2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)
July 7-11, 2015. Busan, Korea

ROSBOT: A Low-Cost Autonomous Social Robot

Guohe Fu and Xinyu Zhang

Abstract— **We present a low-cost social robot system composed of a mobile base (a robotic cleaner, $150), an Intel RealSense RGB-D camera ($100), a touch screen powerful laptop ($800), the ROS(Robot Operating System) ($0) and the Intel RealSense SDK ($0). This social robot has the capability of autonomously navigating in an unstructured and dynamic environment, avoiding collisions with stationary or dynamic obstacles and generating rich social behaviors (such as face detection, hand tracking, take pictures, interact with social networks, etc.). Taking advantage of plenty of open source packages in the ROS and Intel cutting edge perceptual computing software, this social robot system allows researchers and school students to rapidly reproduce an affordable and ideal hardware/software platform for their robotics-related research and education. Such a low-cost robot is expected to lead to faster progress in robotics-related application development and research, such as robot-human interaction, social communication, networking robots and even other non-robotic fields.**

I. Introduction

Due to the fact that the robot requirements in academia and industry is rapidly increasing, many schools and research institutes are seeking high cost-effective robots for students to explore robotics-related education and research. In many other fields such as child education, health, brain science, cognitive science and sociology, the demand for robots is also rapidly increasing for the purpose of research. Moreover, many robotic applications were fast growing in the past few year and this trend is likely to continue. Today's robotics is expected to become as ubiquitous in the next 15 years as computer and Internet technology did in the past. Recently, the demand for robots is being driven primarily by large manufacturers, especially in the automotive sector and in the 3C (Computers, Communications and Consumer-Electronics) sectors. It is very likely that the future demand for robots will be driven by non-industrial robots, especially service or social robots.

Many years ago, building a robot was considered as a very challenging task. Many people with different skills spent years to build a workable robotic platform. There already exist a few good robotic platforms such as Nao[1], iCub [7], PR2[2], Unbounded Robot[3], etc. Most of these robots take research groups years to design and build. As a result, either they are not sold in public or they are often very expensive. For example, a PR2 costs $400,000 and it is very difficult to reproduce one. Recently, some less expensive robots, including Luke [10], Nelson [3], etc, were designed using off-the-shelf devices. They have shown some promising capability, like mobility, taking pictures, facial emotions and other social behaviors. However, their implementation still heavily depends on complex mechanical skills and complex software.

Recent advances in robotics-related technologies such as mobility, artificial intelligence, wearable devices, robotic bases, mobile Internet, sensors, RGB-D cameras, etc., have made it possible to quickly build an affordable robot. In this paper, we present a low-cost, easy to built social robot system including both hardware and software. This social robot has the capability of autonomously moving in an unstructured and dynamic environment, avoiding collisions with surrounding obstacles and showing rich social behaviors. With the ROS (Robot Operating System) [8] integration and Intel RealSense SDK[4], our social robot system allows users to conduct robotics-related research without worrying hardware integration and software complexity. Moreover, most components of this social robot can obtained from online shops (e.g., eBay in U.S.A., gmarket in Korea, Taobao in China). In addition to its low cost, building such a robot requires little mechanical skills. This allows the searchers and school teachers/students to quickly start their research and teaching.

The paper is structured as follows. In Section II, we give a brief review over related work. In Section III, we describe the architecture of our social robot. We introduce the hardware components and software functionalities in Section IV and Section V, respectively. Some preliminary results are given in Section VI. The paper is concluded in Section VII.

