Multi-floor Mapping and Navigation with Uncertainty

Dr. James Ellingson, U. of St. Thomas, School of Engineering

James Ellingson earned his Ph.D. Mechanical Engineering at the University of Minnesota. He joined the Faculty in at University of St. Thomas in 2009 after an extensive career in medical device manufacturing and industrial automation. Research interests include remote sensing, autonomous vehicles mechatronics, embedded systems, machine design and robotics.

Kundan Nepal, University of Saint Thomas

Kundan Nepal is currently an Assistant Professor in the School of Engineering at the University of St. Thomas (MN). His research interests span the areas of reliable nanoscale digital systems, mobile robotics and reconfigurable computing.

Megan Rose McGill, University of St. Thomas
Mitchell J Hoffmann, University of St. Thomas

Multi-Floor Mapping and Navigation with Uncertainty

Abstract

This paper outlines the research using autonomous robots conducted by a group of undergraduate engineering students from the University of St. Thomas. The students were able to develop a mutli-floor mapping and navigation system that allowed a robot, named Turtlebot, to give nearly autonomous tours of their multi-floor engineering building. The Turtlebot can effectively traverse the hallways, use the elevator, and play audio files during tours.

Introduction

Worldwide, robots are becoming further integrated into everyday life. Because robots can be programmed to analyze, act within, or control almost any environment autonomously, robots are valuable assets for tasks ranging from cleaning a cat's litter box to flying an aircraft. An important aspect of robots is their ability to solve complex problems quicker, more efficiently and with more precision than humans can attain. For this reason, robots are useful tools in applications requiring heavy computation such as weather forecasting[1] or natural disaster detection[2,3]. In addition to performing difficult calculations with ease, robots are also capable of performing tasks that are either too intricate, such as small-scale surgeries[4], or too strenuous, such as automobile assembly[5], for humans to perform. For these reasons and countless others, the field of robotics and the wide variety of applications that it encompasses will continue to grow. It is beneficial, therefore, to educate and excite young minds about the wonderful opportunities available in this field of study. In this paper, our undergraduate research team hopes to outline the research we conducted over the course of a summer and to propose that our project could be expanded upon in further research projects or as part of an undergraduate engineering curriculum.

Our project, which began with the intent of programming a robot to give guided tours of our engineering department, allowed us to explore many topics in robotics including multi-floor mapping and navigation. Using a Turtlebot[6], as shown in Figure 1, and the Robot Operating System (ROS) environment, our team successfully developed a mapping and navigation system capable of guiding a robot through the hallways and elevator in our engineering department.

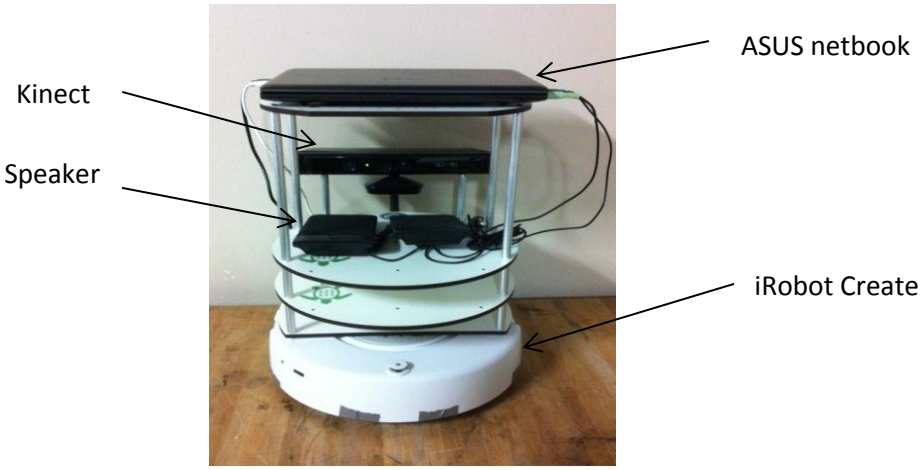

Figure 1:
The Turtlebot robot system integrates the Kinect sensor, the iRobot Create platform, an ASUS netbook computer, and speakers.

Project Setup

As mentioned above, our team chose a Turtlebot™ robotic system (shown in Figure 1), a product of Willow Garage Inc., for our mobile platform. We chose this platform because of its modest cost and its capability to easily interface with ROS. The Turtlebot system utilizes the Microsoft Kinect™ sensor array (shown in Figure 2), iRobot™ Create™ mobile base, and a netbook computer that runs the open-source Robot Operating System (ROS) housed within Ubuntu to map and to navigate dynamic and busy environments --- such as the school hallways that we wished to map.

