goal position/orientation from the user and searches for the best path to the destination. The global planner uses a global map to create a plan over the entire map. We used Dijkstra's algorithm[13] as a default algorithm to find an optimal global path.

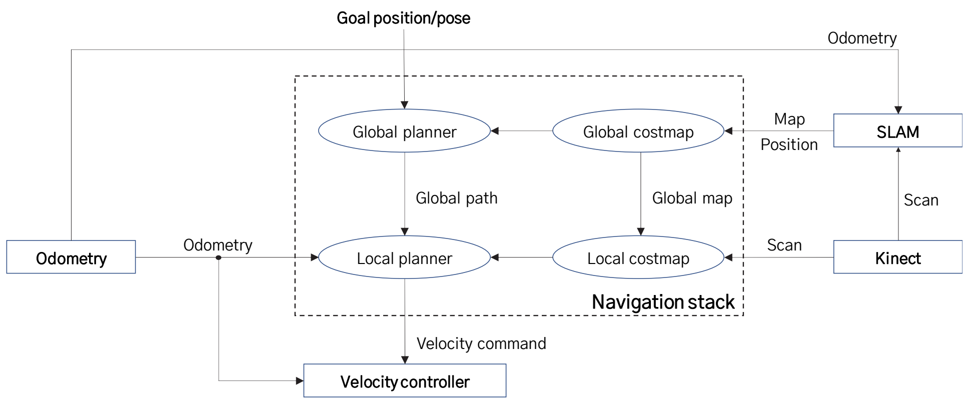


Fig. 6. Software structure for autonomous patrol

The local planner practically moves the robot according to the path-planning result. Based on the local map, the local planner tries to find an optimal local path to follow the global path while avoiding obstacles. We used DWA[14] as a default local path planning algorithm.

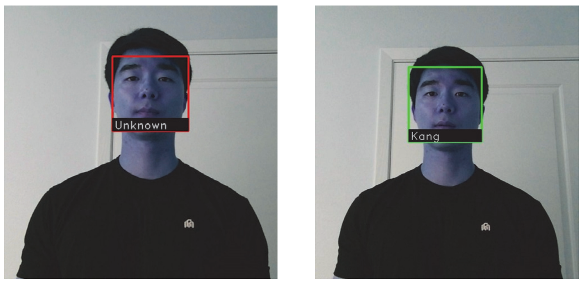
4.3 Face recognition

We used a deep-learning face recognition model[15] to recognize intruders. It uses Convolutional Neural Network (CNN), specifically a ResNet network with 29 convolution layers. This model was trained from 3 million face images and the recognition accuracy is 99.38 %.

In this research, the face recognition program receives 30-Hz live RGB images from the Kinect sensor to distinguish human faces in real time. The face recognition was designed to make three kinds of outputs:

- Nobody: Any face is not detected. The robot continues autonomous patrol.
- Unknown: An unregistered face is detected. The robot starts tracking the intruder.
- Name: A registered face is detected. The robot records the name and continues autonomous patrol.

Fig. 7 shows the visualized results of face recognition before and after registering a face using OpenCV.



(a) Before face registration (b) After face registration

Fig. 7. Face recognition results

4.4 Human tracking

We used skeleton tracking of OpenNI2/NiTE2 for human tracking. OpenNI2/NiTE2 recognizes and extracts human contours from the Kinect sensor's depth image.

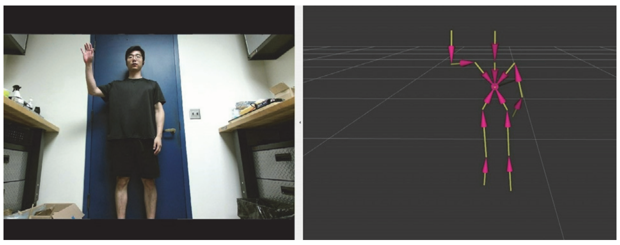


Fig. 8. Skeleton tracking result visualized by RVIZ

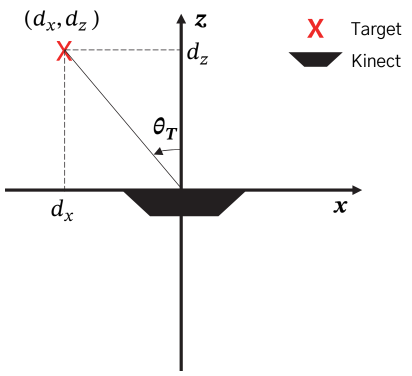


Fig. 9. Coordinates to calculate the target distance and angle

Based on the extracted contours, it tracks 15 skeleton joints such as knees, elbows, head, and torso. Fig. 8 shows the skeleton tracking result visualized by RVIZ.