II. Related Work

There are many efforts dedicated to create various robotic hardware and software platforms in the past. For social robots, we refer the renders to [4] and [5] for the survey. Here we only review the work that inspire us or that is closely related to our work.

iRobot Create[5] is a robust, industrially produced platform with a sensor array and a high payload capacity. This platform can easily be extended by including other sensors and equipments. Many extensions can be found at the ROS community. For example, TurtleBot-1[6] is built based on

The authors are with Shanghai Key Laboratory of Trustworthy Computing, MoE Engineering Research Center for Software/Hardware Co-design Technology & Application, and Intelligent Robot Motion & Vision Laboratory, East China Normal University, Shanghai, China. The corresponding author is Xinyu Zhang. Email: xyzhang@sei.ecnu.edu.cn

[1]https://www.aldebaran.com

[2]https://www.willowgarage.com/pages/pr2/overview

[3]http://unboundedrobotics.com/ubr-1/

[4]https://realsenseappchallenge.intel.com/

[5]http://www.irobot.com/About-iRobot/STEM.aspx

[6]http://wiki.ros.org/Robots/TurtleBot

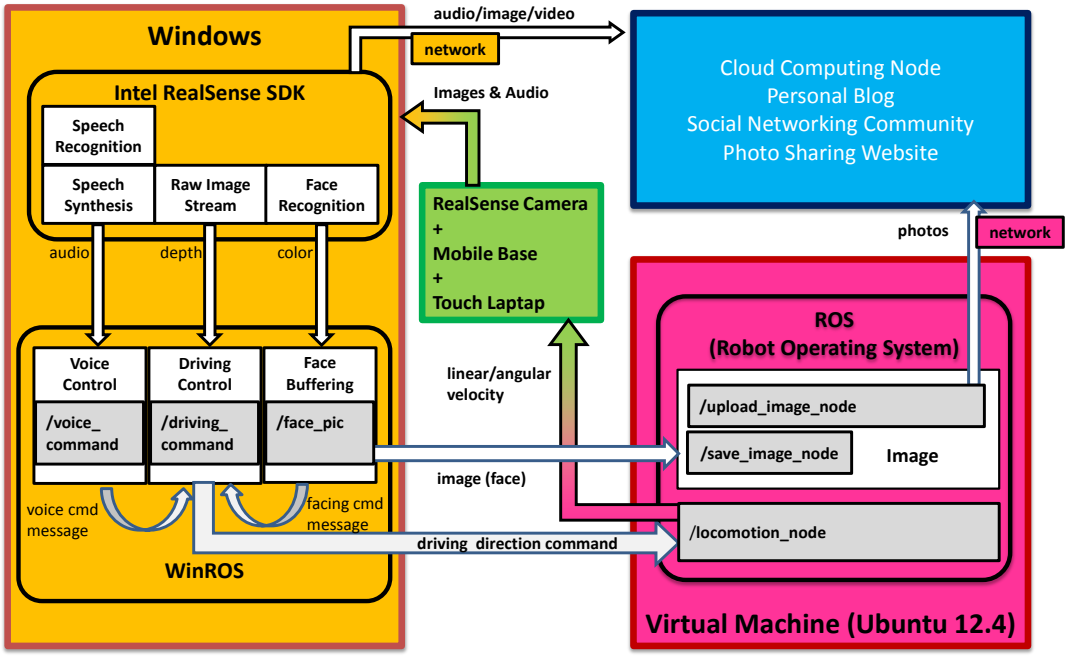

Fig. 1. The Architecture of Our Social Robot System.

iRobot Create. Luke [10] is based on TurbleBot-2[7]. Luke is less complex in terms of hardware installation, but its social behavior is limited and its software is still complicated. Nelson [3] is a low-cost robotic platform. It requires relatively complex mechanical installation. For instance, its head consist of a three degree-of-freedom neck and a seven degree-of-freedom face. Assembling these neck and face servos and mechanical parts requires other advanced stills and tools such as 3D printer and laser cutter.

A few years ago, Aldebaran Robotics has produced the Nao robot [1]. Recently, Nao's sister, Pepper[8] was announced to incorporate more capabilities. Sony's AIBO[9] is an famous robot that has been discontinued for development. These robots are still expensive. Other robots developed by academic groups include CMU Chiara [2], iCub [7], Myro [6] and Tekkotsu [9]. There are many other robots that are able to show different social capabilities and behaviors [4].

Some software platforms for robotics can be found in the survey article [5]. Among them, the ROS [8] is the most well-known. It is an open source robotic platform and has been integrating plenty of libraries and tools that help users build robot applications. Since 2009, the ROS have been widely used in industry and academia.