Figure 2:
Microsoft Kinect sensor system

One of the Turtlebot's key components is the Kinect sensor array. Contained within the Kinect sensor is an infrared depth scanner, a color camera and, a four-element microphone array[7]. The depth scanner uses an infrared laser combined with a monochrome CMOS sensor to gather 3D video data in any ambient lighted environment. The sensing range can be adjusted either manually or automatically. The Kinect sensor array has 10 bits of resolution and a practical range

of 1.3 to 20 feet in distance from the object it is sensing. The color imaging camera supports a variety of formats, all computed from the same raw data. We used the 32 bit linear RGB video stream. It uses 8-bit color (X8R8G8B8) VGA (640 x 480) resolution. 16 bit YUV and 32 bit Bayer color formats are also supported [7]. Due to the Kinect sensor's inability to detect objects within its minimum range of approximately 0.4 m, the sensor is mounted toward the rear of the Turtlebot. This position has the added benefit of promoting optimal lighting conditions for the Kinect. The Turtlebot was designed so that the top shelf shields the Kinect sensors from excessive light. This allows the Kinect to obtain the best possible data while preventing other components of the Turtlebot from obstructing the Kinect sensor.

As shown in Figure 3 below, sensor data is preprocessed by the Kinect and then sent to the master controller. The ASUS net book computer on the Turtlebot is running the Ubuntu operating system, version 10.10 (Maverick Meerkat). The master program controlling the Turtlebot is written in the Open Source Robot Operating System (ROS). The ROS environment is a system that allows for real-time running of multiple programs across a network of computers to help a robot or group of robots function[8]. ROS structures all the messages being sent between computers in an easy-to-understand system wherein "slave" computers publish "messages" to their specific "topics", and a single "master" computer organizes all "topics" [8]. The "master" computer, in our project is usually the robot itself, so all messages sent across the network would go through the ASUS netbook. The "slaves" can either be workstations or other robots. ROS "topics" are organized conversations of similar message types[9]. One particular message type is cmd_vel which is a simple message dictating how much a robot should turn or drive. All messages sent from the ROS network are organized under appropriate topics. Any computer within the network can read from, publish to, and act upon the messages in topics. In the case of cmd_vel messages, the cmd_vel messages are organized under the cmd_vel topic, and can be read from any computer on the network. The block diagram below in Figure 3 shows how the ROS network unifies the Turtlebot and other slave computers.

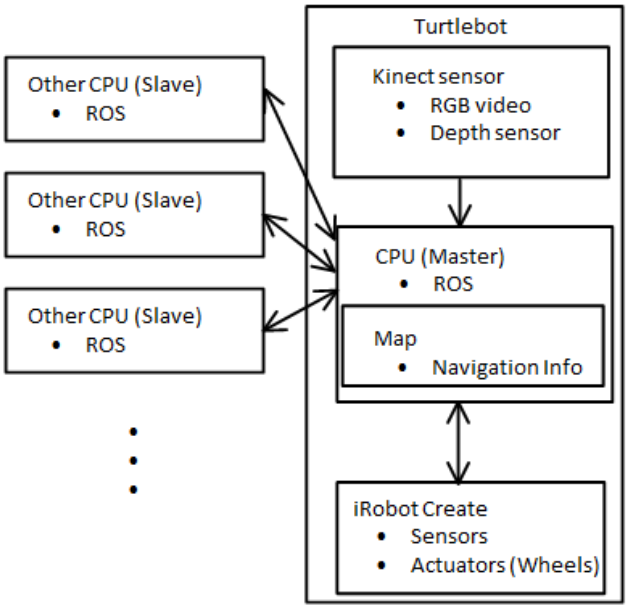

Figure 3:
Block System Diagram for the Turtlebot and the base (slave) CPUs

A useful analogy for the operation of theTurtlebot is to compare it with the human body. The Kinect senor array functions as the "eyes" and "ears" of the Turtlebot, while the Create provides the "feet" for locomotion as well as some "tactile" feeling. The ROS is the "brain", processing the senor data and directing the motion. The Kinect sensory array, the "eyes" of the system, take in visual input such as RGB video feed and a depth array. The iRobot Create, the "legs", provide sensory information from dedicated sensors such as the bump sensor, drop sensor, and wall sensor that detect unexpected changes in the environment as well as encoders on the wheels to help determine distance traveled and orientation. The "legs", like most legs, also provide wheels for movement. Then, using the analogy of a brain, ROS organizes all the sensory inputs, interprets the signals, and responds with appropriate commands. Using programs written in C++, Java, Python, and even to some extent, bash-scripting, the Turtlebot can be instructed to execute of a variety of tasks. With the sensors and processing ability available, the Turtlebot is well-suited to interact with humans and travel in a spacious indoor environment, as needed for giving tours of our engineering department.

