Patrol, Detect, and Question: An Indoor Security Robot

Dennis Chen
Dept. of Computer Science and Computer Engineering
HKUST
Hong Kong SAR
Email: dchenam@connect.ust.hk

Peerawas Archavanuntakun
Information and Communication Engineering
Chulalongkorn University
Bangkok,Thailand
Email: peerawas.a@student.chula.ac.th

Guilherme Sato
College of Electrical Engineering
Campinas State University
Brazil
Email: guilhermesato.unicamp@gmail.com

***Abstract*—We present a security robot that demonstrates the basic functions that a human security guard would perform, specifically patrolling a set location, the ability to accurately detect another human, and a natural language interface to uncover the intruder's identity. The robot could serve as a cheap form of security for buildings o a human guard's assistant to cover a wider area. To demonstrate these functions we used the Pioneer P3-DX, Hokuyo laser scanner, and a laptop as the camera, microphone, and speaker. For the software, we used ROS, Google Cloud Speech API, Baidu's Text to Speech API, and YOLO.**

***Keywords*—Security, ROS, YOLO, Robotics, Speech Recognition.**

I. Introduction

Robots have become increasingly utilized in replacing human tasks that are either too difficult, repetitive, or dangerous. Whether it is collecting samples on mars or repetitively picking packages, the applications are endless. Targeting the security market, our group decided to create a prototype security robot to mimic the tasks of a human security guard. It could serve as a weak, but cheaper form of security, or as an assistant for human security guards to cover a large radius using sensors unavailable to a human e.g. thermal imaging cameras, laser. The basic tasks we would like to cover are patrol an area, detect humans only, and question any intruders. Part of this project is not only to create an demo for class project, but also to learn many aspect of robotics projects and applications. Therefore, we used Robot Operating System (ROS) with commodity hardware part rather than specific specialize software for specific completed package robot.

II. Related Work

From *Navigation and Mobile Security System of Home Security Robot*[2], a intelligent security architecture in robot (ISR) is developed. In this paper, a model of what security robot can be seen. The two most important aspect gathered from this paper is the obstacle avoidance system and the security system. Audio Interaction is also particularly important in home security robot to make to robot feels friendly and less intrusive. Because our goals is similar, our robot principle share some similarity. On the other hand, thanks to the technological advances in sensor, robotics, and computer, our products provide much more accurate object and person detection, as well as much smoother audio interaction.

In *Localization and navi-gation of a mobile robot in an office-like environment*[4], a P3-DX is also used as mobile robot for navigation in indoor environment. Their work use only a web cam for localization which made it really compact and cost effective but at the same time heavily rely on landmarks in the area such as light fixtures. The interesting part is how they use particle filter probabilistic methods to localize the position of the robot when there is no information about the starting location of the robot. In our work, we use Monte Carlo localization as well as its been shown and proven to be very effective method.

III. Methodology

A. Hardware

Fig. 1: Assembled form

The hardware components we choose to prototype our security robot are the Pioneer P-3DX, the Hokuyo URG-

LX sensor, and we used an ASUS UX501VW laptop as the speaker, microphone, and web camera. We chose these components because they were the only equipment available to us at the time, and we decided to omit the usage of a Microsoft Kinect, because we faced configuration issues.

Fig. 2: Hokuyo URG-LX

B. Software

For the software architecture of the robot, we decided to use ROS because it allows for parallel peer to peer communication among different independent processes called nodes, where data can be asynchronously communicated using a publisher/subscriber model. The framework is inherently object oriented which abstracts away many low level details and reducing the complexity of understanding the robot system as a whole. Each node is encapsulated and with only an interface that communicates with other nodes via a message topic. This decentralized system allowed for rapid prototyping and a lot of flexibility, allowing for quick switching of subsystems without the need to reconfigure other processes. One example is for the ID recognition, we were able to rapidly experiment between face recognition, traditional OpenCV id card segmentation, and Zbar, which we decided to use in the end. Another advantage in ROS is the integration of other libraries like OpenCV and its big open source communities which contributed many packages like gmapping and the navigation stack.

