The Creation of a Low-Cost, Reliable Platform for Mobile Robotics Research

by

Taylor Harrison Gilbert

Submitted to the Department of Mechanical on May 6, 2011 in partial fulfillment of the requirements for the degree of

Bachelor of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

Author .....
Taylor Gilbert
Department of Mechanical Engineering
February 1, 2010

Certified by .....
John Leonard
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.....
John H. Lienhard, V
Samuel C. Collins Professor of Mechanical Engineering
Undergraduate Officer

The Creation of a Low-Cost, Reliable Platform designed for Mobile Robotics Research

by

Taylor Harrison Gilbert

Submitted to the Department of Mechanical Engineering
on May 6, 2011 in partial fulfillment of the requirements
for the degree of Bachelor of Science in Mechanical Engineering

Abstract

This work documents the planning process, design, fabrication, and integration of a low-cost robot designed for research on the problem of life-long robot mapping. The robotics platform used is the iRobot Create. This robot also employs the PrimeSensor, a sensor with the ability to provide a pixel-matched, colored depth field in real time. This sensor was later purchased by Microsoft and leveraged in their popular gaming device, the Microsoft Kinect. The robot has a powerful Acer Aspire 1830T-6651 laptop with an Intel Core i5 to perform processor-intensive, real-time image processing.

The actual construction of the robot consisted of two phases: the physical integration of the components on a chassis and the software integration through the computer. The physical integration is mainly a central chassis made from laser-cut acrylic. This chassis is capable of securely holding the laptop computer in place and provides an elevated mount for the PrimeSensor. This mount has the ability to change the viewing angle of the sensor and lock that angle at 5° increments using a pin.

The software integration was completed using open-source packages for the Robot Operating System (ROS) developed by Brown University and a not-for-profit company called OpenNI. These packages were installed on the onboard laptop and the ROS core functions running on the laptop provide the foundation to run new code on this testing platform.

This robot is low in cost and provides a reliable, robust, and versatile platform for vision-based artificial intelligence research. The mapping software and vision algorithms developed on this platform will contribute to the development of more intelligent and meaningful vision capabilities for tomorrow's robots.

Thesis Supervisor: John Leonard

Title: Professor of Mechanical Engineering

Acknowledgements

Professor John Leonard for genesis of the idea, for support during the testing of the apparatus, and feedback on written work.

Hordur Johannsson for software help and for providing direct access to a researcher in the field for which this robot is intended. He provided much of the feedback on design specifications and goals of the system.

Garratt Gallagher for his previous work on similar platforms, namely Bilibot, and for his assistance during the initial troubleshooting stages. He also assisted with the initial problem solving and providing a context in which we could start our research.

Sam Reineman for his direct assistance with the computer system selection, installation of software, and during the physical design and fabrication of the robot.

Table of Contents

List of Figures

1. Introduction

The goal of this research is to develop an affordable, versatile, and robust platform for the testing and development of cutting edge artificial intelligence software. The primary use of the robot developed during the course of this study is to autonomously navigate interior spaces using an onboard computer and a sensor developed by the Primesense Company of Tel-Aviv, Israel. Through real-time computation of sensor data and compilation of this data into a continuously-updating map of the surroundings, the system will be able to perform Simultaneous Localization and Mapping.

1.1. Motivation for Research

1.1.1. Differentiation of Skills

At the level of complexity of today's artificial intelligence research, specialization of the duties of roboticists is essential. The differentiation of mechanical engineers, electrical engineers, and computer scientists provides a simple model for the creation of this robotics platform. A problem commonly faced by individual researchers or research groups is the lack of communication or availability of resources falling into each of these three categories, so this piece of research is an attempt to bridge that gap.

Too frequently the complex software written by artificial intelligence computer scientists cannot be fully or reliably tested, because the physical platform this computer scientist has developed is poorly constructed or unreliable. The software being developed is novel and has flaws of its own, but the valuable time of the artificial intelligence scientist is spent debugging not their algorithm or code, but mechanical shortcomings of the physical platform on which they are testing their software. Having the ability for a computer scientist to test and improve their software without the hassle of debugging mechanical problems will speed the rate of software development.

1.1.2. Affordability of Research Platform

When individual robots are assembled to serve a specific robotics research goal, their parts may have each been special-ordered and the design of the robot will be tailored to serve one use. However, when parts need to be serviced, replaced, or upgraded, the time spent researching the methods and practicality of these actions is further time an artificial intelligence researcher need not spend away from their work.

Additionally, with such purpose built robots, if the field of research or funding shifts away from the intended purpose of the robot, either attempting to adapt the robot for the new purpose or simply creating a new platform are expensive and time-consuming processes. Though adaptation seems faster and more cost-effective, the time spent later debugging incompatibilities from one platform to the next may very well outweigh the cost of simply creating an entirely new platform for each research goal.

1.2. Simultaneous Localization and Mapping

This section provides a brief background on the branch of artificial intelligence concerned with Simultaneous Localization and Mapping (SLAM), then specifically the problem of life-long mapping.

1.2.1. An Overview of SLAM

The goal of robots attempting SLAM is to build an onboard map of an unknown environment while simultaneously using this new map to navigate through the environment. Fundamentally, this is what human beings and other animals achieve on a continuous basis. An example of human SLAM is entering an unfamiliar room, locating a door, and proceeding through it. The human algorithm begins first by using predominantly the eyes to survey the new space, and establishes a preliminary mental map of wall locations and the locations of prominent objects in the room. In this case the human also identifies a door in the room, and modern-day humans are trained to know that more often than not, a door leads to a new space. Determining the human's location in the room and his or her location relative to other objects in the room is Localization.

Once the human has determined his or her location and the locations of objects and features of the room, a more detailed mental map is constructed. If the goal of the human is to completely explore the space, this is the time that using the newly constructed mental map, the human calculates a suitable path through the room to the door. At this point, the map is still incomplete, and the path the human begins to pursue is still a best guess. If the human encounters an end table obscured by a couch upon crossing the room, this table is entered into the human's mental map, and a new path must be planned and executed. This is the process of continuous Mapping and path planning.

1.2.2. Current State of Research: Life-long Mapping

Stated above are examples of three different tasks a robot must perform to navigate a room: localization, mapping, and path planning. In general the research body has split their efforts between SLAM researchers and path planning researchers, as the needs and goals of each research are different. SLAM researchers are attempting to input an unknown physical space and create a digital map of that space in real-time. Path planning and obstacle avoidance researchers take that digital map and optimize a route from one mission point in the map to another point in the map. While these tasks may be coupled in various implementations of robotics platforms, the aim of researchers today is to decouple them in both study and implementation.

Life-long mapping is the sub-problem of SLAM whereby the map can be saved and used again over the course of a long duration of testing. This area of study is concerned with optimization of data storage and correction of drift in presumed position based on object recognition in the existing map. The problem of creating a map is essentially solved; the current area is finding practical and efficient ways to use the data for the life of a robotics system.