Mapping and Navigation

Out-of-the-box, the Turtlebot has the ability to make maps of its "new" environment. The first time in a given environment, the Turtlebot cannot navigate because it lacks understanding of the environments' boundaries. The Turtlebot can be driven around while running a built-in mapmaking program, and with the Kinect's depth sensor, gather data and determine where walls

and other impassable objects are. The map begins as completely unknown, save for a small wedge that is visible in front of the Turtlebot. As the Turtlebot explores the new terrain, it fills in areas on the map that can be driven though with a certain color and draws lines in another where the depth scanner had detected a solid object. There are three colors that can be found on the map: white, which represents open areas that can be freely traversed by the Turtlebot, black, a color signifying impassable objects such as walls, and grey, the color used to express unknown territory that has not yet been explored and categorized. When the operator of the Turtlebot is satisfied with the map, the mapping program finishes by saving the map in two parts: a portable graymap and a descriptor file that helps interpret the graymap for future use. These maps cannot be changed later, so it is critical to make sure the map is well-made before stopping (see Figure 4 below).

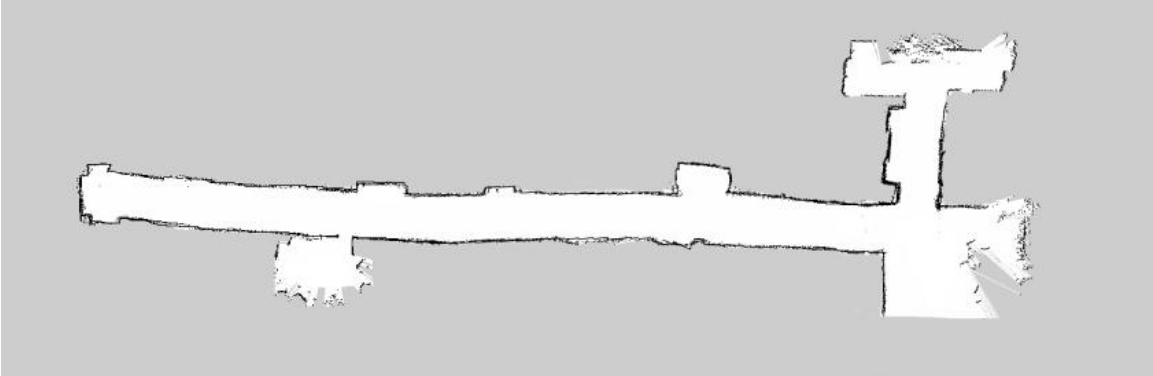

Figure 4: A raw map of the Engineering hallway created by the Turtlebot.

Although the Turtlebot is capable of mapping a single-level, simple environments, it cannot reach another floor without assistance. Since our team's goal was to have the Turtlebot give automated tours of our engineering department, which spans multiple floors, we decided to have the Turtlebot use the elevator system as a means of getting to the next floor. Though the elevator is traversed countless times each day by students and faculty, programming the Turtlebot to use the elevator proved much more difficult than one would anticipate.

The elevator bank poses several challenges to the Turtlebot. First, since the elevator doors are normally closed, the mapping program does not recognize them as passable space. Second, because our school's elevator bank has two doors, and we do not know which door will open first, the Turtlebot must position itself so as to detect an open door and move through it. Finally, once inside the elevator, the Turtolbot loses wi-fi connection and can, therefore, no longer communicate to other nodes.

Because the Turtlebot lacks a definition of "door" and cannot tell if one is open or closed as a person can, the Turtlebot needs assistance in determining when it is able to pass through a given threshold. Our elevator bank has two doors. In order for the Turtlebot to recognize if either door is open, it must be able to see both doors. Identifying this position from which both doors are

visible on the map is trivial. Accurately and repeatably positioning the Turtlebot at the required location is not trivial.