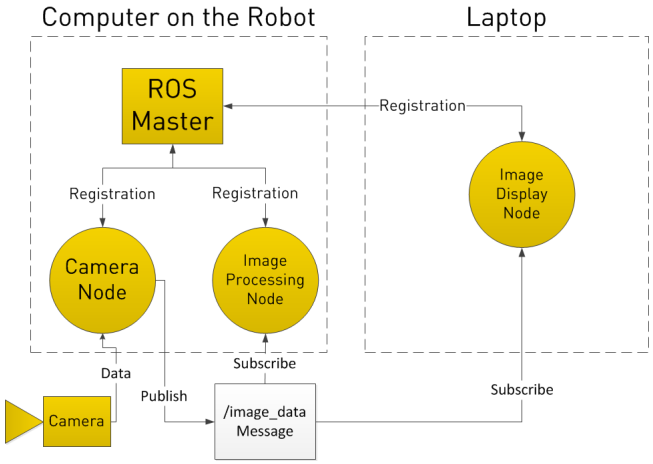

Fig. 3: example of ROS nodes

For the patrolling task, we used slam with gmapping integrating the laser data from the Hokuyo and the odometry from the pioneer. After creating a map of the desired area to patrol, we then used Adaptive Monte Carlo Localization to localize the robot within the map. Using the Move Base Package from the navigation stack the robot was able to easily move between set locations in simulation and the physical robot. In the future, we would want to experiment with Hector SLAM or Google Cartographer to see if better maps can be created.

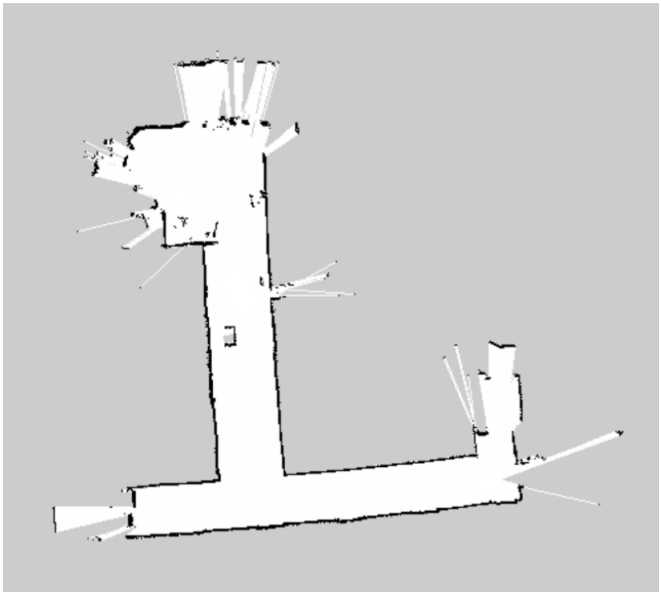

Fig. 4: example of map created

To detect an intruder, we experimented with two different techniques, Histogram of Oriented Gradients(HOG) and the YOLO algorithm trained on the MS-COCO data set, which includes the person class. We found that HOG was good at finding faces, but performed poorly when only shown partial parts of a person. The other algorithm we experimented with is YOLO, which is an algorithm that utilizes convolutional neural networks to divide an image into a grid with multiple grid cells. Each cell is then given two bounding boxes with confidence scores, and conditional class probability. Where confidence is defined as the probability of an object * Intersection Over Union/IOU of the predicted box and ground truth label box. This model has several benefits over traditional object detection algorithms such as being extremely fast, it reasons globally when making predictions, and is highly generalizable so it is less likely to breakdown when presented with new domains or unexpected inputs. For these reasons, we decided to use it for detecting intruders. In fact, YOLO is also much more versatile in that it can detect different types of objects, which can be expanded upon to detecting a person wielding a dangerous objects like guns or knives. Future works can expand upon this flexibility to create a more secure robot that could recognize a dangerous intruder and act accordingly before questioning them.

To confirm the intruder's identity with dialogue, we integrated several cloud APIs such as Google Speech, Baidu's Text to Speech, and DialogFlow to facilitate dialog, and experimented with several programs for identity verification. We chose Google's cloud speech API because it was the most

Fig. 5: YOLO algorithm in action

accurate compared to pocketsphinx or Bing's speech API. Baidu's TTS was the best at pronouncing Chinese words, and Google's DialogFlow allowed variant replies in addition to analyzing an intent when compared to Facebook's Wit.ai. To recognize an intruder, the initial idea was to use OCR to be able to recognize the card and process the image in OpenCV. It recognizes patterns close to each other, therefore recognizing letters and bound them together (bounding rectangles). Then, the image was thresholded and cropped so the OCR would be able to recognize the letters and numbers. The implementation worked; however, we faced a problem where the OCR program takes too long process each frame, which made the recognition system really slow and ineffective for our application. We decided then to use a simple bar scanner called zbar, which publishes the content from the bar scan. Another method experimented with is face recognition, but we found it to be slightly too slow for a streamline experience.