1.2.3. Primesense and Kinect

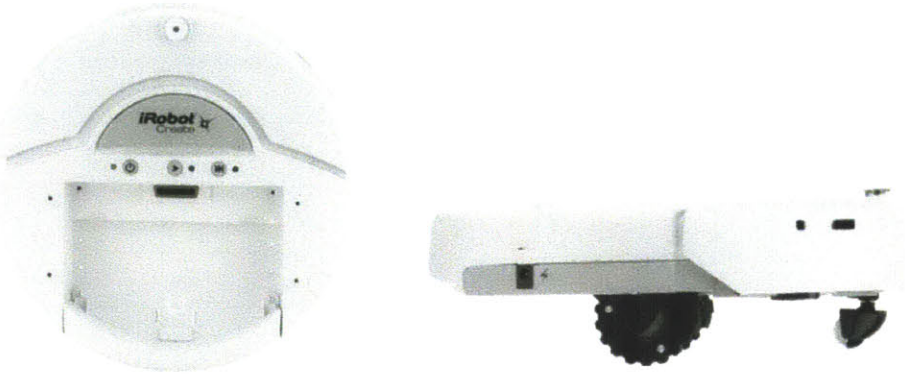
Recently, new tools to help SLAM researchers have become available at low cost and with high reliability. The Primesense Company originally created the technology later licensed in the United States by Microsoft for use in their Kinect controller as an add-on to their popular Xbox 360 gaming platform. These sensors, leveraging fundamentally the same technology, create a colored 3D point cloud of information, based on coupled visual camera and infrared depth sensing camera technologies. The details of this sensor will be discussed further in this work, but the impact of these technological advances for SLAM research have already been, and will continue to be profound.

2. The iRobot Create as a Robotics Platform

The iRobot Corporation, a commercial robotics company from Bedford, Massachusetts, produces a multitude of robots, ranging in size and application. One such platform has two different flavors. Their niche housework helper, the Roomba, is an autonomous vacuum cleaner, featuring a front mounted bump-sensor to map and vacuum an entire room. iRobot realized that the chassis they had commercially developed for household chores also had a market in the larger robotics research world. The company currently sells the Roomba platform without the vacuum accessories under the name Create. This platform provided the mobility and chassis for the robot created during this research.

2.1. Robot Capabilities and Features

The Create itself has several configurations, and provides several locations for the addition of supplementary modules to the robot, as shown in Figure 1.



A. Top View B. Side View

Figure 1. The iRobot Create

The Top View (A) of this robot shows the basic buttons for user interaction with the robot, and at the bottom of the photo, the bay for transport of cargo or the addition of extraneous modules is visible. The upper semi-circle in this view is the robot's bump sensor for alerting the robot when striking walls or other objects. The Side View (B) shows the robot with 3 of the 4 wheels in place, but with all wheels in the downward position. When the robot is placed on the ground, these wheels collapse into the body, giving the robot a lower profile and providing feedback that all wheels are touching solid ground. Source: iRobot.com

2.1.1. Features for Navigation

The Create has an infrared sensor mounted at it's front, for automatic return to charging base or to stop the Create if it detects the crossing of a virtual wall. Supplementary to the Create and Roomba are small infrared transmitters that can create a plane of infrared light, designed to be placed at doorways or in front of dangerous areas that the robot should not travel.

The robot also features a 180° bump sensor to alert the robot when a physical object is obstructing its path. Underneath this bump sensor, there are four height sensors, which bounce infrared light from the ground into a detector. These sensors are capable of detecting whether the ground below the sensor is at the appropriate height for the robot to drive in this area, and if a sensor detects that the height is too low (i.e. a set of stairs or the edge of a desk is approaching) the sensor will alert the robot's onboard computer to the risk.

Each of the robot's three, or optionally four, wheels is spring-loaded and will detect if the robot has left solid ground by extending into the free position (shown in Figure 1, B). The purpose of this capability is two-fold: first to alert the robot of the ground dropping away if one of the height sensors has failed to detect it, and second to alert the robot if a user has picked up the robot, or if some outside force has caused it to fall over, releasing the wheels from the ground.

2.1.2. Features for Expansion

The create has a cargo bay toward the rear of the robot, accessible from above, which originally housed the Roomba's vacuum module. In the case of the Create, as this is no longer desired functionality, this bay is made available for adding additional computing modules, the carrying of cargo, or in the case of this piece of research, the wires and additional connecting accessories that link the components of this robot together.

In the cargo bay, there is a serial connecter designed to be used with the iRobot Create Command Module, an optionally purchased module to assist in development of code using fully capable programming languages developed on a computer, rather than using the simple scripting commands available on the standard onboard computer. Our implementation uses a software shell to simulate the command module, which is discussed in the following section.

There are also four screw holes on the top of the body of the Create, designed to permit the attachment of custom modules such as the module used in this research.

2.2. Advantages of Commercial Production

As mentioned in the Introduction of this work, there are many advantages to using a standardized platform that is produced commercially. In terms of product support and part availability, having a customer service center for the robot on which your software is being tested is a vital resource in saving the time of artificial intelligence researchers. Additionally, having a common platform on which others in the field are developing means that the researcher

has other scientists at his or her disposal as resources on common problems caused by the system.

2.2.1. Affordability

Moving from a custom-built system, likely assembled in a lab by a researcher, to a commercially manufactured robot marketed to a wide and populous user base has many advantages. When a company such as iRobot aims to earn a profit on a product, it is in their best interest to make the unit affordable to consumers, easy to service, and have modular, replaceable parts. By spending the upfront money on design for manufacture and design for assembly, they can use their robust design to produce many units at a lower cost. In general, as the number of units produced increases, the cost of production per unit decreases (known as "economies of scale"). iRobot has the incentive to make the Create a low-cost, simple platform, and to design in a robust way to minimize maintenance costs.

The current (May 2011) retail cost of the Create is just under $130, so for any research goal, this platform is certainly an affordable and robust option.

3. Robot Operation System

The Robot Operating System (ROS) is the operating system that was chosen for research on this platform. Packages exist to encapsulate the iRobot Create and the Primesense sensor already, so the implementation of autonomy software should require little additional software development.

3.1. History and Background

The Robot Operating System (ROS) is an open-source (BSD) language for the development of robotic functionality on a variety of hardware platforms. ROS is has a steadily growing base of users and contributors, and all downloads can be found at `http://www.ros.org`.

ROS is based on Switchyard, a software package developed by Stanford's Morgan Quigley. The development of ROS has only grown and gained more users since its first release in 2007. The development of ROS from Switchyard has occurred primarily at Willow Garage, a robotics research institution committed to open-source robotics research on a common platform, and continues today with the release in March 2011 of ROS Diamondback, the third major release of ROS since its first distribution release in March of 2010.

3.2. ROS Architecture

In general, the ROS is a peer-to-peer "graph" of processes that communicate using the ROS communication infrastructure. ROS is designed to allow for hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing, and package management. There are three levels of ROS concepts, covered below.

3.2.1. ROS Filesystem Level

The ROS Filesystem Level is where the basic code elements of ROS exist. There are packages of software that may contain datasets, configuration files, or runtime files to provide the basic software-level implementation of a researcher's code. There are many packages available from `ros.org` for download and this list is only growing as ROS becomes more accepted as a standard operating system for robotics research. Stacks of packages are also contained on this level. A stack of packages is an aggregated tree of dependent packages that can be grouped to assist with a single category of tasks, such as navigation.