III. Robotic System Overview

To utilize existing hardware devices and variety of software packages both in Windows and Linux, our social robot, named as ROSBOT, uses a dual-system hierarchical architecture, as shown in Figure 1. The two systems are running in a touch screen Laptop. Communication between the two operating systems is performed using a strategy of message exchange. This is achieved by inter-process message passing between the WinROS (a ROS version for Windows) and the Hydro ROS (a primary ROS version for Ubuntu)[10]. These messages can be driving commands, voice, images obtained in one operating system, and then sent to the other operating system (e.g. sent from the WinROS to the Hydro ROS).

In Figure 1, the orange block describes the components and functionalities in Windows and the red block describes the components in Ubuntu. The arrows indicate the messages and their forwarding directions. The software packages in Windows are primarily used to capture audio and video data (RGB images and depth images). The packages in Ubuntu are responsible for driving a mobile base and taking care of other servos. The green block shows the robot hardware, consisting of an iRobot mobile base, an Intel RealSense RGB-D camera[11] and an touch screen laptop. The blue block shows some Internet resources accessible to our robot. These resource can be a cloud computing node, an image/video sharing website, a blog, a social network, etc.

The work flow of our system is described as follows. ROSBOT captures audio/image/video streams (e.g. RGB and depth images) in Windows using an Intel RealSense RGB-D camera. The RGB-D streams are used to perform collision

[7]http://www.turtlebot.com
[8]https://www.aldebaran.com/en/a-robots/who-is-pepper
[9]http://www.sony-aibo.co.uk
[10]http://www.ros.org
[11]Like many devices, the Intel RealSense RGB-D camera has drivers only for Windows (more specifically Windows 8)

avoidance and then execute motion control. Moreover, RGB images are used to detect the objects of interest (e.g. human face). Further emotion detection and recognition can be implemented using Intel SealSense SDK. Audio stream is used to perform voice-based robot control via voice detection and recognition. As a result, motion controlling commands can be given from different modules such as collision avoidance (e.g. turning left/right to avoid obstacles), voice (vocal commands by a user), gesture (e.g. a thumb-up means GO) and facial/emotional order (e.g. moving away if an angry face). Driving commands and other additional messages are sent to from the WinROS under Windows to the Hydro ROS run in Ubuntu. For example, when ROSBOT is approaching an obstacle, ROS uses these driving commands to actuate the robot's servos. ROSBOT can generate some interesting social behaviors. For instance, this social robot can share audio or fun images with friends through social network. Using gesture recognition provided by Intel RealSense SDK, this robot can play with friends. Using the touch screen laptop, a user can interact with this social robot in a more natural manner.

IV. Hardware Components

Our goal is to create a low-cost social robot, so that we try to make the hardware structure as simple as possible. Specifically, this social robot consists of three main components: mobile base, RGB-D camera and touch screen laptop, as shown in Figure 2. In order to mount these devices, we build three-layer support using aluminum sticks. The basic structure is similar to TurtleBot 1, but the cost ($150, the base plus all accessories) is much lower than the latter.

A RGB-D camera is mounted on the top and its elevation angle can be adjusted accordingly. It comes with a built-in microphone. A touch screen laptop can be mounted in the front of the robot. During the test, we just put the laptop in the middle layer for safety. We will use the speakers built-in the touch screen laptop to synthesize speech.

A. Mobile Base

We adopt a low-cost, yet high effective iRobot Create ($150 together with other accessories) as our mobile base. It is a complete robot development kit that allows a user to program robot behaviors without having to worry about mechanical assembly and low-level code. It can handle up to 4kg payload and its maximum speed is 50 centimeters per second. The iRobot Creates Open Interface provides users with a set of commands to control it. We use a serial cable to connect between the iRobot Create and the touch screen Laptop. A USB-RS232 convertor can be used to connect to a USB port of the laptop in case there is no RS232 serial port in the touch screen laptop. After connection, it starts a serial terminal program that is capable of sending data.