The torso center provided by skeleton tracking was used as a target for human tracking. Given that the robot moves on the ground, the target distance and angle are only related to the target position in 2D coordinates as shown in Fig. 9. Then, the target distance d_T and angle d_θ can be directly calculated.

$$d_T = \sqrt{d_x^2 + d_z^2}, \qquad \theta_T = \tan^{-1}\left(\frac{d_x}{d_z}\right) \tag{23}$$

where d_x and d_z are the distance to the target along the x- and z-axis respectively.

We designed the robot's position-control system to follow a target and maintain proper position. Fig. 10 shows the position control system for human tracking. The input is the robot's desired position to follow the target, and the output is the robot's position from the target. The control system has two components, the position controller and the velocity controller. The position controller receives the position error and transfers the velocity commend to the velocity controller. The velocity controller receives the velocity error and moves the robot's motors. In the case of the velocity controller, the controller designed in Section 3 was used.

The position controller consists of a distance controller and an angle controller. The distance and angle controllers respectively receive distance and angle errors and transfer linear- and angular-velocity commands to the velocity controller. The desired position of the robot is set as 2.0 m and 0° from the target given the Kinect sensor's depth range, mounting height, and vertical field of view.

4.5 Overall software architecture

In this research, we constructed the overall software structure using the ROS message interface that communicates various types of data between each software node. Fig. 11 shows the overall software structure connecting software nodes based on the hardware connection.

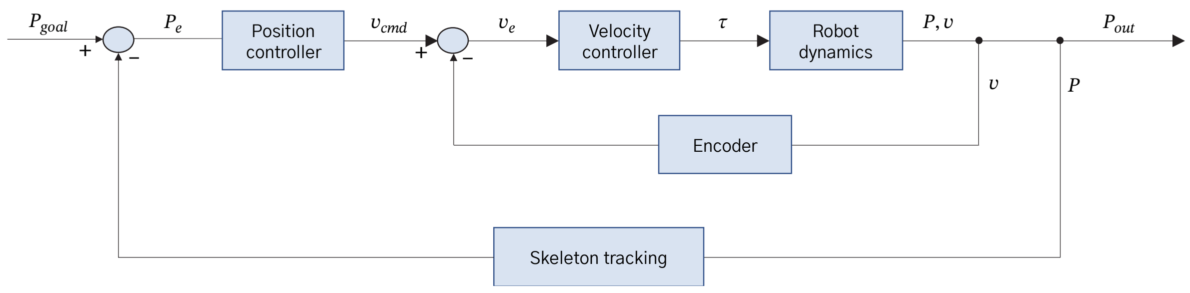


Fig. 10. Position control system for human tracking

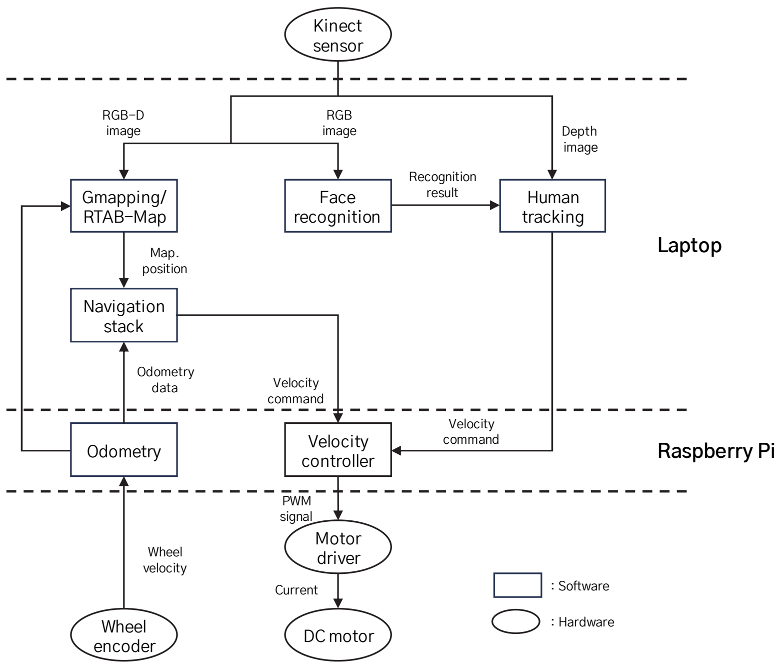


Fig. 11. Overall software structure designed for the robot

The Kinect sensor transfers RGB-D images to Gmapping and RTAB-Map for SLAM. In case of Gmapping, RGB-D images are converted to 2D laser-scan data for 2D SLAM. Gmapping and RTAB-Map create maps and find the robot's current position/orientation.

