Mobile Robot Platform at Nursing Home for Elderly with Application Service

Choi JiHwon, Jeong JongHyeon, Kim JunSeo and Lim SeungHyeon

Capstone Design(SWE3028), SungKyunKwan University

Abstract. The demand for labor in nursing homes for the elderly has increased rapidly due to the recent increase in the elderly population. Elderly care labor is expected to increase in demand in the future, but it is still suffering from a shortage of manpower. To solve this problem, robots propose a way to provide some care service to save the labor. By easy controls of robots through mobile apps, it will help both elderly people and caregivers by providing functions as patrols and stroll assistants.

Keywords: Robotics · Pose estimation · Mobile app

1 Introduction

Korea has entered an aging society, and the number elderly people is rapidly increasing. While the demand for nursing homes increases as well, the number of nursing care workers does not. Compared to the high demand of nursing care workers, it is difficult to provide appropriate care service to every elderly. In this project, the robots is proposed to compensate for the shortage of manpower.

2 Motivation

This project begins from the thought about what can be achieved through the basic driving function of the robot. What is the solution that robots, which are centered on driving functions, can provide in conjunction with the aging society? What robots could do by looking everywhere on behalf of people and accompanying them? This project may help nursing home to efficiently relocate manpower that are always short-handed.

3 Objectives

This project aims to implement the rapid detection function of fallen accident of elderly people through pose estimation and secure the appropriate distance between robots and users during in stroll assistant mode through object detection.
The pose estimation proceeds by referring to the paper "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields"[1] to learn and

recognize the pose of the fallen person.
In object detection, the distance between the user and the robot must be measured in real time, so an appropriate model should be selected.
In selecting the scope of the project, considering the capabilities of team members, the task was set to implement robots in Gazebo with ROS, deep learning-based model for pose detection, and implementation of web applications for user control.
Both robot testing and implementation of this project are carried out in a virtual environment called Gazebo. This is due to the ease of use of free sensors and location selection. Therefore, the task for actual production of robots is removed so that restrictions can also be ignored such as design of the robot, cost of making robot, and location, and produce the finished product in a short period of time.

4 Backgrounds & Related works

4.1 Object detection

There have been many techniques for classifying images into pre-trained objects through deep learning models. In addition, as the image processing speed has dramatically developed, object detection in moving images as well as still images become possible.
Object detection includes a two-stage method that provides selective navigation through area suggestions, and a single-stage method that calculates the probability that each area contains an object by dividing the image into predefined grid areas.
First, the two stage method has a model of the Region-based Convolution Neural Network (R-CNN) series. The R-CNN-based model has the advantage of being generally more accurate than the single-stage method, but has the disadvantage of being slow in image processing because CNN learning is performed after setting a section with a high probability of having an object as a Region-of-interest (ROI).
On the other hand, You Only Look Once (YOLO)[2] is single-stage method. The YOLO model divides the input image into grids and obtains bounding boxes and confidence through grid cells, indicating where the object is located and the probability that the object is contained.
In this project, YOLO was selected for application in a real-time environment. Object detector is used to enable robots to recognize people and track their location. In addition, it is easy to expand into various services such as object transport and collection through object learning in future studies.
In this project, COCO dataset was used for anonymous human detection in the simulator, and it was well applied to the anonymous model in simulated environment without additional learning.