Other alternative robotic base include Kuboki[12] or iRobot Create 2. These alternative are relatively expensive, but have similar performance to iRobot Create. Moreover, we tested other old model iRobot cleaners (e.g. Roomba 5104). These old models work fine and the performance looks the same. They are cheaper than iRobot Create. The only issue with these cleaners, is that they have no mounting holes on the back side. With simple tools like pistol drill, one can easily make some holes.

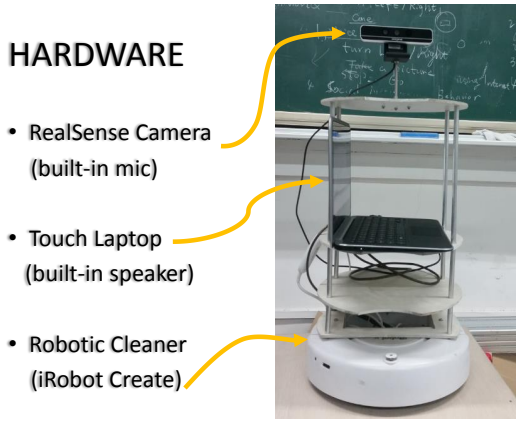

Fig. 2. Hardware Components Cost about $1050

B. Vision

In order to allow the robot to navigate autonomously, we mount an Intel RealSense RGB-D camera ($100) on the top. This small camera has full 1080p color and a depth sensor, which gives our robot RGB-D vision. Its depth sensor has 0.2∼1.2 meter range. Working together with its accompanying SDK, ROSBOT can provide more powerful capability with facial recognition, emotion tracking, 3D scanning, and background extraction, and even 10-finger gesture recognition.

Other alternative RGB-D cameras include Asus Xtion Pro Live ($180), Microsoft Kinect ($250). These two types of RGB-D cameras may not have rich supporting packages like Intel RealSense SDK.

C. Voice

There are two microphones that we can use. The one is built inside the touch screen laptop and the other is equipped with Intel RealSense RGB-D camera. In addition to the normal environmental noise, both the microphones surfer from the heave noise caused by the moving base. The noise will greatly affect the accuracy of speech detection and recognition. We found that the microphone built in RealSense RGB-D camera has more satisfactory anti-noise capability than the one built in the laptop. We also tested an external blue-tooth microphone, it is less satisfactory than the built-in ones. We guess the audio sensor of Intel RealSense RGB-D camera has a better noise removal filter.

D. Computing Power

We use a DELL touch screen laptop as the computing resource of our robot. It has Intel Core i5 4210U Dual-Core 2.7G CPU with hyper threads. It consists of 8GB DDR3L memory (1.6GHZ) and 256GB SSD harddisk. Its 12.5 inches

[12]http://kobuki.yujinrobot.com/home-kr/

HD touch display enable many intuitive interaction between the robot and human. It has over 8 hours of battery life. A Hydro ROS [8] is installed under Ubuntu 12.04 LTS, which is run in a virtual machine. The WinROS, the drivers of the Intel RealSense RGB-D camera and their accompanying SDK packages are installed under Windows. All software packages and modules, including collision avoidance, face detection and recognition, voice recognition and synthesis, social networking activities, are run on this DELL touch screen laptop. Its high performance meets the requirement for robot-human interaction.

V. Software Packages

Writing software for robots is difficult, particularly for social robots that requires great integration efforts of mathematics, perception, mobility and beyond. In ROSBOT, we use the open source ROS and the well-designed Intel RealSense SDK to build our software platform.

A. Collision Avoidance & Motion Control

ROSBOT is able to autonomously move around in a complex environment. This functionality is achieved by the collision avoidance and steering direction decision modules. The algorithm used in our robot is based on the point cloud generated from a depth image. It consists of the following steps.

1) Intel RealSense SDK retrieves a depth image from its RGB-D camera and then a point cloud is generated from the depth image. An octree-based filter is used to simplify the dense point cloud. This can significantly speedup up the point cloud processing.
2) Steering direction is determined by the centroid (coordinates x, y, z) of the simplified point cloud. Steering angle can be computed by the relative position between the centroid and a user specified point of interest. If the depth value of the centroid is greater than the depth value specified by the user, the robot can move straight forward, otherwise the robot moves backward or steers its direction.

