Building a Telepresence Robot Based on an open-source Robot Operating System and Android

Ha M. Do, Craig J. Mouser, and Weihua Sheng

***Abstract*—The open-source Robot Operating System (ROS) provides operating system like services to operate robots such as hardware abstraction, device drivers, mapping, autonomous navigation… We explain our low cost approach that uses an iRobot Create, ROS, and an Android device to build a Telepresence Robot. Besides using existing ROS packages, such as uvc_camera, hokuyo_node, and gmapping, we used move_base package to develop to efficiently handle motion planning for the robot. Moreover, we built an Android application based on rosjava and android_core to control and view a live video stream from the remote robot. The robot could also be used to have a video conference.**

***Index Terms*— Android Tablet, iRobot Create, Ros, RosJava, Telepresence Robot.**

I. Introduction

THIS project is in support of the Laboratory for Advanced Sensing, Computation and Control (ASCC), and for the graduate course in the Electrical and Computer Engineering School at Oklahoma State University. The goal of this project is to develop an affordable telepresence robot controlled by an Android tablet for the remote users. The end goal for this project is to create a robot that can be remotely navigated in order to effectively communicate with others and have smart features such as hand gesture and voice recognition abilities. Our project is going on the second phase with the addition of those smart features to the robot. In the first phase of our project we restricted our goal to a few key components. Our goals for this phase include three main components. First, we wanted to build a telepresence robot platform based on ROS to control a robot from an Android tablet. Second, we wanted to stream a live video feed from the remote robot to the Android tablet. Our third goal was to create a 2D map from data collected from a laser mounted on the remote robot and to implement motion planning feature for the robot using this map.

In Section II of this paper, we will explain the motivation for creating an affordable and simple telepresence robot. In Section III, we will cover previous work done in this area. We will discuss how the previous work can be incorporated into our research. Section IV will cover the hardware we used to create our telepresence robot and the Android tablet we used for development. Section V will cover the software used and developed for this project. Section VI will cover the results of our research and development during this project. Section VII will discuss potential future work that could be done to improve our design. We will offer our conclusions to the project's research, development, and progress at the end of this paper.

Manuscript received December 14, 2011. This work was supported in part by the ASCC Lab at Oklahoma State University

Ha M. Do, Craig. J. Mouser, Weihue Sheng are with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74075 USA, (e-mail: ha.do@okstate.edu, cmouser@okstate.edu, weihua.sheng@okstate.edu).