3.2.2. ROS Computation Graph Level

This level of the abstraction of the ROS structure is where the inter-process communication occurs. The peer-to-peer network of processes running in any single ROS environment must use the native message system and Parameter Server for publishing and subscribing to messages. The primary reason to use a peer-to-peer architecture at this level is to decouple various processes from each other. For instance, there may be a process running a vision sensor, and a process running the path planning, but these two processes will remain separate and modular for easy debugging and overall system stability.

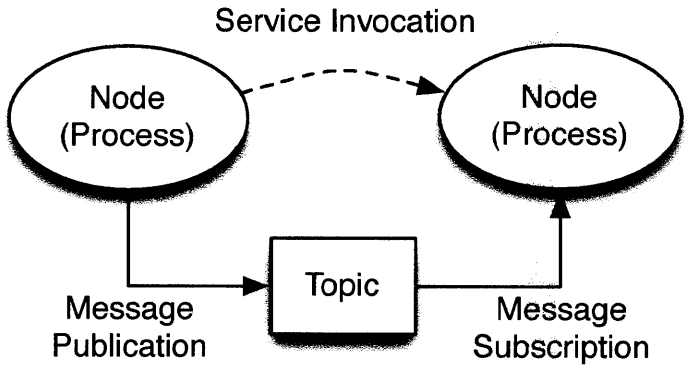


Figure 2. Basic peer-to-peer ROS communication implementation

This is an abstraction of a ROS peer-to-peer communication between two processes (Nodes). The left process publishes a message of a given Topic to the messaging system built into the ROS core, and the right process, which has previously subscribed to messages of that Topic from the ROS core, receives a copy of that message, whether it contains location data, a command, or other forms of message information.

3.2.3. ROS Community Level

Put quite simply, the ROS Community is the entire global community of ROS users and developers. This level of abstraction manifests itself primarily on `ros.org`, allowing for the collaboration of any ROS users and for the posting and maintenance of any code resources. All ROS packages, stacks, and documentation exist in this repository and as ROS is a free, open-source utility, this is the location all new users can go to obtain the latest release for installation on their system.

4. Primesense PrimeSensor and Microsoft Kinect

The PrimeSensor, developed by Primesense of Tel-Aviv, Israel, is an integrated vision system tailored for use with personal electronics to act as a gesture-based user-interface experience. This technology is novel in its depth and color sensing and has been successfully adapted to suit many applications.

4.1. Technology behind the PrimeSensor

The core of the PrimeSensor consists of the three optical devices depicted in Figure 3.

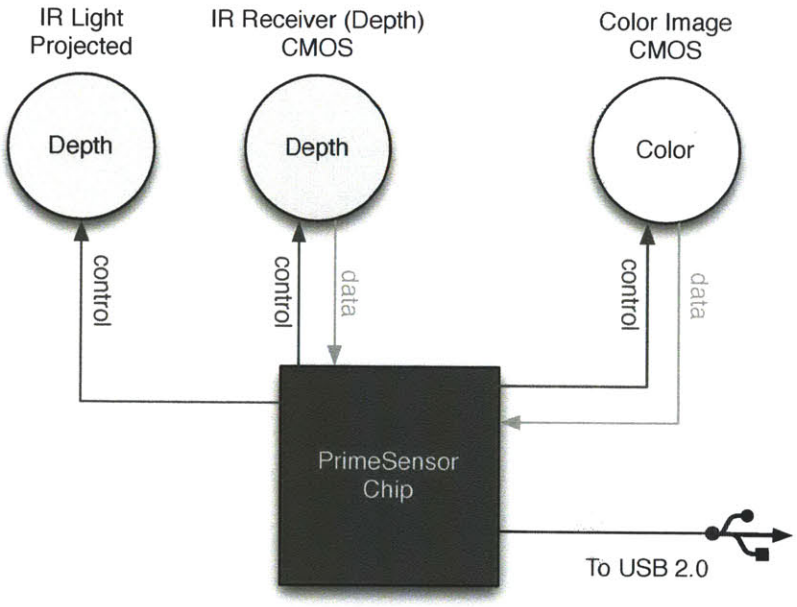


Figure 3. Simplified Block Diagram of the PrimeSensor

This is a view of the important components in the PrimeSensor that enable 3D colored image capture. An infrared projector projects a non-repeating grid of dots that are interpreted by the infrared receiver near it on the device. The advancement of this sensor is that the onboard chip matches the color (visual light) image with the depth field to create a colored depth-map. This diagram neglects the directional audio capabilities of the sensor, as they are not important for this application. (Diagram adapted from a similar diagram on www.primesense.com)

The PrimeSensor packages three optical devices into one lightweight sensor that is capable of producing a UXGA (1600x1200) 3D color video stream at 60 frame/sec. To begin the process of developing this video stream, the first of the three optical devices pictured in Figure 3 projects a non-repeating, slightly scattered array of infrared points, as shown in Figure 4. These points travel away from the sensor and coat the depth of field of the device in a pattern

of bright infrared dots (naked to the human eye), that the sensor can use to determine depth in a room.

Figure 4. Infrared Dots Projected by PrimeSensor

These dots are the infrared points projected at all times by the prime sensor. These are of course invisible to the human eye, but this is the information the infrared receiver uses to gather depth information. Source: dailyvsvids.com

After the depth field is gathered, the onboard chip uses the third optical device, the color camera, to map color and visible light information, pixel-by-pixel, to those depth points in a 3D space.

Previously in research exploring this technology, the most difficult aspect of achieving the level of fidelity the PrimeSensor does is the fact that the visual and infrared cameras must be perfectly in phase with each other. PrimeSensor uses a centralized controller to control the timing of each camera and ensures that the only signal that is sent along the USB cable to the computer using the sensor is a perfectly timed infrared with color information.

4.2. Microsoft Kinect Sensor

In June of 2010, Primesense licensed the technology for the PrimeSensor to Microsoft, and the popular video game controller for Xbox 360 called the Kinect was born. Microsoft took the small PrimeSensor, which was already optimized for human gesture interpretation, and gave the sensor a new body, to match the appearance of the newest Xbox 360 released to that date. Additionally, Microsoft added a motor to the base of the sensor, allowing it to pan up and down autonomously to determine the best viewing angle for those in the room.

Since the development of the Kinect was targeted specifically toward the Xbox 360, Microsoft also changed the standard USB 2.0 connector to a proprietary Xbox connector that also supplied a small amount of power to drive the motorized base. The combination of the excess size and weight of having a motorized base, the use of the non-standard Xbox connector, and the fact that the USB signal is no longer sufficiently powerful to power the sensor and the motor make the selection of the original PrimeSensor a clearly superior choice than the Microsoft Kinect.

Though the Kinect is not the best choice for robotics applications, Microsoft recently announced that they plan to release their software API as an open-source package for independent developers. This decision, along with the popularity of the sensor overall, are both very beneficial developments, even for roboticists using the PrimeSensor. As with any developing commercialized industry, with a larger user population, there is a greater concern for reliability and development of new technologies will occur at a faster rate.