Unfortunately, though this algorithm is simple, it does not always work. One of the challenging scenarios in an unstructured environment, is that the robot may be trapped into a deadlock at a corner. This scenario is described in Figure 3. At beginning, the robot find the centroid of the captured point cloud is on its left side, then it turns right to avoid obstacles (see Figure 3-(a)). However, after having turned right, the centroid of the captured point cloud moves to the right, then the robot has to turn left in order to leave away from the obstacles (see Figure 3-(b)). After that, the centroid moves to the left and the robot has to turn right (see Figure 3-(c)). The robot repetitively turns left/turn right, but cannot move out of the corner (Figure 3-(d)).

To avoid this quandary, we propose a new algorithm, called Binary Exponential Turning (BET). As shown in Figure 4, we record the turning direction, 0 for turning left and 1 for turning right. Once we find there exists 1010 pattern in the past, we make two extra left turns after a previous

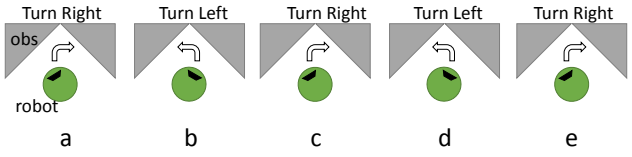

Fig. 3. Deadlock at A Corner. From (a) to (d), the robot (in green) repetitively turns left, turn right,..., but cannot move out of the corner (in grey).

turn, as shown in Figure 4. If two successive 1010 pattern are observed in the historic record, we make 4 ($= 2^2$) extra left turns. If a third successive 1010 pattern occurs, make 8 ($= 2^3$) left turns. In general, after n successive 1010 pattern, 2^n left turns are applied. With this algorithm, the chance of the robot being trapped in a deadlock would be negligible. It is trial to extend the algorithm to a 0101 pattern or hybrid patterns (e.g. 1010 followed by 0101).

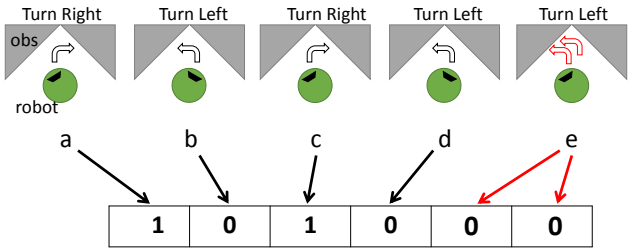

Fig. 4. Deadlock Prevention using Binary Exponential Turning. After n successive 1010 pattern, 2^n left turns are applied to reduce the chance being trapped at a corner.

B. Face Detection & Recognition

When ROSBOT is navigating, it keeps detecting the objects of interest. For instance, the robot searches for a human (face) to whom it wants to talk and take a picture. There are various algorithms for object detection, especially for face detection. In our robot, two types of images are available, RGB images and depth images, to perform face detection. Luke [10] uses depth image to perform human and face detection. Since the depth range of our RGB-D camera limits from 0.2∼1.2 meters, it is not practical for our robot to detect movable objects far off the camera. Considering the distance from the camera to the human face is typically range from 1.5∼3.0 meters, we use RGB images to perform face detection. Intel RealSense SDK makes this module relatively simple. Besides face detection, Intel RealSense SDK gives our robot capability of face recognition, face tracking, facial emotion detection. If the robot plays as a friend finder, it can keep clear images with desired emotions (smiling, angry, sad, etc.) for photo sharing.

C. Voice Recognition & Synthesis

To handle the speech detection and speech synthesis, we use the voice features provided in Intel RealSense SDK. We use a built-in microphone of RealSense RGB-D camera to receive a user's voice. The controlling commands are

Fig. 5. ROSBOT follows his human buddy when his human buddy is wandering in a building lobby.