As observed in Figure 4, straight walls are not created perfectly straight on the Turtlebot's maps. The Turtlebot navigation and data are subject, therefore, to a level of uncertainty. Due to the accumulated error, the Turtlebot was sometimes not facing both doors in the elevator bank when its destination was reached. Additionally, because the Turtlebot relies upon the agreement of sensory data and previously created maps to self-locate, the Turtlebot can get lost if the elevators are open while it aligns itself in front of the elevator bank. Thus, we encountered our first real obstacle to achieving our goal: positioning the Turtlebot in full view of both elevator doors quickly upon arriving in front of the elevator bank.

This obstacle was overcome by using a simple image processing technique. Since the elevator doors in our department are darker than the surrounding walls, we were able to convert the image of the doors from a color image to one in only black and white. Edge detection is greatly simplified since the doors to become clearly distinguishable from everything else in the monochrome image. Then, we used our understanding of where the doors—represented as black pixels—would be located in relation the picture, to write a short program to find where in the picture the highest concentration of black pixels were. If the doors were in the wrong spot, the Turtlebot would adjust itself in the correct direction. The only problem was that the program ran slow, since it took a good deal of time to count pixels. We overcame this problem by counting only five rows of pixels, rather than all of the door's pixels, in our program. Note the edges we are detecting cover the entire field of view, further simplifying our image problem to a line-scan problem. That is, we only needed to consider a few scan lines of data to discover where a doors lay in relation to the image, since some part of the door was always visible near the vertical center. With this improved strategy implemented, the Turtlebot could arrive at the correct position orientation quickly and repeatedly.

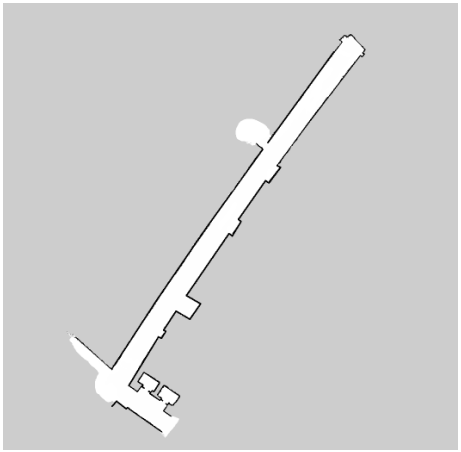

Figure 5: Edited map showing the (added) elevators

Once we had the orientation problem fixed, we realized that there was another problem: when an elevator door opened, the Turtlebot was unsure whether to trust its map which stated that there was still a solid wall, or its sensory data which detected an open space. It occurred to us that the uncertainty factor of the environment, while being a continual issue in this project, could also be a viable solution to our current problem. The grey, or "unknown" area, that was marked on the map was treated by the Turtlebot as having the potential to be an open route but, because it was unknown territory, the robot was wary of obstacles that could potentially lie within that space. We were able to edit the map, see Figure 5, to include the insides of the elevators, separating them from the main hallway with a line of grey "unknown" space taking the place of the doors, as shown in Figure 6. Because of this, the doors could be either open or closed, and the Turtlebot would deal with either case accordingly. While moving around near the elevators, the doors can either be opened or closed and the sensory information will not disagree with map, and thus the robot will not be confused while positioning itself or deciding whether it can enter the elevator.

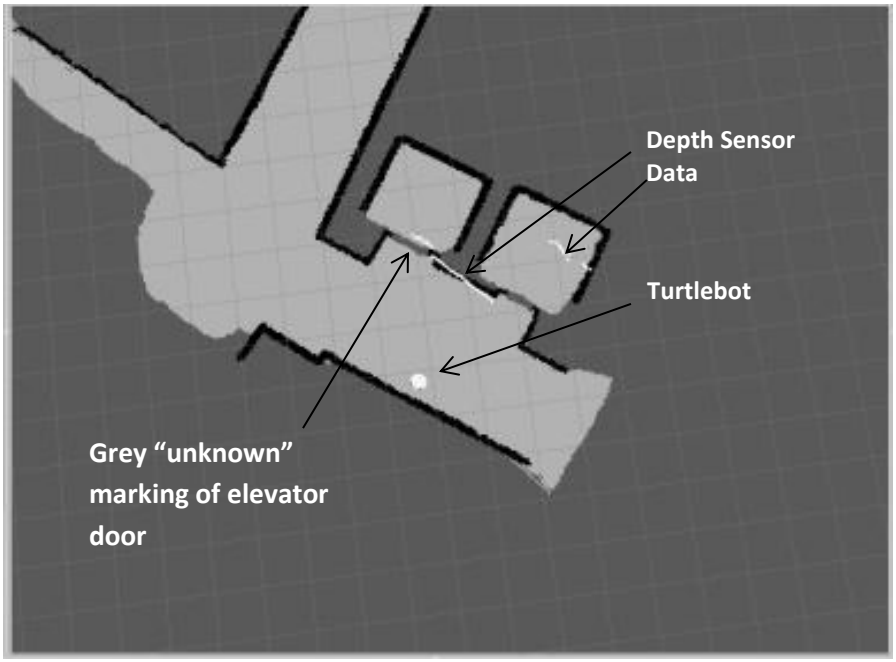

Figure 6: ROS image of depth scanning with right door open, displayed in the edited map.