Based on the map, robot's position/orientation and odometry data, Navigation Stack searches for the best path to the goal. and sends velocity commands to the velocity controller.

The face-recognition node receives RGB images from the Kinect sensor and sends the recognition results to the human-tracking node. If the result is 'Unknown,' the human-tracking node extracts the skeleton data from depth images. Based on the target's skeleton-position data, the position controller in the human-tracking node calculates the needed velocity and sends it to the velocity controller.

A flowchart for the system algorithm was designed based on the overall software structure to achieve the goal of the robot. Fig. 12 shows a simplified system flowchart. The robot has two modes of autonomous patrol and human tracking. In the autonomous patrol mode, the robot moves toward a designated destination based on the created map and path planning results. In the human-tracking mode, the robot tracks the detected person and maintains an appropriate distance and angle from the target based on the 3D skeleton tracking data.

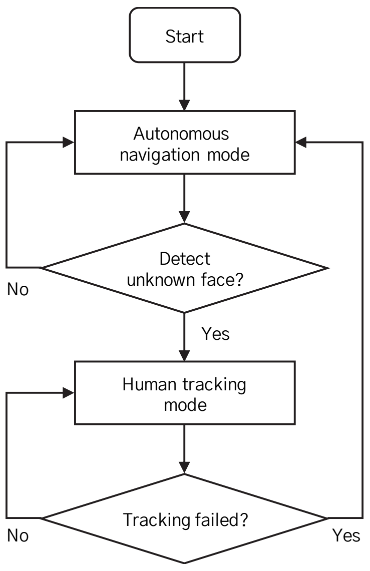


Fig. 12. Simplified system flowchart of the surveillance robot

The robot uses face recognition as a switch to convert modes from autonomous patrol to human tracking. When the robot detects an unknown face, it switches to the human-tracking mode. If the robot loses its tracking target, the robot converts the mode to autonomous patrol and moves toward its original goal.

5. Experimental result

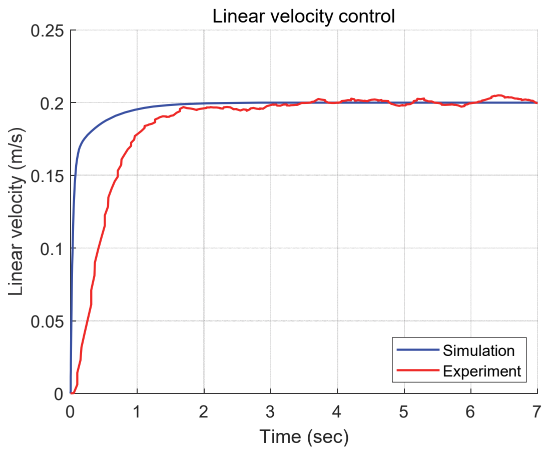
5.1 Velocity control

We conducted experiments to obtain the robot's linear- and angular-motion responses. A 0.2-m/s step input was used for the linear-velocity control and a 0.3-rad/s step input for the angular-velocity control. The linear and angular velocities were calculated using the wheel encoders.

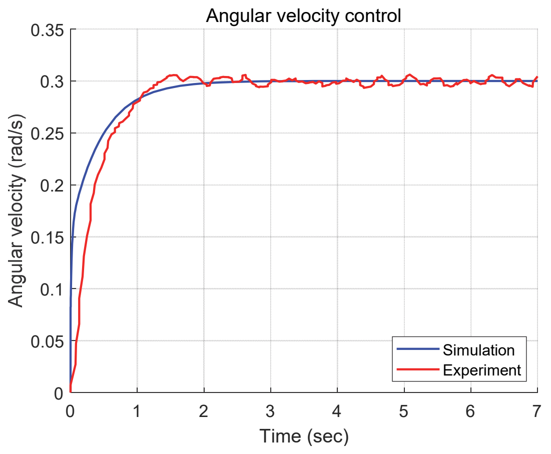
Fig. 13(a) shows the step response of the linear-velocity control. The experimental result shows a 1.06-s rise time and 1.64-s settling time with no overshoot and steady-state error. Compared to the simulation, there is a 0.8 s delay and a small oscillation.