(a) (b) (c)

Fig. 6. ROSBOT is able to avoid collisions with stationary (a) or dynamic obstacles (b) and keep a safe distance with his human buddy (c).

associated with predefined speech words. Speech synthesis is also using user specified sentences. We use the speaker built in the touch laptop to speak out the words. It does not require the database of speech samples and digital signal filters. This can significantly reduce the computational resource requirement like CPU and memory, which makes ROSBOT more interactive and high performance. Right now, our robot supports some simple speech commands like 'stop', 'go', 'take a picture', 'follow me', 'find a friendly face', etc.

D. Communication Through Internet

ROSBOT is fully connected with the Internet through WiFi router in an in-door environment. ROSBOT can login its own blog, post texts and images using the APIs provided individual Web sites. The robot also can store the photos that were taken during the navigation and share them with friends using social networking capability. This is done using UPLOAD API.

E. Other Human-Robot Interactions

There are many other features we are still developing, like voice chatting, game playing, emotion control and so on. We can switch the controlling manner from autonomy to manual manipulation. With a WiFi router, a user can manually control the ROSBOT through a mobile phone. This allows ROSBOT quickly navigating a room, for instance, to generate the room map.

VI. Social Behaviors

ROSBOT shows various social behaviors.

A. People Following

The problem of following people is more challenging than it looks like. It requires understanding and integrating many techniques such mathematics, sensor, perception, human social behavior and so on. Our robot uses a simplified algorithm based on depth images to perform people following. We describe this algorithm as follows.

1) The user switches ROSBOT to following-me mode (using speech, gesture or mobile phone).
2) The RealSense RGB-D camera retrieves RGB images and depth images. Collision detection and face detection are performed.
3) Human face is specified as the region of interest. It computes the relative configuration between the region of interest and the centroid of point cloud (generating from the depth image). It then calculates the percentage occupied by the region of interest. Then the distance from the human and the robot can be estimated.
4) Like the description in Section V, this percentage can also be used to determine if the robot to move forward or backward. The ratio between the left region of interest and the centroid of point cloud can be used to determine steering directions.
5) If the robot loses the face, depth image can still be used to follow the moving object (e.g. a walking person), as described in Section V.

As shown in Figure 5, ROSBOT follows his human buddy when his buddy is wandering at a building lobby.

B. Autonomous Navigation

ROSBOT can autonomously navigate in an unstructured and dynamic in-door environment using the collision avoidance strategy described in Section V. In particular, the Binary Exponential Turning algorithm can prevent the robot from being trapped at a narrow corner. ROSBOT can handle stationary obstacles, including chair, bookshelf, cabinet, desk, wall, as well dynamic obstacles like walking person. The mobile base has additional sensor to avoid low obstacle and even steps. During the autonomous navigation, we let the robot move at the speed of $0.2m/s$. If the speed is too high,

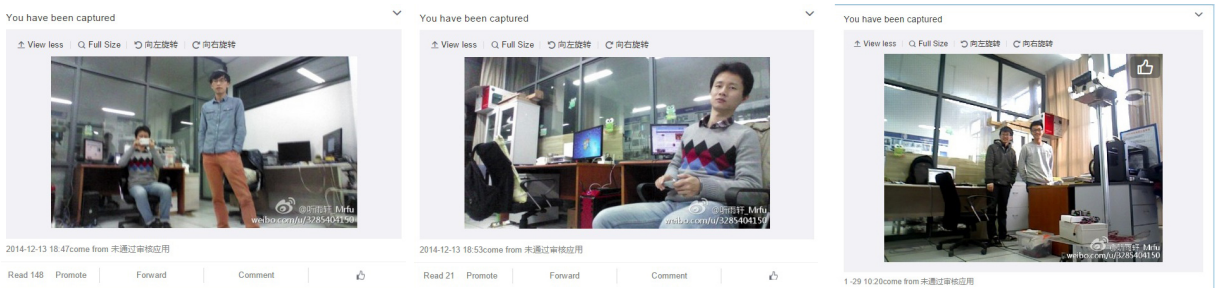

Fig. 7. ROSBOT found the faces of some labmates when autonomously navigating the laboratory room. After taking pictures, ROSBOT immediately uploads them to a photo sharing website or its online album. My labmates look very excited when see their picture taken by a robot.

it is more likely to miss the object of interest (e.g. human face). If the speed is too low, the robot looks less smart. Certainly, there are some limitations. Since we use only a single RGB-D camera, there are some blind spots where the robot can see nothing. As shown in Figure 6, ROSBOT tries to avoid collision with his human buddy.