4.3. Economic Considerations of PrimeSensor

As stated, though the Kinect is likely to be adopted by a significantly larger user population than the PrimeSensor is, the hardware is identical and serves the same function. Therefore, any development for the Kinect that is done will be, at least in a logical sense, transferable directly to the PrimeSensor. Additionally, since the Kinect currently has complete market share in the sector of gesture-based gaming, anyone who wants this platform must buy the same hardware as the sensor that has been installed onto the robot documented in this work.

In addition to the greater user base of Kinect and PrimeSensor combined, there will be more of an incentive for these corporations to improve the reliability and accuracy of these combined infrared, color systems. The PrimeSensor is cheaper, smaller, lighter, and uses less power than the Kinect, making it the perfect choice for robotics applications. As the number of packages developed for these technologies increases, the software and hardware collaboration may move in a trend similar to other technologies in the robotics world, that is to say, toward open-source and to the cheaper, lighter version of the technology. The market for both sensors exists, but for our robotic applications, one option is superior.

5. Selection of Onboard Computer System

5.1. Requirements

To perform the SLAM algorithms this robot is designed to undertake, a lightweight system is simply not an option. The entire purpose of this robot is to accelerate the development of SLAM and life-long mapping software, so it makes no sense to install a computer system that will take time away from the researcher's software development by limiting the speed of data analysis. Other systems on the Create have been developed using similar sensor suites, but the goal of this robot is to have the processing ability, combined with battery life, to accomplish heavy computation for long periods of time in an autonomous fashion.

Secondly, as this computer will be operating for long periods of time without connection to a base computer to recharge batteries or offload data, there needs to be a sufficiently large hard-drive to store data collected by the computer and sensor over the course of an hour or longer autonomous study. Again, removing all limiting factors except the roboticist's code is the primary goal.

The final consideration, as with any design is price. Eventually this system could have many like counterparts, so designing from the beginning to minimize cost per unit is essential. To select the computer system for this robot, several designs were considered. These designs are summarized in the following sections.

5.2. Computer Options

5.2.1. Home-Built Option

In first approaching the discussion of the selection of the computer system to anchor the real-time SLAM robot, factors like cost, processing power, weight, durability and battery life are the primary considerations. Many robots in research labs today have custom-built computer systems, and for some applications, this is in fact the best option.

When we considered this option, the primary variable by which we narrowed our search was having an Intel Core i5 or Core i7 processor, to provide the necessary image processing capabilities for real-time SLAM, while still maintaining long battery life. We found several options, but the base price range of these processors alone was between $140 and $280. Next, the size of the system must be minimized so that it will comfortably fit on top of the Create, so a Mini ITX motherboard with the necessary slot for an i5 or i7 was selected. In further searching, this component proved to be an additional $180 to $300.

RAM is not the most costly of components, and the numbers we looked at were in the $60 to $130 price range. The next component was the hard-drive, needed to store an hour or more of very high fidelity data. Though this data may not actually occupy a large enough space to warrant very high numbers for storage, we decided to allow for the researcher to store multiple files on the computer without needing to offload after every test run, so a 500 GB hard drive was settled on as the target size. When searching for this component, we learned that such a drive would cost between $40 and $90.

So far in our search, we have built the ability to process and store data, and the estimates of price ranged from $420 to $800. Still to be included are any additional graphics processing units, a power supply system, cooling system, any keyboard and mouse additions, and of course a display. Without intimate knowledge of home-built laptop batteries, the notion of building our own onboard computer was becoming decreasingly competitive.

5.2.2. Factory-Built Option

For comparison at this point, we began searching for build to order laptops from the various major laptop computer manufacturers. To provide for equal comparison among brands, the following parameters were decided: Intel Core i5 processor, 11.1" screen, 4 GB memory, 500 GB hard-drive space, and other hardware devices compatible with the Ubuntu operating system (reasons for which are discussed in a later section).

Very quickly we began to find results from various manufacturers that were well within the price range we had arrived at based on the home-built option. One advantage of such systems was that all input and output devices (keyboard, mouse, display) are already integrated,

reducing complexity of the computer system and later the chassis needed to support the system. Another advantage of the factory-built computers was battery life. With the home-built computer system, determining expected battery life and determining the optimal system to purchase to power the computer would be a complex and error-prone approximation at best. The manufacturers of these factory-built systems could simply supply us with battery life estimates, eliminating this entire branch of complexity from our design.

The final and most compelling reason to purchase a factory-built computer in spite of what may be a higher price tag was reliability. Having a corporation whose sole purpose is to produce quality computer systems put their logo on their product, guaranteeing for some warranty period at least, that this system would function as advertised, is a very compelling argument to purchase a factory-built computer.

5.3. Final Selection

When the in-depth study of each option of system design had been carefully considered, the factors of reliability and robustness pushed us to select a factory-built computer. The final selection was at the low end of our estimate on total system cost to meet the necessary parameters for this mission.

The computer we selected was the Acer Aspire 1830T-6651, featuring a 1.33 GHz Core i5 processor, 4 GB of onboard memory, a 500 GB hard-drive, an 11.6" display, and an advertised battery life of nearly 9 hours. The total cost of this system was under $700, falling inside the range of the components researched for the home-built system, not including power supply and input/output devices. The system is 11.2" wide, 8.0", merely 1.1" tall, and weighs 3.1 lbs. These dimensions absolutely could not have been achieved in a home-built system.

5.4. Additional Modifications

Though this computer system shipped with a full copy of Microsoft Windows 7 Home, this operating system is not the desired OS for our research. The standard in the MIT robotics community is a Linux operating system, usually the latest release of the open-source Ubuntu operating system.

5.4.1. Reasons for Ubuntu

Free and open-source operating systems decouple the system on which you rely from decisions made by a corporation whose goal is not to have your system running 100% of the time, but to maximize their profit. While the argument for open-source software regarding large companies is valid, there are other reasons why a UNIX-based OS is greatly preferred.

Many pieces of software developed are targeted toward a more universal application, and aim to decouple the operating system specifics of a computer from the core functionality of the algorithm of the code being tested. UNIX-based operating systems, such as Ubuntu or Mac OSX, provide for programs to be run from a standardized and well-known command-line environment, common to all UNIX systems. This means that software developers and

researchers no longer need to develop a user interface for their software and can focus more of their attention on the robustness of their code and the core of the algorithms they choose. Additionally, when producing a program designed to run from a UNIX terminal, if many such programs must be started or initialized in a prescriptive manner, Bash provides an easy-to-use, lightweight scripting language for automating any such procedure. Once a small bit of code is tested and proven reliable, it can be encapsulated in a shell-script for later use from an abstracted point of view. Robotics software developed on this system and on many others in the research world made the use of the UNIX command line the research area standard.

The final and most critical reason for choosing Ubuntu is that it is the only operating system with full support of ROS. While other open source packages have been developed, only partial functionality is advertised for Windows and other mainstream operating systems.

6. Design and Fabrication of Support Chassis

Now that the mobile base, computer, and sensor are selected for this robot, the design of the support chassis for the computer and sensor can begin. General goals are for this chassis to be inexpensive to produce in a robotics lab, and for it to have a low profile while still elevating the sensor to a height from which it can see both the ground and above. If the sensor is mounted too low, the ground will only occupy a narrow band of the sensor's field of view, making identification of features on the ground difficult.