Fig. 13(b) shows the step response of angular-velocity control. The result shows a 0.87-s rise time and 1.24-s settling time with a 1.9 % overshoot and no steady-state error. Compared to the simulation, there is a 0.3 s delay and a small oscillation.

The oscillations and delays in step responses are caused by the uncertainty of the friction force and the hardware limitations such as the sensor noise of the wheel encoders and the dead zone of the DC motors. However, the oscillations and delays are small and acceptable enough to be used for robot control.



(a) Step response of the linear–velocity control



(b) Step response of the angular–velocity control

Fig. 13. Experimental results of velocity control

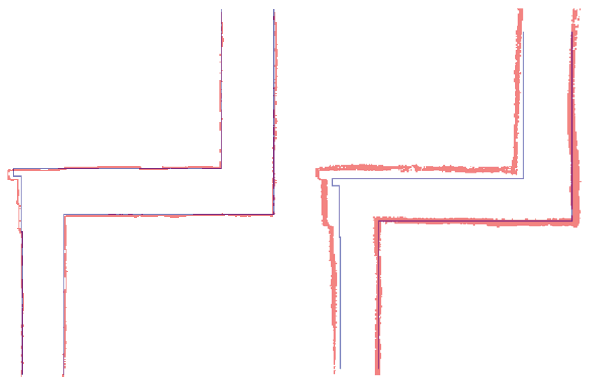
5.2 SLAM

Experiments were designed to evaluate the reliability of the robot's SLAM. The robot mapped the same place using 2D SLAM of Gmapping and 3D SLAM of RTAB-Map. The experiments were conducted in a corridor of a general building.

Fig. 14(a) and Fig. 14(b) show the experimental results of Gmapping and RTAB-Map. The created maps were overlaid on the ground truth map to compare the results of SLAM with the actual map.

An error measurement method proposed in [17] was used to evaluate the accuracy of the created maps. The method uses the K-nearest neighbor. For each occupied cell of the ground truth map, the distance to the nearest occupied cell of the created map is measured as an error. This method provides intuitive error metrics to analyze the accuracy of a map in terms of cells. Table 1 shows the error estimation for the created maps.

As shown in Table 1 and Fig. 14, the map created by Gmapping is highly accurate. In the case of RTAB-Map, the error and noise are relatively large.



(a) Gmapping (b) RTAB–Map

Fig. 14. Created maps overlaid on a ground truth map (red: created map, blue: ground truth map)

Table 1. Error estimation of the SLAM results

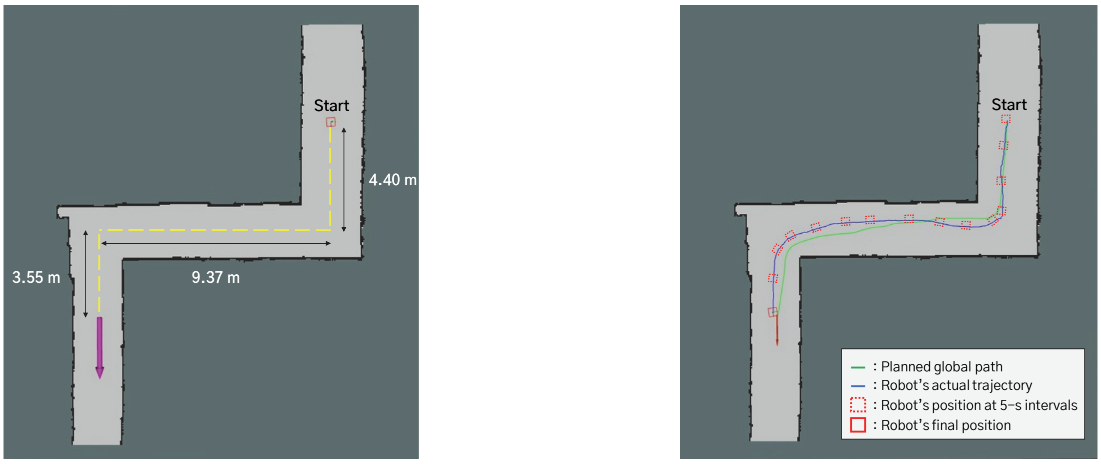
(Unit: pixel)

Algorithm	RMSE	Mean	Max.	Std.
Gmapping	0.9881	0.6466	4.2426	0.7475
RTAB–Map	4.6783	3.1274	12.1655	3.4809

5.3 Autonomous patrol

To test the robot's autonomous patrol, an experiment was designed. Fig. 15(a) shows the experimental setup for the autonomous-patrol experiment. The yellow dashed line is the expected route, and the purple solid arrow is the destination's position and orientation.