C. Interacting with People using Social Networks

Social networking communities are very active and people spend plenty of their time on blogging, chatting, sharing texts and images, etc. It is very fun to allow a robot play with social networking. Our preliminary results have demonstrate our robot has very exciting social behaviors through networking. It can take a picture for a person when moving around, detect the emotion of the person, give a simple description of the scene and the person (like where I am, how the person looks like and how he/she feels), post the image and texts to its own blog or uploading the pictures to a photo sharing website. As shown in Figure 7, ROSBOT automatically uploads images to his blog space.

VII. Conclusions

Driven by growing applications in robotics and related fields, the demand for robots, especially for social robots, is increasing fast. We presented a low-cost, easy-to-reproduce social robot for research and education. The total cost is less than $1050. This robot is built using off-the-shelf devices and existing software packages. Under the open source platform, ROS and Intel RealSense SDK, this robot is able to achieve rich social behaviors, such as autonomously moving in an unstructured and dynamic environment, avoiding collisions with obstacles, following people, voice interaction, sharing information through Internet. Such as a low cost social robot is an good option for teaching considering that increasing robotics-related requirements.

Our algorithm has some limitations. Firstly, our collision avoidance algorithm can not handle very low obstacles due to the capability of its RGB-D camera. The algorithm cannot handle very complex in-door environment like narrow passages. Secondly, our robot often moves with a low speed due to its base capability. For people following, sometimes our robot may miss the target if his human buddy walks too fast. Thirdly, noises greatly affect the accuracy of speech detection. A possible direction is to use a bluetooth device to communicate between the robot and his human buddy.

Besides, in social behaviors for robots, privacy issues is increasingly paid attention. It would be interesting to investigate privacy issues in future. We would also like to include other social behaviors like group-forming motion, social distance among participants, side-by-side walking and other tasks.

ACKNOWLEDGMENT

This research work was supported in part by the Foundation for Innovative Research Groups of the NSFC(No.61321064), the Shanghai NSF(No.14ZR1412300), the SRF for ROCS, SEM and the Open Project Program of the State Key Laboratory of CAD&CG(No.A1504), Zhejiang University.

References

[1] Aldebaran-Robotics, "NAO," 2004, https://www.aldebaran.com/.
[2] Z. Dodds, L. Greenwald, A. Howard, S. Tejada, and J. Weinberg., "Components, Curriculum, and Community: Robots and Robotics in Undergraduate AI Education," *AI Magazine*, vol. 27, no. 1, p. 1122, 2006.
[3] M. Ferguson, N. Webb, and T. Strzalkowski, "Nelson: a low-cost social robot for research and education," in *ACM Technical Symposium on Computer Science Education*, 2011, pp. 225–230.
[4] T. W. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, 2003.
[5] J. Kramer and M. Scheutz, "Development Environments for Autonomous Mobile Robots: A Survey," *Autonomous Robots*, vol. 22, pp. 101–132, 2007.
[6] D. Kumar, D. Blank, T. Balch, , K. O'Hara, M. Guzdial, and S. Tansley, "Engaging computing students with ai and robotics," in *AAAI Spring Symposium on Using AI to Motivate Greater Participation in CS*, 2008.
[7] L. Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, and G. G. Sandini, *The iCub platform: a tool for studying intrinsically motivated learning.* springer-verlag, 2013.
[8] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng., "ROS: an open-source robot operating system," in *In Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*, 2009.
[9] E. Tira-Thompson, "Tekkotsu: A rapid development framework for robotics," Master's thesis, Robotics Institute, Carnegie Mellon University, May 2004.
[10] M. Zabarauskas and S. Cameron, "Luke: An autonomous robot photographer," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1809–1815.