Available to users of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL, the location at which this study was conducted), is a laser-cutter capable of precision machining of many types of plastics and wood. The supply of these materials is readily available in the laboratory space, so this seemed the natural choice for a building material for our chassis. Additionally, any future iterations or copies of this robot will be constructed in the same lab space, so publishing the design of such a robot would benefit users of the lab in the future.

6.1. Critical Dimensions of the Create

6.1.1. Height Considerations

Since the design of this chassis is to be as low as possible, keeping the center of mass low for increased stability, the maximum vertical dimension of the Create and the support components must be determined. Overall, the Create maintains a low profile and an even height across the top surface of the device, but using the USB to serial interface used in this study, the serial port on the right rear quadrant of the top of the Create causes the wire to rise approximately three quarters of an inch above the top plane of the Create. As the laptop will cover this area, this is the minimum height of the bottom of the chassis.

6.1.2. Machine Screw Locations

To attach the chassis, iRobot has provided four machine screw locations on the top of the Create. The locations of these screws, along with the location of the impinging serial port, are summarized in Figure 5 (below). They are centered horizontally and are offset by 0.300 inches

from the geometric center of the Create. As a limiting value, we assumed diameter of the Create, giving room for the bump sensor, was 12.25 inches. The screws can screw directly into the Create, however to accommodate the extra 0.750 inches necessary for the serial cable, the screws will screw into struts, of the same thread count, to be discussed in a following section.

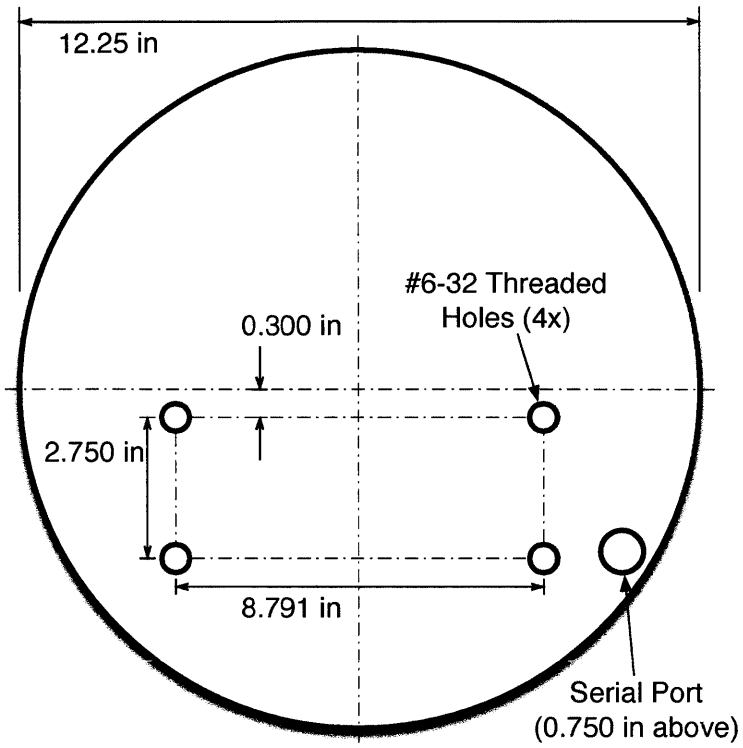


Figure 5. Machine Screw Locations and Serial Port Location on the Create

The circles above are representative of the location of the four #6-32 machine screw receptacles on the top of the Create. These locations are dimensioned from the geometric center of the Create. The circle in the lower right quadrant represents the approximate location of the serial port, the tallest feature of the Create when plugged in.

6.1.3. Button Access

As shown in Figure 1, there are three buttons on the top of the Create: one used to power the Create on and off, another for running a program programmed directly into the device, and a third to skip to the next program stored in the device's memory. As we will be using a wrapper software package for the Create and running code directly from our onboard computer, only the button used to power the Create on and off is necessary for accessibility. In the design of our chassis, we allowed for access to all three buttons, bearing in mind the possibility of other uses for the device in the future.

6.2. Dimensions of the Laptop Computer

6.2.1. Height, Width, and Depth

The laptop we selected as our onboard system, the Acer Aspire 1830T-6651, has outer dimensions of 11.6 inches by 8.0 inches by 1.1 inches at the maximum. As it turns out, the height of the laptop varies from front to rear, increasing from around 0.8 inches to the full 1.1. In our design for the holder for the laptop, this slope was considered.

6.2.2. Screen Visibility

When placed on the Create, the Aspire laptop overhangs the rear of the robot by approximately 2 inches in each corner, and is flush with the round edge of the Create on the front corners. As the PrimeSensor needed to be elevated off the ground slightly, the support structure for this must occupy the same space above the bottom of the platform as the opening and visible area of the laptop screen would.

Tests were conducted at various viewing angles with the laptop on top of the Create, to simulate the in-lab testing of software likely running on the Create. In order to optimize for the best angular resolution of the screen from the crouched position above the robot, these tests consisted of tilting the screen as little as possible while retaining comfort in the crouching position and good visibility of the screen. An angle of just under 30° beyond full vertical was established as the minimum comfortable tilt of the screen. As described in the following section, this number was used to position the PrimeSensor supports behind the screen, and out of the view of the user of the computer.

6.2.3. Accessibility of Ports

As the Aspire is a somewhat small laptop considering its power, both sides of it feature ports for the majority of the length of the side of the laptop. The robot will be moving around an environment as it is conducting tests, so the secure fastening of the laptop to the robot is essential. In our implementation, this meant securing all four corners of the laptop at the full height allowable with the screen open and hands on the keyboard. However, the height of this wall must be reduced along the sides of the laptop to allow for access to the ports.

The idea of creating individual holes in the wall for each port was considered for a short while, but the complete reduction in wall height was selected for two primary reasons. The first reason is a practical material consideration, as the ports are so close together that the thickness of material between ports would be fragile and brittle under heavy usage. The second reason for opening the walls completely is so that if the researcher wants to lift the laptop up for any reason, say to read fine text or access the buttons underneath our chassis, the cords do not need to be unplugged. Since they are free to rise vertically upward and away from the chassis, the laptop's motion is unencumbered.

6.3. PrimeSensor Viewing Capabilities

6.3.1. Overall Dimensions of Sensor

The PrimeSensor is relatively small, measuring in at merely 5.5 inches across, 2.0 inches tall, and 1.4 inches high. Designing support platform to house the PrimeSensor and accommodate its three forward facing optical elements is not a difficult task.

6.3.2. Viewing Angle Limitations

The sensor is capable of achieving a 58° horizontal viewing sweep and 45° vertical, meaning that if placed on the ground with no increase in angle, the camera would only view 22.5° of vertical space ahead. In our implementation, the goal is to raise the camera approximately 14 inches above the ground and allow for ± 45° angle modification on the mount.

Upon first consideration of the adjustable mount, a researcher raised the issue of repeatability and consistency if the camera was able simply to pivot up and down. His concern was from the point of view of the computer scientist attempting to use the incoming data, if a camera was tilted at an arbitrary angle, the scientist would have no idea the actual angle at which the camera was tilted. A design iteration of the mechanism by which the camera changes angle resulted in a freely rotating camera base, with the addition of a pin at every 5° of vertical view rotation, both as a reference for the scientist and to enable consistency from one usage to another, even if the angle of the camera has been altered between uses. Figure 6 demonstrates our solution to this problem.