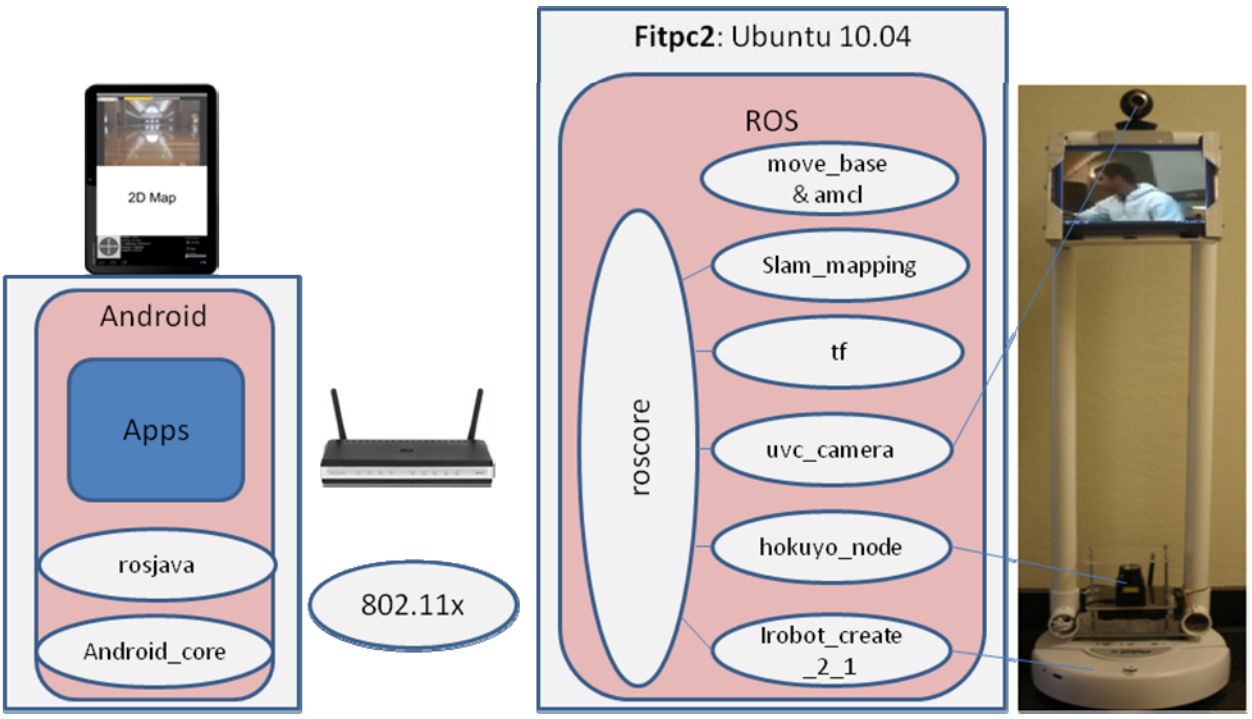
II. Motivation

Telepresence robots can allow a user to interact with remote people through video and audio communication. They have a wide variety of uses in everyday life [1]. These robots allow their users to be in two or more places at once. Having this ability would allow for great increases in productivity for businesses. It would also be very effective for those who are ill. They can still attend class or work without risking spreading the illness to others. Also, it could be used by doctors and nurses to remotely check on patients, even in their own homes [2].

A well-built telepresence robot would allow the user to effectively teach, communicate and learn from afar. These technologies would allow you to act as a businessman, engineer, teacher, student, and more without even leaving your home. It would also allow for those unable to attend events because of physical illness or disability to experience the event as though they were there.

One key component of this project is to make the telepresence robot affordable. Our motivation for this aspect is to increase the effectiveness of using our robot. For a company to see a significant improvement from using a telepresence system, it needs to be available to many people. The more people using telepresence, the more money the company can potentially save. From a business standpoint, the company must save more money by using the telepresence system than the system itself costs.

Android tablets are becoming ever more popular [3]. They are becoming faster, more efficient, and easier to use with time. These features make them a convenient fit as a platform for our development. The tablets are lightweight and have an extended battery life. This makes them more portable than laptops or desktop systems. Many tablets also have a data plan through a cell phone carrier. This allows access in public places or while traveling across town.



(a) (b) (c) (d)

Fig. 1. Hardware structure and software architecture of ASCC telepresence Robot.