Fig. 15(b) shows the experimental result of autonomous patrol with the robot's trajectory and global path planning. The surveillance robot



(a) Route for autonomous patrol (b) Experimental result of autonomous patrol

Fig. 15. Experimental setup and result of autonomous patrol

plans a global path toward the destination without unnecessary detours. At the two corners, the robot plans the shortest possible path while maintaining a safe distance from the wall.

The difference between the planned path and the actual trajectory is caused by localization error and delay in control. However, the deviation is small, and the robot smoothly follows the path, keeping a safe distance from the wall until it reaches its destination. The final position and orientation errors were 0.21 m and 0.14 rad.

5.4 Face recognition

An experiment was designed to evaluate the performance of the robot's face recognition according to the target distance. For the experiment, 1,000 faces of various ages, races, and genders were registered on the database. Also, the recognition target's face was registered. To examine the actual applicability of face recognition, the target was asked to wear various accessories such as a cap, mask, and glasses at each distance.

Table 2 shows the experimental result. The maximum recognition distance was 5.5 m. When the target wore a hat or glasses, the maximum distance decreased to 5.0 m, and when wearing a mask, it decreased to 3.0 m.

Table 2. Experimental result of face recognition

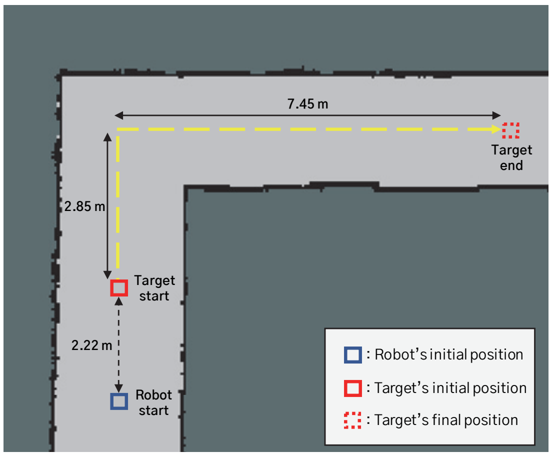
Target	Distance (m)									
	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0
Face	○	○	○	○	○	○	○	○	○	×
With glasses	○	○	○	○	○	○	○	○	×	×
With a cap	○	○	○	○	○	○	○	○	×	×
With a mask	○	○	○	○	×	×	×	×	×	×

5.5 Human tracking

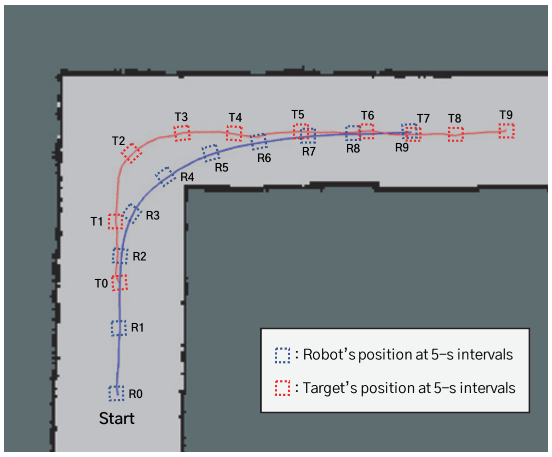
An experiment was conducted in order to estimate the robot's human tracking with a moving target. Fig. 16(a) shows the experimental setup. The target was set at 2.2 m and 0° from the robot. The target was asked to move along the corridor. In this experiment, the robot's velocity was measured by the wheel encoders, and the target position was measured by the Kinect sensor.

Fig. 16(b) shows the experimental result with the robot and target trajectories. As shown in the result, the robot smoothly followed the target and maintained the desired position (2.0 m, 0°).