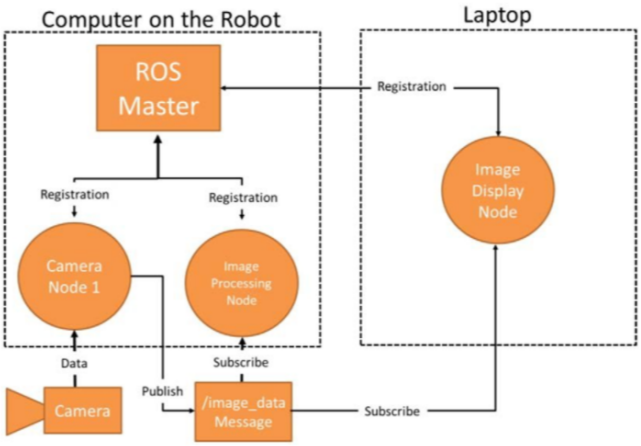

Fig. 1: Simple ROS network exmaple

4.2 ROS(Robot Operating System)

The robot consists of numerous sensors and devices, and accordingly, the software is designed in a complex manner. Therefore, in order to control the robot, it is necessary to build a system that integrates and manages all of thes . In this project, the overall framework was constructed using the Robot Operating System (ROS) based on the TCP network. The ROS consists of several nodes. Each of these nodes exchanges messages with each other to exchange data. By dividing one huge robot framework into nodes, the program can be divided and be able to collaborate with other people. In addition, ROS eliminates problems related to data standardization and API compatibility that may occur due to segmentation.

Figure 1 shows a simple ROS network through topic. All nodes are always connected to $ROSMASTER$. The image data received from the camera goes to $CameraNode1$ first. The $CameraNode1$ node simultaneously publishes the topic $/image\_data$ to $ImageProcessingNode$ and $ImageDisplayNode$. In this way, the image is displayed on the laptop through $ImageDisplayNode$, and $ImageProcessingNode$ processes raw images to better understanding of it. In this way, the ROS may transmit data through a node and implement various functions.

In this project, sensor data of the robot is loaded through ROS, and the robot is controlled by ROS also. In addition, the Navigation Stack provided in the ROS open-source package allows the robot to move without additional mapping or localization. Finally, ROS is well linked with the gazebo simulator and openv, an image processing tool, so it was possible to easily connect the simulator and robot. Object detection and navigation through image processing were also possible.

4.3 Gazebo Simulator

In this project, the goal was to implement robot in the simulator, a virtual environment, to avoid the financial and spatial constraints of robot production. Gazebo simulators are 3D dynamic simulators and are generally used to test robot algorithms or scenarios. Gazebo has high capability of connecting with ROS and provides a unified interface for each api. In Gazebo, the test can be conducted using various sensors such as Lidar sensor and camera sensor, so there are few restrictions on robot manufacturing. Since Gazebo is an open source with an Apache 2.0 license, there are few restrictions on development also.

4.4 Pose estimation

Pose estimation is a computer vision technology that detects a person's posture. It is a common technique that detects the location and direction of an object in computer vision, measuring and estimating the location of a person's body joint, the keypoint, and how it is organized. Pose estimation has two approaches: top-down and bottom-up. The top-down method is a method of detecting a person first and then estimating each person's posture, and there is a disadvantage in that it cannot be measured if a person is not recognized, and as the number of people increases, the amount of calculation also increases. On the other hand, the bottom-up method detects joint areas first and connects them to each other to estimate everyone's posture, but there are many combinations that can match the joints found, and it takes a long time to match properly, so there is a problem of accuracy.

A. Realtime Multi-person 2D pose Estimation - CVPR [3]

This is a method of learning a filter capable of encoding a limb (a joint-tailored portion of a body part) containing location information and direction information of a body part as a 2D vector. The accuracy and speed of pose estimation were improved using a bottom-up approach that performs pose estimation by several people.

B. OpenPose: Realtime Multi-Person 2D Pose Estimation - TPAMI [**?**]

The TPAMI version finds Part Affinity Fields (PAFs) and Part Confidence Maps, which are heat maps containing information on the location of joints and bones. While connecting the two models in series, they were able to improve the model in terms of memory and speed. Confidence maps that inform the location of joints can be found accurately and quickly through PAFs.
PAFs can be trained through ground truth data. Therefore, pose estimation is performed by connecting the joints found through PAFs. In this model, redundant PAF connection was added to increase accuracy for nearby joints.