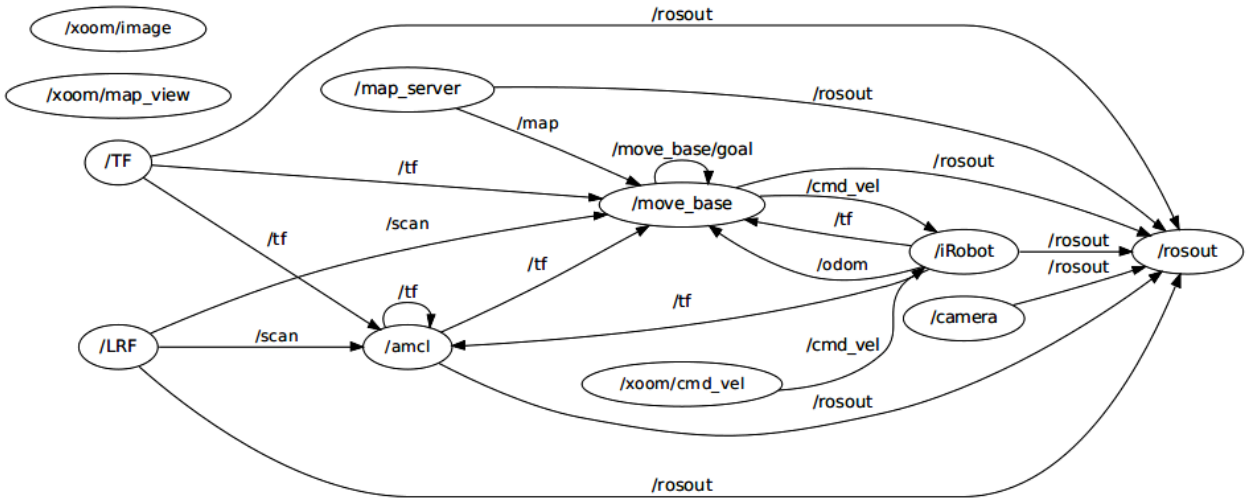


Fig. 2. ROS computation graph of software on the robot.

III. Previous Work

Over last decade, there have been many telepresence robots developed in various research projects related to remote communication focused on multimedia conference such as: tele-medicine, tele-education [4] [5], assistance or communication for home care of the elderly [6] [7]. Moreover, there are already some commercial telepresence robots in the market with a wide variety of applications ranging from ad-hoc conversations at the office, to patient rounds at medical facilities, and to students from school such as Willow Carage's Texal and PR2, Anybots' QB, VGo Communications' VGo, iRobot's AVA, InTouch Health's RP-7i, RoboDynamics' Tilr, HeadThere's Giraffe, MantaroBot, etc. By using those telepresence robots, from anywhere, with a computer and a Wi-fi connection, the operators can use them to hear, talk, see and be seen and move around a workplace from far away.

With the strong growth of the mobile devices such as smartphones, tablets, it is essential to develop telepresence robots capable of working with those devices. We can find some telepresence robots able to work with tablet such as WU telepresence robots [8] can work with iPad, Taxei and PR2 [9] from Willow Garage can work with Android Devices, MantaroBot TeleMe uses iPad2, Galaxy Tab 10.1, iPhone 4/4S for robot site. However, except Willow Garage's robots, currently, it is not popular to use android tablet in remote user for both controlling and video conferencing using the robots.

Most commercial robots fall into one of two distinct categories. Either they are aimed at the research market or at general consumers. These systems, though all very capable, are prohibitively expensive for consumers and hobbyists, many costing upwards of $10,000 USD [10] [11]. Research robots tend to be prohibitively expensive, lack application-level software, and have a daunting learning curve [8]. On the other hand, consumer robotics products, tend to be inexpensive, but are often engineered for a single purpose, lack sophisticated computation resources and sensors, and are not really designed for tinkering with [8]. There are numerous hobbyist-grade telepresence projects such as Sparky Jr.3, LabRatTM [12], which was designed to be an extremely inexpensive robot for classroom labs, hobbyist and research use. Unlike these robots, based on iRobot Create and Android Tablet, we are not trying to provide a commercial telepresence platform, or a super-low cost hobbyist robot, but rather a capable, extensible open development platform which can be configured for telepresence purposes at a cost of an order of magnitude less than what is currently available in similarly capable robots.