Fig. 6: ID recognition node

IV. Experimental Setup

We set up the nodes to start together as a ROS launch file and tested most of the components beforehand in simulation using Gazebo. First, we tested if the SLAM worked with correct odometry transformation and laser scan input. Next, we cleaned up small smudges in the map using GIMP, because they interfere in the cost map calculation later on. For the localization and movement we used AMCL to localize the robot and the move base package from the ROS navigation stack. The move base package enables automatic trajectory calculations based on a global cost map from the obstacles displayed from the previously created map and a local cost map that detects new unseen obstacles, like a person, from laser scans. We initially planned to have the robot follow an intruder based on laser scans alone trained on people leg patterns, but we discovered that it wasn't stable enough to use without including camera depth information from a Kinect.

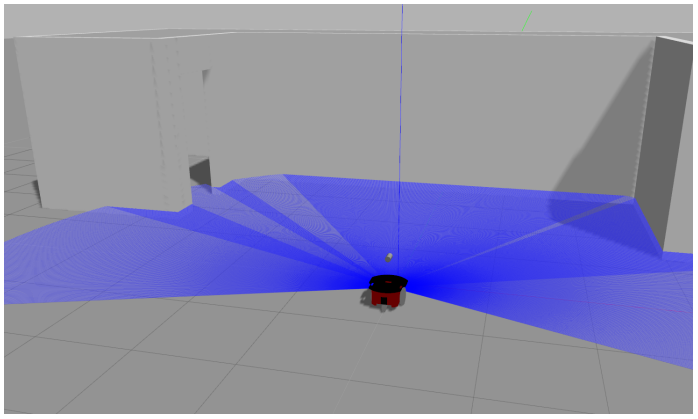

Fig. 7: Pioneer in Gazebo

We used pretrained weights downloaded online to use for the YOLO person detection module and configured it so that the robot will stop moving when it detects a person. It would then initialize the dialog node which uses Google's Speech API, Baidu's TTS, and Dialogflow to demand the detected person to provide identification at another computer terminal. The reason we used another computer terminal as the barcode scanner described in the software section is that zbar proved extremely difficult to use and was incompatible with the ASUS UX501VW's web-cam.

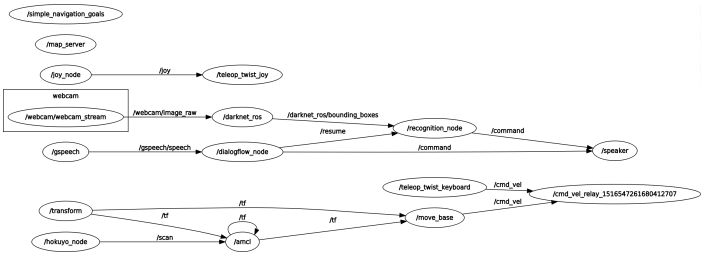

Fig. 8: Graph of the ROS nodes using RQT

V. Results

The results are shown in an annotated video in the link below. The robot worked as functioned but still has much room for improvement. One of the most noticeably one is the user interface. Even though the laptop has fully functioned graphic and audio feedback already, that's only the case when every element of the stack is already been setup. Setting one up still required some understanding of the system in order to start and connect them together as we haven't create a unify start-up interface yet but rather a bash script that will load up ROS, necessary node, and configuration files. Overall, we were able to utilize many facets of robotics like SLAM, computer vision, and ROS, and learned about the importance of teamwork.

VI. DEMO

https://youtu.be/C2pYGdFkoqg

VII. CONCLUSION

The experience of creating a robot together was wonderful. Unfortunately one of our team members disappeared halfway into the semester and we couldn't reach all of our initial expectations for the project. Despite the many hurdles we faced in communication and teamwork, we believe that everyone got to learn something out of this experience.

ACKNOWLEDGMENT

33% - Guilherme Sato camera vision system
33% - Peerawas Archavanuntakun navigation and path planning system
33% - Dennis Chen team leader, speech recognition, and SLAM

REFERENCES

[1] Gapark Marek and olek Peter, *Design the robot as security system in the home*, MMaMS 2014

[2] Ren C. Luo, Po K. Wang, Yu F. Tseng and T. Y. Lin, *Navigation and Mobile Security System of Home Security Robot*, IEEE International Conference on Systems, Man and Cybernetics, 2006

[3] Zeng Dehuai, Xie Cunxi and Li Xuemei, *Design and implementation of a security and patrol robot system*, IEEE International Conference on Mechatronics and Automation, 2005

[4] Paulo Alves, Hugo Costelha and Carlos Neves, *Localization and navigation of a mobile robot in an office-like environment*, 13th International Conference on Autonomous Robot Systems (Robotica), 2013

[5] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.