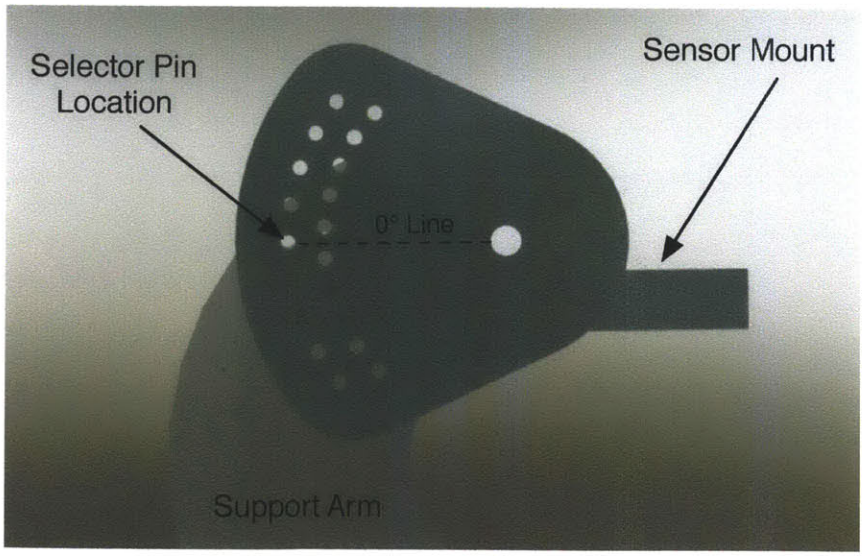


Figure 6. Angle Selector Pin on PrimeSensor Mount

This is a rendered photo of the angle selector for the PrimeSensor mount. It shows the platform on which the PrimeSensor will sit (horizontal component to the right) and the plate with holes at 5° increments from -45° to +45°. The curved arm approaching from above is a clear piece of acrylic, designed such that the user can view the hole into which he or she places the pin. Depicted is the angle selector, just below 0°.

6.4. Complete Assembly

Factoring all of these considerations, and optimizing for ease of assembly and use, the final design for the robot chassis is shown in Figures 7 and 8 below.

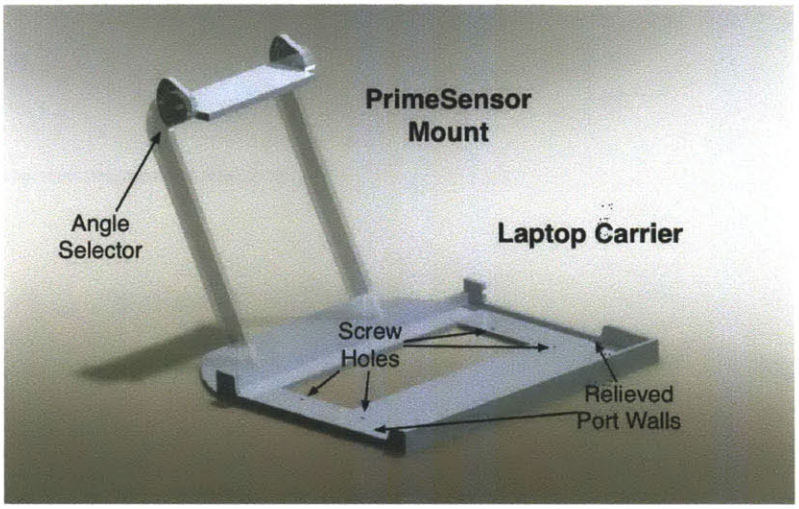


Figure 7. Complete Design of Laptop and Sensor Chassis

The Laptop Carrier section of the chassis is exactly the height (sloped upward from bottom to top) of the keyboard and trackpad surface of the open laptop. The walls to the right and left are relieved to allow access to the ports in the computer without locking the laptop in place if it needs to be lifted. The central hole allows for access to the buttons on the Create itself, but once the robot is on and the computer is connected, these buttons serve no function. Above is the PrimeSensor mount, featuring the angle selector.

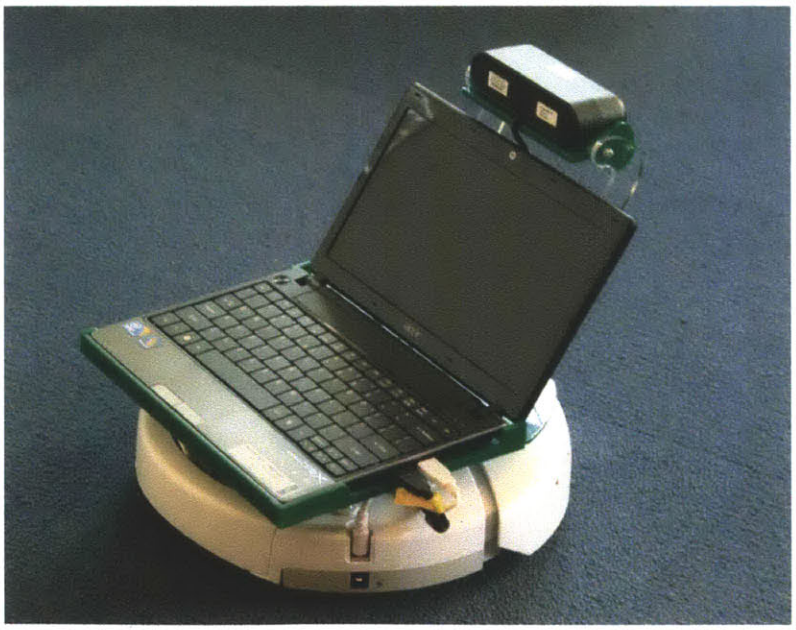


Figure 8. Photograph of Completed Robot

Visible from this angle are the cables connecting the PrimeSensor and the Create to the laptop. Just below the right side of the laptop is the vertically-mounted USB cable in the Create which mandates the height of the entire support chassis.

The 30° angle calculated as the optimal viewing angle for the laptop screen was used to angle the support beams for the PrimeSensor mount. In an effort to reduce the additional space in front of the laptop occupied by the sensor and its mount, the mount is set behind the support beams. The height of the mount was calculated to allow the laptop lid the full extension until it intersects the support arms, and the distance the sensor is set back was calculated so that, in the optimal viewing position, neither the sensor nor the mount obscure the laptop.

6.4.1. Vertical Struts

In order to raise the chassis the required 0.750 inches above the serial port, small aluminum struts were manufactured with a male #6-32 threaded rod on one end and a female #6-32 threaded hole on the other. This way, the original screws supplied by iRobot can still be used, and the struts can be attached and tightened separately. Figure 9 shows a basic schematic of the support struts. A safety margin of 0.250 inches was added to the struts, to avoid unnecessary crimping of the serial wire as it collides with the support chassis.