Fig. 17(a) and Fig. 17(b) show the target distance and the robot's linear velocity. At first, the robot increased its linear velocity to maintain the desired distance of 2.0 m. From 4.19 s to 45.89 s, the robot maintained the goal distance. At 45.89 s, when the moving target suddenly stopped, the robot decreases the velocity and moves backward slightly to maintain the target distance. When the target completely stopped, the distance error was 0.02 m.

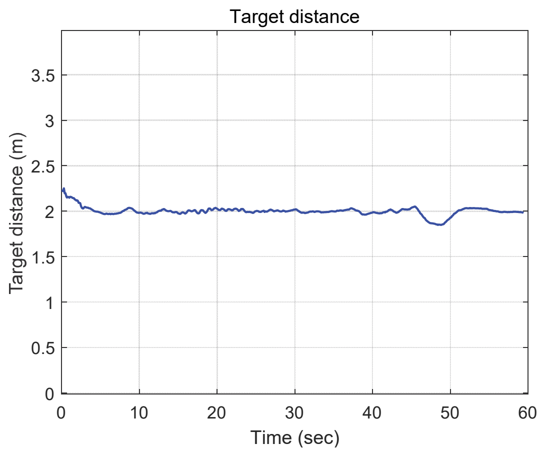

7.45 m
Target end
2.85 m
Target start
2.22 m
Robot start
: Robot's initial position
: Target's initial position
: Target's final position

(a) Experimental setup for human tracking

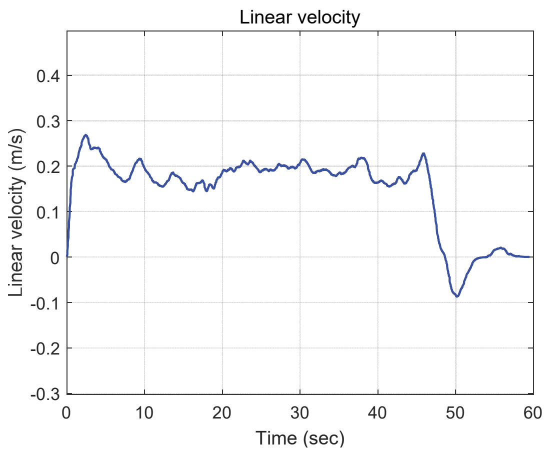

T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
Start
: Robot's position at 5-s intervals
: Target's position at 5-s intervals

(b) Trajectories of the robot and the target

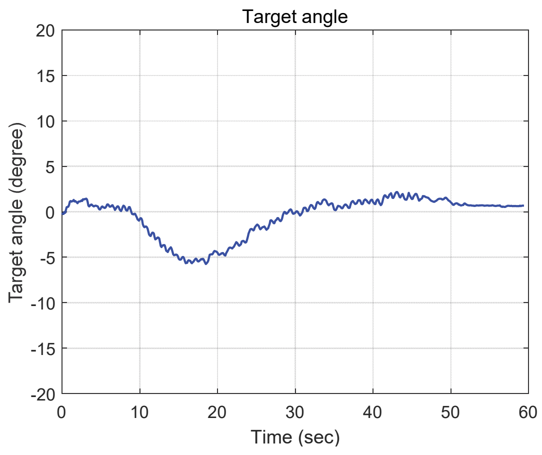
Fig. 16. Experimental setup and result of human tracking



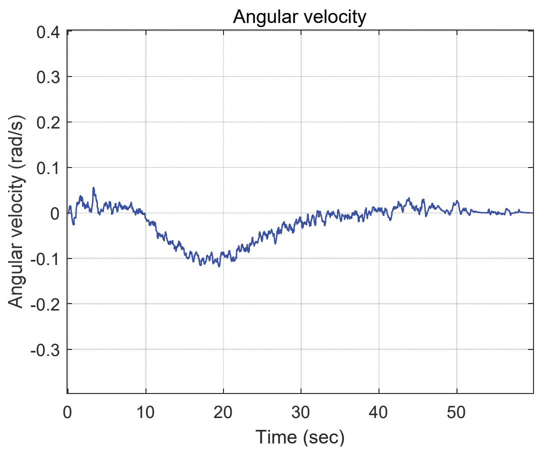
(a) Target distance



(b) Robot's linear velocity



(c) Target angle



(b) Robot's angular velocity

Fig. 17. Result of human-tracking experiment

Fig. 17(c) and Fig. 17(d) show the target angle and the robot's angular velocity. At 8.90 s, the target started a right turn along the corridor, and the robot decreased the angular velocity to negative to reduce the target angle. At 30.11 s, the robot made the target angle 0° and stopped rotating. When the target completely stopped, the steady-state error of the target angle was 0.62°.