4.5 Navigation

The robot's driving function uses the navigation package of the ROS. For navigation, you need to know the current location of the robot on the map through localization and receive the destination as an input value. Based on this, the global route is calculated, and the robot keeps updating the location of itself as it moves, continuously modifying the route to the destination in consideration of surrounding obstacles. Therfore robot finally arrives at the destination. We are applying the LiDAR geometry method using LiDAR sensor to the odometry that identifies the robot's current location. It compares map and sensor information and tracks the robot's current location.

5 Problem Statement / Proposed Solution

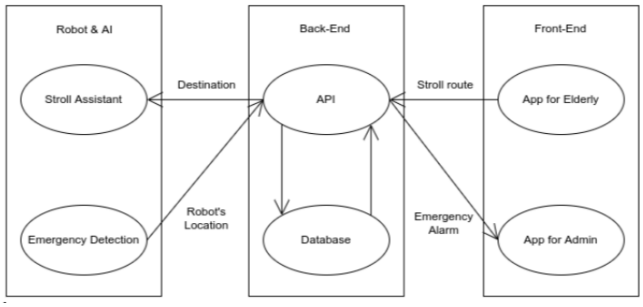

Fig. 2: Overall architecture

5.1 Overall architecture

The overall architecture of nursing home robot services is as follows.2 We will create an app for elderly people who need help from nursing home managers and families, and an administrator app for admins who need to manage elderly people. Development was largely divided into robotics, mobile apps, and back-end.

Robots have two functions in total. The first is the function of accompanying for stroll so that the elderly do not get lost. When the elderly choose a route to take a walk through the app, the robot drives on the route through the map. Another function is to detect emergencies. When an emergency occurs, such as an elderly person behaving abnormally, the robot will take a picture of it as it notices and send emergency alarm to the manager through the app. For the App for admin, the administrator can check the real-time location of the robot and the previous events through log.

The backend connects Robotics and front-end. It builds a database for managing real-time locations, event logs and delivers them to users through APIs.

Stroll Assistant Stroll assistant function was added to help the elderly take a walk. The user sets predefined routes through the application to take a walk on the map, such as figure3(a), and the robot starts driving to the target location based on this. In this case, driving allows the robot to set a path by avoiding obstacles on its own through the Navigation Stack provided by the ROS. While driving, the robot continues to identify the location of the target elderly through object detection to check whether the elderly are following the robot well. We are going to talk about the situation in which the robot fails to detect and misses the elderly in 2-2 Emergency Detection. The robot delivers real-time location information to the server, and the user can know the current location of the robot and the elderly through an application.

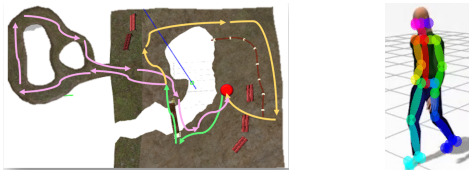

Fig. 3: (a) Sample map in Gazebo and stroll pahts, (b) Rendered human animated image with keypoints

Emergency Detection In emergency detection the robot prevent accidents that could occur to the elderly. The emergency situations are defined in three ways.

Emergency situation
1. User falls down and can't get up. 2. User disappears from the robot's sight and cannot locate it. 3. User directly informs the application that it is an emergency.

When the user fells down while robot receive images from cameras in real time, the image is obtained by human pose estimation to position keypoints such as head, hand, joint, and knee in the frame like figure3(b) above. In the way to detect a person falling and lying on the floor, we have to measure whether the head is located at the bottom of the frame first. At the same time we need to check whether the line connecting the foot and head is parallel to the floor to decide if they're lying down or not . This minimizes the ambiguity that may occur in the actual situation.

Also if the robot cannot detect user by the camera for a certain period of time, the robot stops and looks around. If robot fails to detect a person, he

judges it as an emergency situation.