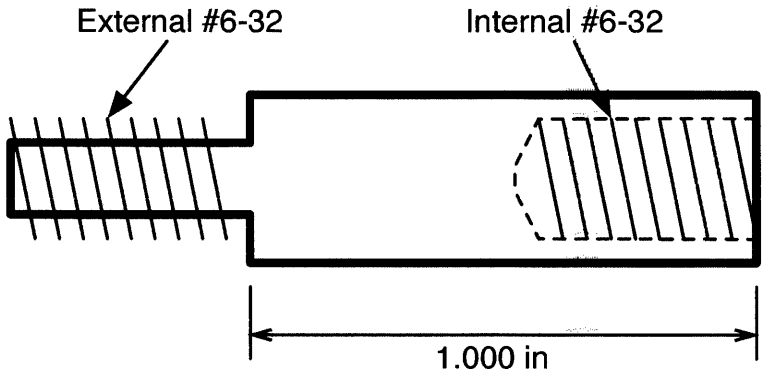


Figure 9. Diagram of Support Struts

The four support struts of the above design separate the chassis from the Create by 1.000 inches to make room for the serial cable used to communicate with the computer. These struts are available for purchase, but in this implementation, they were made by hand on a lathe. For future iterations, these struts should be replaced by outsourced parts.

7. ROS Packages to Link Devices

7.1. ROS on the Acer Aspire

In order to run a ROS robot, the first step that must be taken is to install and set up the ROS core drivers, the ROS Master, on the host machine. As ROS is an open-source package being continuously developed and updated, the most recent releases of the software are usually the best. As of the writing of this paper, the third major ROS release called *Diamondback* is the most current, and it can be downloaded from `www.ros.org`. From that page, the link *Install* has a list of operating systems from which to choose. Currently, the only OS that is stated to be

fully supported is Ubuntu (another factor in considering the operating system to install on the laptop).

The use of the UNIX terminal shell is critical in the installation of this software and much of the software used on this system. The install page on `www.ros.org` provides a simple, step-by-step process for installing and configuring ROS to run on a laptop computer.

7.2. Using a ROS wrapper around the Create

The next challenge we faced in the selection of packages for installation was finding a way to programmatically represent the Create and pass it commands. Many ROS packages exist and are readily available for download and installation. The package we selected, especially because other users of Creates are already developing software using it, is a package distributed by Brown University called `brown-ros-pkg`. Again, as of the publishing of this paper, this software is available at `code.google.com/p/brown-ros-pkg`.

This package has its own set of install criteria, which are easy to follow given the on-screen instructions. One small challenge is that some of the commands in the install section may not have been updated to reflect the most recent ROS release, so in place of `boxturtle`, you may need to type `diamondback`, or the title of a later release.

After this installation, the final package to allow communication with the Create is to install the specific Brown University package called `irobot_create_2_1`, available from the main page of `brown-ros-pkg`. This package provides the foundation of a C++ or Python class to act as a code object representing the Create. This object has methods for driving, turning, polling the states of the sensors onboard the Create, along with others. For any development of software on this system, this is the code point at which one would begin.

7.3. ROS Package for Kinect or PrimeSensor Support

The final piece of software to integrate is the PrimeSensor. In theory, the PrimeSensor will need to be placed in the existing ROS core when opened as another node, and there are packages with this capability. The package used by this robot is one developed by the not-for-profit company called OpenNI, whose mission is "to certify and promote the compatibility and interoperability of Natural Interaction (NI) devices." The description of this package and installation instructions are available at `www.ros.org/wiki/openni_kinect`. This package was designed and released to build on ROS Diamondback, and is not backwards compatible with any previous versions.

8. Improvements for a Next Generation

8.1. Hardware

8.1.1. Robot and Chassis Interface

Currently the interface between the robot and the chassis is established through custom built struts, each requiring about 10 minutes on a hand-operated lathe, with a cutting tool, die, grooving tool, center drill, drill bits, tap, and a knurling tool. While these elements are perfectly designed and serve their purpose well, they were only created using this method out of necessity; the build of the robot could not have been held up waiting to buy these parts from a retailer. For purposes of mass construction of a fleet of these robots in the future, a 1-inch #6-32 hex standoff can be bought for a very low price.

8.1.2. Laptop Support

Since this laptop is so small, one usability issue with the current model of the laptop carrier is that the laptop does not weigh enough to stay in place when the lid is opened. Two hands must be used when opening the lid from the closed position. This minor problem could be mitigated with a small tab (made of a soft material, so as not to scratch the frame around the screen) placed at the corner of the laptop that is closer to the operator. This tab could be screwed into place on the support wall and be thin enough to not block the screen from closing by a significant amount.

Additionally, I found that over time with the laser cut corner of the laptop support being at the center of the area where the user of the laptop rests his or her hands while working, the sharpness left there by the corner of the acrylic becomes painful and unpleasant. This can be greatly improved by filing those two corners to a rounded edge using any standard metal file.

8.1.3. PrimeSensor Support Beams

In the current design, the entire sensor support structure is held in place by two screws through the bottom of the chassis and into the support beams. The actual manufacturing of these holes and the drilling head-on into the end of the beam proved to be a needlessly complex task demanding far too much time. In future generations, this design should be altered to be a simple mortise and tenon joint (shown below).

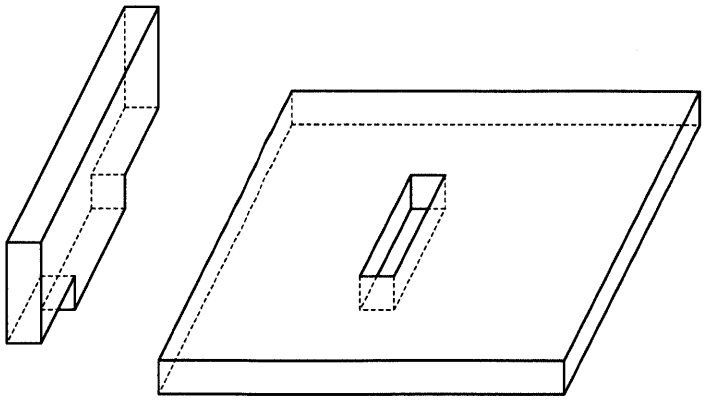


Figure 10. Mortise and Tenon Joint

This joint type is used very commonly in woodworking, laser-cut acrylic parts, and many other fields. It is very simple and strong when affixed with acrylic epoxy. This joint is used elsewhere in the support chassis, as it is the cheapest and fastest way to prototype with laser-cut acrylic.

8.1.4. PrimeSensor Angle Adjuster

An improvement that can be made to the angle adjuster is to reduce the number of downward tilting angle options. A thought that was not considered in this iteration of the design was that it is unlikely the researcher using this platform will want to perform SLAM on the ground at a 45° angle below the horizon. Perhaps limiting the view to 45° up and 20° down (leaving only 2.5° of view above the horizon) is a more reasonable limit.

Also, in order to achieve 5° increments, the size of the holes in the angle adjuster had to be a tiny $1/16^{th}$ of an inch (0.063"), making use of this feature require some dexterity. Perhaps doubling the size of all features on the adjuster might make this feature stronger and easier to use. Reducing the number of options from 5° increments to 10° ones will offer slightly less variability, but with the larger, more user-friendly interface.