IV. Hardware

The telepresence robot system can be divided into two parts: the robot end and the remote user shown in Fig. 1. The robot end is ASCC telepresence robot built on an iRobot Create base with PVC pipes used to hold up a Tablet as a robot end monitor. It is capable of holding our attachments which feature a camera, a laser rangefinder, and a fit-PC2 netbook. It also has a battery aboard to power these devices. The camera mounted in iRobot Create for our project is USB Logitech QuickCam C250. This model is a kind of pinhole Camera. It is able to capture video up to 800x600 resolution, 24-bit true color, and 30 frames per second. The laser rangefinder is Hokuyo URG-04LX-UG01. It is a low-power laser rangefinder with wide-range up to 5600mmx240^0, and accuracy of ±30mm [15]. It is used to build the 2D map of the environment around the robot. A fit-PC2 Netbook running Robot Operating System (ROS) on Ubuntu 10.04 is used as our computing system on the robot. It has a dual-core 1.6 GHz Intel Atom processor, and 1 GB Ram [14]. It will control the robot's motion and collect sensory data from the camera, the laser rangefinder, as well as robot's poses. It also does wireless communications with the tablet for the tele-operator. On the remote user end, we will be using the Motorola Xoom as our Android based tablet. The Xoom has a dual-core NVIDIA Tegra 2 processor, 802.11n wireless capabilities, and a 10.1", 1280x800 display [18]. We will create a standard Android application that can be run from the Xoom using the Android SDK [17]. The Android SDK will allow access to all of the features of the Xoom using the Java Language.

V. Software

Our software development was essentially broken into two main sections. The first section was the software that was developed for the telepresence robot. The second section was the software developed for the Android tablet.

A. ROS

We setup ROS (Robot Operating System) run on Ubuntu 10.04 for running the robot's software. ROS, a Linux based software framework, is an open-source, meta-operation system for robots [13]. It uses the concept of packages, nodes, topics, messages, and services. It provides services similar to real operation systems, including hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes, and package management. The distributed computing feature of it can also facilitate multi-agent applications in a wireless network. In ROS, a program can be divided into different *nodes* which can be distributed to different computers in the same network. *Nodes* are separate processes which can receive and publish information from and to any other *nodes*. The driver of one component of the hardware can be treated as a node, while a data processing method can be made as one *node* as well. The information transferred between nodes is called a *message*. *Messages* are routed via a transport system called *topics* with publish/subscribe semantics. A node which sends messages on a topic is called a publisher and the receiving node called a subscriber has to subscribe the topic to receive that message. As an example of ROS *nodes, topics, and messages* is shown ROS computation graph in Fig. 3.

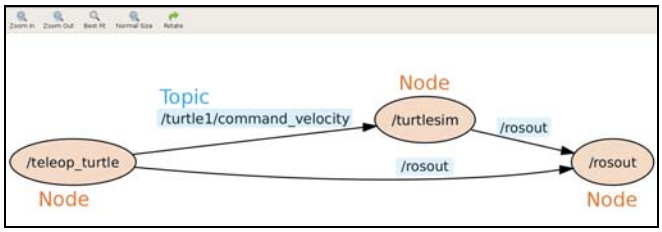


Fig. 3. ROS computation graph [13]

In ROS, all the *nodes* share one "ROS master". Nodes connect to other nodes directly. The ROS master only provides lookup information, much like a DNS server. Nodes that subscribe to a topic will request connections from nodes that publish that topic, and will establish that connection over an agreed upon connection protocol. The most common protocol used in a ROS is called TCPROS which uses standard TCP/IP sockets.

B. Telepresence Robot Software

Architecture of Software for our robot is shown in Fig. 1. For the most basic function in the Robot, we have the ROS version driver for iRobot create called irobot_create_2_1 from Brown University repository in ROS community[16]. Besides that we also utilized some exiting ROS packages for interfacing with hardware on robot such as hukoyo_note for hokuyo LRF, uvc_camera for webcam, tf for coordinate transformations, slam_mapping for create 2D map and navigation, move_base for motion planning. In addition to employing existing works, we designed and implemented our

own package to efficiently handle motion planning for the robot. Those allow the user to tell the robot where to go and it automatically moves to that location. This frees up the user's time because they no longer have to drive the robot around.

C. Android Software

The software written for our Android tablet combines two key libraries. The first key library is the Android SDK. The Android SDK is a free to use, open source operating system. It is specifically designed for use with mobile devices. The Android SDK has a plugin for development in the Eclipse IDE which makes it very simple to develop, install, and run applications on an Android based device. It uses the Java programming language for implementation.