Finally, in preparation for an emergency situation that is difficult for the robot to decide weather it is emergency situation or not, the user can directly notify the emergency situation through the application.
The biggest limitation of the emergency service is that we cannot distinguish a person to person; just detect where a person/people is/are located. Therefore, there is a possibility of detecting the behavior of others other than "our elderly" and misjudging it as an emergency situation. In preparation for such a situation, when an emergency occurs, the robot takes a picture so that the manager who uses the application can see the picture at the same time as notifying the occurrence of the emergency.

5.2 App Functions

App for elderly

1. Log in
2. Call robot
3. Set stroll path
4. View robot's current location

In the user app, user can call the robot for stroll assistant or the current location of the robot can be identified. When the app calls the robot, it sends the user's current location to the coordinates to call the robot, and the robot moves to the designated location and starts the stroll assistance function. In the stroll assistant mode, the user can set one of the various predetermined courses and enjoy a walk with the robot. During a walk, a button to send a call to the manager app is activated, and if you call the manager as necessary, the current robot and user's location are transmitted to the manager's app.

App for admin

1. Log in to Admin
2. Call robot
3. View robot's current location
4. View command log
5. Receive alarms from robot

When the robot recognizes the falling posture through pose estimation, it captures the situation and sends an alarm along with a warning message to the administrator's app. Since there may be various cases of falling posture, the manager finally decides whether there is an emergency.
An important function that the manager app can do is to allow users to view command logs using robots for robot management. Through the command log, it is possible to identify when and which user used the robot, and to identify the location of the robot. This makes it easier for managers to manage robots.

5.3 Backends

Web Framework Among many web frameworks, our team chose Python's Flask. In addition to simply sending http requests and working with databases, our work requires that we re-export the input information back to the msg format of the ROS, or use the service of the ROS to get the required data. Also, Flask was used rather than heavy Django because this project does not require complex API structures, and robots do not have to process various tasks simultaneously.

Database Among several databases, we used PostgresSQL. Most of our work does not modify the value of DB, and most of our work is to inquire or add. Therefore, PostgresSQL with good performance for this function was selected.

6 Planning in Detail

<table>
<tr><th></th><th>Week6</th><th>Week7</th><th>Week8</th><th>Week9</th><th rowspan="7">Week 10 presentation</th><th>Week11</th><th>Week12</th><th>Week13</th><th>Week14</th><th rowspan="7">Week 15 final presentation</th></tr>
<tr><td>Robotics</td><td>Setting on Gazebo</td><td>Add user model</td><td>Navigation Stockport Edit robot model</td><td>Set Navigation Stack parameter</td><td></td><td>Check navigation function</td><td colspan="2" rowspan="3">Test and trouble shooting</td></tr>
<tr><td>DL</td><td></td><td></td><td>Train model</td><td></td><td></td><td></td></tr>
<tr><td>Frontend</td><td>Build app for user</td><td>Build app for user</td><td>Login feature</td><td>Connect with robot</td><td>Emergency alert message</td><td>Build app for admin</td></tr>
<tr><td>Backend</td><td></td><td>Design Database</td><td>set API for emergency</td><td>Connect robot with app</td><td>Set API for command log</td><td></td><td colspan="3"></td></tr>
<tr><td>Document</td><td>Proposal document</td><td></td><td></td><td></td><td></td><td></td><td colspan="2">Final document</td></tr>
</table>

Fig. 4: Team B Weekly Planner

This project starts from setting the Robotics' virtual environment in Gazebo and model implementation and deep learning model exploration. We are planning to complete the implementation of the necessary robotics and deep learning by week 8. For pose estimation and object detection, select a model, adjust the parameters to suit the needs, and then train will be scheduled continuously. After that, from week 9, the program is integrated after working at the front end and back end for the implementation of the service side.

References

1. Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
2. e. a. Redmon, Joseph, "You only look once: Unified, real-time object detection.," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
3. Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.