Currently, the angle adjuster has angle markings, calling out the nominal difference in angle from the arm to the sensor. These markings, however, are on the inner component of the adjuster, necessitating a clear support arm. In a future iteration, the markings and the wide array of angle selection holes could be placed on the support arm itself and have the single selection hole defining the angle on the inner part. This allows the entire build to be made out of one, non-clear material if necessary and instead of needing to align two visible lines in a plane to read an angle marking off of the inner part, the $1/8^{th}$ inch pin will act as the indicator and the user can read the angle marking corresponding to the location of the pin.

8.2. Software

Only a few small improvements can be offered at this time, as the software associated with this specific robot is still in its infant stages. First, allowing the computer to be accessed and controlled wirelessly with SSH will make testing and implementation of code much easier. Currently, if the robot is driving (carrying out a mission) and the process needs to be stopped, the researcher must approach the robot and press, in coordination, `Ctnl + C` to stop the running

process. While in theory this seems like an easy task, if your script mandates that the robot turn, start, and stop without warning, coordinating two fingers simultaneously on a keyboard that is moving as well can be a daunting task.

SSH access would also allow the researcher to operate the robot with the laptop lid closed. As the robot drives around, testing software, it is likely to strike other objects or possibly lose balance and topple. The laptop is the most expensive component on this robot, and having its screen and input tools protected during actual testing may save the laptop from potentially crippling damage during operation.

9. Summary and Conclusions

9.1. Motivation

Over the course of this project, we have discussed Simultaneous Localization a Mapping, a process by which a robot can simultaneously determine its location in an unknown area and create a map around itself to use to navigate and retain information about that area. Previous versions of robots have had to use expensive Light Detection and Ranging (LIDAR) sensors, which only give a small field of view and give no color information.

New technological developments have allowed far more information to be gathered for a vastly reduced price. Sensors such as the PrimeSensor, made by Primesense of Israel, are capable of producing a depth-field similar to that of a LIDAR, but in two dimensions rather than just one. Additionally, this sensor has a hardware matched light camera, which supplies color information to the 3D point recorded by the depth camera. All in all, this sensor can very inexpensively provide a colored, 3D representation of a point of view and relay this information to a computer or robot.

The possibilities of this application are endless, but for now the field is working on faster and autonomous 3D reconstruction of large areas by using a moving platform to scan and map an interior space. These robots needn't be large or complex, because they are providing the foundation for technology that will give robots computerized eyes that give comparable information to that of the human eye.

9.2. Construction of a Robot

To provide this foundation, a simple and inexpensive robot needs to be constructed. This robot needs to have the ability to move around a room, it must support a small sensor such as the PrimeSensor, and it needs to hold a computational system to link the two and build and store the map. iRobot has provided a simple and adaptable robotics platform based on the niche product, the Roomba, called the Create, which can provide this mobile platform and has the flexibility to allow the attachment of the computer and sensor.

Mechanical integration of these three systems is accomplished relatively easily with a custom designed chassis designed specifically for a laptop. The laptop in this case is designed with the intentions of being lightweight and having great computational power and battery life.

The chassis will support both the laptop and the sensor in such a way that the operation of any one system does not interfere with the researcher's ability to operate any of the other systems.

Software integration was achieved using the Robot Operating System (ROS) as its backbone. Other research groups or corporations have developed various software packages to provide shells for the physical components of the system, and installation and integration of these packages is straightforward. Once the packages have been installed and properly linked, the ROS core provides the entry point for all of the researcher's SLAM and life-long mapping software.

9.3. Conclusion of Research

This robotics system is a robust and affordable platform for the development of mobile robotics mapping software and other vision-based artificial intelligence technologies. It will be able to easily navigate and map complex indoor and some outdoor spaces, with the possibility of logging and processing continuous runs of data in excess of two or three hours. The developments of life-long mapping will enable the robot to use previous and existing data sets for optimization and increased error correction while mapping a partially known space.

The strength of the computer system onboard will allow serious visual processing, object-recognition, path planning, and obstacle avoidance as the data are being recorded. Because of onboard network capabilities and the UNIX operating system, access to the laptop and its core processes can be very easily managed remotely through SSH and a UNIX shell. This will free the roboticist from the need to chase the robot around the room or develop and deploy code on the 11.1-inch screen, again, increasing his or her effectiveness and the rate at which code can be developed.

The entire goal of this robot is to provide an inexpensive, robust, and versatile platform for the development of artificial intelligence software. By using this system, a researcher can reduce all other occupations of his or her time, allowing the highest amount of his or her thoughts to rest in the improvement of the novel software. As inexpensive and capable as these systems are, a fleet of them could be produced for educational purposes, or to research cooperation among various systems. These robots can serve as any platform for the research of 3D spatial artificial intelligence software.

The software developed on these robots has applications far beyond the inner walls of a lab or test area. Currently, technology is moving forward in the field of autonomous cars and aircraft. These vehicles will leverage software developed on a robot as simple as this one. This software will help shape the way many future robots interpret their surroundings, making it easier and easier over time to abstract the software stacks running robots. By giving robots genuine, native vision, and the tools to interpret this information in a meaningful way, we move the field of artificial intelligence closer to a point where robots are capable of autonomously interacting with the world with the capability to rival that of human beings.

10. Works Cited

Ben-Horin, B. *Introducing OpenNI.* < http://www.openni.org/>

Bouffard, P. *ROS/Concepts – ROS Wiki.* 03-2011. <http://www.ros.org/wiki/ROS/Concepts>

Brown University. *Brown-ros-pkg s Brown University Repository for ROS Packages.* 2011 Google. < http://code.google.com/p/brown-ros-pkg/>

Conley, K. *openni_kinect – ROS Wiki.* 04-2011. < http://www.ros.org/wiki/openni_kinect>

iRobot Corporation. *iRobot Corporation: About iRobot.* 2011 <http://www.irobot.com/sp.cfm?pageid=74>

iRobot Corporation. *iRobot: Robots: Programmable: iRobot Create Programmable Robot.* 2011 <http://store.irobot.com/product/index.jsp?productId=2586252&cp=2804605.2591511&ab=CMS_IRBT_CreateSuperCat_CreateFam_111309&s=D-StorePrice-IRBT&parentPage=family>

Kinect Infrared Projection Dots vs. Night Vision Camera & IR Goggles. 2010. < http://3.bp.blogspot.com/_tWqvsW7WRl0/TNgi4rQbtRI/AAAAAAAAb1k/AbP8TUrwtnU/s640/Kinect+Infrared+Projection+Dots+vs.+Night+Vision+Camera+&+IR+Goggles+1.jpg>

Primesense, Ltd. *Primesense, Ltd. Chip.* 2010. <http://www.primesense.com/?p=488>

Primesense, Ltd. *Primesense, Ltd. Company Profile.* 2010. <http://www.primesense.com/?p=689>

Quigly, M. et al. *ROS: an open-source Robot Operating System.* ICRA 2009. <http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf>

Riisgaard, S. and Blas, M.R. *Slam for Dummies.* 2005. <http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf>

Schramm, Mike. *Kinect: The company behind the tech explains how it works.* 06-2010. < http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/>

Willow Garage. *ROS | Willow Garage.* 2010. <http://www.willowgarage.com/pages/software/ros-platform>