We are creating an Android application to run our software. This way it can easily be integrated into an existing tablet, and could eventually be downloaded from the Android marketplace if it were decided to publish it. Our application could be downloaded and run just like any normal application on the Android Market.

The second key library that we used is RosJava. RosJava is a complete implementation of ROS in pure Java. It also has Android support, which made it an optimal candidate for our use. To use RosJava with Android, the android_core package must also be used. The combination of these two allows for complete integration with both ROS nodes and the Android SDK.

We investigated a few alternate options before settling on ROSJava. The most promising alternative was rospy. It is a python implementation that can be installed on android. You then use the Python language to run the code. However, it lacked integration into the Android SDK's interface abilities. It only allowed for some very basic dialog boxes and a terminal window.

To begin we had to get the RosJava code to build correctly. RosJava is still in alpha development, with updates being released regularly. It was a challenge to get ubuntu and the many packages of ROS we were using to all build correctly using the rosmake command. However, we eventually got the projects to all build. Conveniently, this outputs a grouping of Eclipse projects with all the necessary RosJava packages imported.

With an Eclipse project created, integrating ROS and the Android SDK was painless. From here, We integrated a portrait graphical user interface (GUI). With slightly more work, it should also work in landscape mode. Android has a series of prebuilt views that we were able to take advantage of, including an ImageView for the video, TextView for debugging output, and a HorizontalSlider for scaling the speed. RosJava implemented a RosImageView class which was capable of reading in a compressed image from the sensor_msgs/CompressedImage type in ROS, and displaying it to the interface. The ROS computation graph of whole software is shown in Fig. 2.

VI. Results

Through our work on this project, we were able to successfully implement all three of our goals:

1) Remote control of the robot
2) Live video stream from robot
3) Build 2D map from the laser sensor and develop motion planning feature for the robot using this map.

Utilizing RosJava and the Android SDK, we were able to communicate between our Android based tablet and the telepresence robot using ROS with ease. Once we found packages to support all of our hardware, and develop some software on our own, we could easily publish and subscribe to the necessary topics to communicate our data between nodes.

In our testing, we were able to successfully drive the robot around the laboratory. We were connected to a wireless local area network, giving us ample bandwidth. The video successfully streamed at 640x480 at 10 frames per second. There is a slight latency delay between the robots video feed and when it is displayed on the tablet. The telepresence robot also correctly built a 2D map that could be displayed. After some time, the green arrow representing the robot in the 2D map would lose its position and orientation. This is most likely due to the fact that it is using dead reckoning for its localization. Our results is shown in Fig. 4.

Fig. 4. Results of the project in first phase

VII. Future Work

As previously mentioned, we were forced to limit the goals of this project in order to fit the amount of work into one semester. This leaves work to be done in the future. This future work comes in a few different areas, all of which are necessary to create a fully developed telepresence robot.

The first area of work is to improve motion planning for the robot. It should have worked with larger 2D maps associated with the Simultaneous Localization and Mapping (SLAM) technique.

There is still a need to add a screen to the iRobot Create so that people can see the remote user's face when they are using the telepresence robot. Then, with the addition of an audio stream, the user should have two way visual and audio interaction with others. The tablet we are using has a front-facing video camera that could be used to retrieve the video, as well as a microphone for collecting the audio stream and speakers to play the incoming sound. The iRobot Create would have to be equipped with speakers and a microphone for these same reasons.

Another task that should be addressed is the video streaming capabilities. Currently we can smoothly stream 640x480 video at 10 frames per second. However, there is a noticeable delay of nearly one second. We have only tested our device on a local network so far, but that delay may grow when connecting through an internet connection. It would be beneficial to address this problem in order to provide an accurate and smooth video stream to the user, regardless of their location.