Using the two new programs, the robot could now travel to the elevators, position itself so that both doors were in clear view, wait for a door to open, determine which door opened, and move inside the open elevator. This operation is executed with a single command from our base workstation. Upon the robot entering the elevator, however, we stumbled upon the third problem. In order for us to continue giving commands to the robot, our workstation required constant wireless connection with the Turtlebot. While in the elevator, the wireless connection is lost. To achieve multi-floor navigation, we had to find a solution to this problem.

Our solution to this new problem was to allow the Turtlebot to initiate and function in "stand alone" mode. Instead of the workstation always giving commands to the Turtlebot, we wrote a new program that launched a shell on the base station in order to terminate communications with the Turtlebot once the Turtlebot had reached the elevator bank. Once the desired floor had been reached and the robot had exited the elevator, wireless access was regained and the Turtlebot would command the workstation to reinitiate communications as they were before entering the elevator. This program worked concurrently with the ones already in place, so this proved an effective solution to our wireless problem.

Discussion and Future Work

In the future, we hope to find a successful way to load new maps upon arriving at new floors. After accomplishing this task, multi-floor navigation with the Turtlebot will be a success. A whole new host of opportunities are available in robotics with multi-floor navigation capacity. With what we've done so far, the Turtlebot could handle a large variety of multi-floor tasks autonomously, such as giving tours. With the ideas we have implemented thus far, programs could be written for numerous mobile robotic systems in fields varying from the medical industry to military applications.

In addition to swapping maps, we could improve on our system by attaching and controlling arms on the Turtlebot to totally eliminate the need for human assistance when opening doors[10]. Because our robot currently lacks an arm, a human is required to push elevator buttons for the robot. The possibilities for expanding upon our project are endless, just like the field of robotics. As undergraduate students, this project has opened our eyes to many opportunities, new ideas, and new challenges with which to prepare ourselves for whatever we may encounter in the working world. We believe that either further research or a course focusing on the ideas explored in our project would be both educational and exciting to students. Projects such as this have the potential to educate students about components such as the Turtlebot while providing a real-world application. It has been a wonderful experience expanding our horizons and working together on topics that encourage such creativity and ingenuity.

Works Cited

[1] Arampatzis, Th, J. Lygeros, and S. Manesis. "A survey of applications of wireless sensors and wireless sensor networks." In Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation, pp. 719-724. IEEE, 2005.

[2] Viguria, Antidio, Ivan Maza, and Anibal Ollero. "Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions." Advanced Robotics 24, no. 1-2 (2010): 1-23.

[3] Merino, Luis, Fernando Caballero, J. R. Martinez-de Dios, and Aníbal Ollero. "Cooperative fire detection using unmanned aerial vehicles." In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp. 1884-1889. IEEE, 2005.

[4] Troccaz, J., and Y. Delnondedieu. "Robots in surgery." In Proc. Int. Symposium on Robotics Research, pp. 155-166. 1996.

[5] Yamada, Y., Shigetaka Nagamatsu, and Yoshimasa Sato. "Development of multi-arm robots for automobile assembly." In Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on, vol. 3, pp. 2224-2229. IEEE, 1995.

[6] Garage, Willow. "Turtlebot." Website: http://turtlebot. com/last visited (2011): 11-25. (accessed March 14, 2013)

[7] Kinect for Windows Website: http://msdn.microsoft.com/en-us/library/hh973075.aspx (accessed March 14, 2013)

[8] Robot Operating System (ROS) Wiki Website: http://www.ros.org/wiki/ (accessed March 14, 2013)

[9] Quigley, Morgan, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. "ROS: an open-source Robot Operating System." In ICRA workshop on open source software, vol. 3, no. 3.2. 2009.

[10] Chitta, Sachin, Benjamin Cohen, and Maxim Likhachev. "Planning for autonomous door opening with a mobile manipulator." In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1799-1806. IEEE, 2010.