6. Conclusion

This paper presents the development of an indoor autonomous surveillance robot based on a single RGB-D sensor. The robot's hardware was designed as a differential-drive robot with two wheels with independent actuators at the front and a caster wheel at the rear. Based on the hardware design, we developed the robot's mathematical model and designed control system. As software, SLAM, autonomous patrol, face recognition, and human tracking were successfully implemented and integrated into a single system based on ROS.

Through experiments for each function, the robot's performance in a real-life environment was demonstrated. The robot created accurate maps with Gmapping and RTAB-Map. The root mean square error of the created maps were 0.9881 for Gmapping and 4.6783 for RTAB-Map in pixel. Based on the created map, the robot successfully planned a patrol path and autonomously moved along the path to the destination without any collision. The final position and orientation errors of the autonomous patrol were 0.21 m and 0.14 rad. The robot reliably detected and recognized a face at 1.5 m – 5.5 m and tracked a moving target while maintaining the desired position without tracking loss. The steady-state errors of the human tracking were 0.02 m and 0.62°.

References

[1] T. Theodoridis and H. Hu, "Toward intelligent security robots: A survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 42, No. 6, pp. 1219–1230, Nov. 2012.
[2] M. Saptharishi, C. Spence Oliver, C. Diehl, K. Bhat, J. Dolan, A. Trebi-Ollennu, and P. Khos, "Distributed surveillance and reconnaissance using multiple autonomous ATVs: Cyberscout," IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, pp. 826–836, Oct. 2002.
[3] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," IEEE Transactions on Robotics, Vol. 29, No. 3, pp. 734–745, Feb. 2013.
[4] M. Labbé, F. Michaud, "RTAB-Map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation," Journal of Field Robotics, Vol. 36, No. 2, pp. 416–446, Oct. 2018.
[5] G. Goswami, S. Bharadwaj, M. Vasta, and R. Singh, "On RGB-D face recognition using kinect," in Proceedings of the 2013 IEEE Sixth International Conference on Biometrics: Theory, Applications, and Systems, Oct. 2013, pp. 1–6.
[6] Y. Lee, J. Chen, C. Tseng, and S. Lai, "Accurate and robust face recognition from RGB-D images with a deep learning approach," in Proceedings of the British Machine Vision Conference, Sep. 2016, pp. 123.1–123.14.
[7] L. Spinello and K. O. Arras, "People detection in RGB-D data," in Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sep. 2011, pp. 3838–3843.
[8] J. Liu, Y. Liu, G. Zhang, P. Zhu, and Y. Q. Chen, "Detecting and tracking people in real time with RGB-D camera," Pattern Recognition Letters, Vol. 53, pp. 16–23, Feb. 2015.
[9] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in Proceedings of the CVPR 2011, Jun. 2011, pp. 1297–1304.
[10] S. G. M. Almeida, F. G. Guimarães, and J. A. Ramírez, "Feature extraction in Brazilian sign language recognition based on phonological structure and using RGB-D sensors," Expert Systems with Applications, Vol. 41, No. 16, pp. 7259–7271, Nov. 2014.
[11] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling," in Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Apr. 2005, pp. 2432–2437.
[12] E. Marder-Eppstein, E. Berger, T.Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in Proceedings of the 2010 IEEE International Conference on Robotics and Automation, May

2010, pp. 300–307.
[13] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik, Vol. 1, No. 1, pp. 269–271, Jun. 1959
[14] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” IEEE Robotics and Automation Magazine, Vol. 4, No. 1, pp. 23–33, Mar. 1997.
[15] D. E. King, “Dlib-ml: A machine learning toolkit,” Journal of Machine Learning Research, Vol. 10, pp. 1755–1758, Jul. 2009.
[16] G. F. Franklin, J. D. Powell, and A. Emami-Naein, Feedback Control of Dynamic Systems (8th Edition), Pearson, Jan. 2018.
[17] J. M. Santos, D. Portugal, and R. P. Rocha, “An evaluation of 2D SLAM techniques available in robot operating system,” in Proceedings of the 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, Oct. 2013, pp. 1–6.