At the robot end, a hand gesture recognition algorithm based on computer vision is implemented for the people around the robot to control it. Imagining the following scenarios: the remote end user somehow loses control of the robot and it is still running; the remote user is not realizing the robot is too close or too far away from the people he wants to talk to; or the remote user wants the people around the robot to lead him to observe the remote environment. In the above scenarios, the camera-based gesture control at the remote end will be very useful.

VIII. Conclusion

In our research and development time, we have discovered that it is possible to create an affordable telepresence robot that can be controlled with an Android based device. This is important because many people already use Android tablets in their daily lives. With the addition of an affordable iRobot Create robot, they could have a complete mobile telepresence system at a very reasonable price.

With future developments on top of our current work, this cheap system should be a full featured telepresence robot that can compete with modern telepresence systems in both price and capabilities.

Acknowledgment

We all thank Dr. Sheng for his supervisions and valuable suggestions. Ha M. Do thanks Jeremy Evert and Gang Li for their kindly assistance in our installation iRobot create. Craig J. Mouser thanks Damon Kohler for his assistance in successfully building the RosJava packages.

References

[1] G. Walker and P. Sheppard, “Telepresence - the future of telephony”, BT Technology Journal, vol. 15, no. 4, October 1997.

[2] Clay Dillow, “Children's Hospital Boston Sends Telepresence Robots Home With Post-Op Patients”, http://www.popsci.com/technology/article/2011-12/childrens-hospital-boston-turns-telepresence-robots-post-op-patient-care, Dec 2011.

[3] Rob Waugh, “Resistance is futile! More Androids are activated every day than babies are born”, http://www.dailymail.co.uk/sciencetech/article-2086144/CES-2012-More-Androids-activated-day-babies-born.html, January 2012.

[4] C.-fen Shih, C.-wei Chang, and G.-dong Chen, “Robot as a Storytelling Partner in the English Classroom - Preliminary Discussion,” in *Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on*, 2007, no. Icalt, pp. 678 – 682, 2007

[5] O.-Hun Kwon, S.-Yong Koo, Y.-Geun Kim, D.-Soo Kwon, “Telepresence Robot System for English Tutoring,” *Advanced Robotics and its Social Impacts (ARSO), 2010 IEEE Workshop on*, pp. 152 - 155, 2010.

[6] T.-C. Tsai, Y.-L. Hsu, A.-I. Ma, T. King, and C.-H. Wu, “Developing a telepresence robot for interpersonal communication with the elderly in a home environment”, *Telemedicine journal and e-health : the official journal of the American Telemedicine Association*, vol. 13, no. 4, pp. 407-24, Aug. 2007.

[7] H. Gross et al., “Progress in Developing a Socially Assistive Mobile Home Robot Companion for the Elderly with Mild Cognitive Impairment,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 2430-2437.

[8] D. A. Lazewatsky and W. D. Smart, “An Inexpensive Robot Platform for Teleoperation and Experimentation,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1211-1216.

[9] Robots, http://www.willowgarage.com/robot/overview, 2011.

[10] Anybots Inc. Anybots - your personal avatar. http://anybots. com/#qbLaunch, August 2010.

[11] Erico Guizzo. When my avatar went to work. IEEE Spectrum, September 2010.

[12] M. Desai, K. M. Tsui, H. A. Yanco, and C. Uhlik, “Essential Features of Telepresence Robots”, *Technologies for Practical Robot Applications (TePRA), 2011 IEEE Conference on*, pp. 15-20, 2011.

[13] ROS wiki, “http://www.ros.org/wiki/,” 2011.

[14] Fit-PC2, “http://www.fit-pc.com/web/,” 2011.

[15] Hokuyo laser, “http://www.hokuyo-aut.jp/,” 2011.

[16] Brown University, “http://code.google.com/p/brown-ros-pkg/,” 2009.

[17] Android SDK, "http://developer.android.com/sdk/index.html," 2011

[18] Motorola Xoom, "http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Tablets/ci.MOTOROLA-XOOM-with-WiFi-US-EN.alt#anchor", 2011.