LESS: Low-Cost 3D Environment Sensing

Timothy Tufts, CSE, Alexander Montes McNeil, EE, Gabriela Correa, EE,
and Alexander Maerko, EE

Abstract—**Three dimensional outdoor mapping systems for robot navigation and obstacle avoidance are in rare use among hacker communities, due to prohibitive costs. We explore a method of making this technology more affordable by expanding a Microsoft Kinect's functionality to outdoor environments. A novel optical system is built and tested in order to accomplish this. Implementation with an ATRV-Jr mobile robot is shown, and results for final implementation are presented.**

I. Introduction

SOME of the greatest tech companies that we have today started as simple garage projects [1]; among the most famous are Hewlett-Packard, Apple, and Google. The struggle that these inventors face varies from maintaining a working knowledge of modern development methods, to the expenses of state of the art technology. The Low-Cost 3D Environment Sensing System (LESS) project aims to ease this anxiety for one particular aspect of robotics. We seek to bring outdoor 3D environment sensing to the weekend technology warrior, opening the garage into the backyard.

Already autonomous vehicles, such as the Kiva robot at Amazon, are being used for more efficient commercial systems. The Kiva enhances the packaging process of Amazon by delivering products inside an expansive warehouse to a human for packaging [2]. Giving the hobbyist the capability to create a robot like Kiva that can leave the warehouse could create a wildfire of innovation. We would like to give our technology to inventors so they can apply it to other technologies we have yet to conceive.

Affordable robots already apply technology similar to what LESS hopes to achieve in homes around the world—the Roomba can vacuum your floor while you're at work and the Winbot can clean your windows. This begs the question, where is my automatic lawnmower? Besides the obvious safety issues, the technologies employed in these indoor helper bots only work in environments that have low levels of sunlight or are not accurate enough for something like an outdoor helper bot [3]. When this outdoor technology is put in the inventor's hand, a new wave of robots can emerge that will perform menial tasks humans never wanted to do themselves. Ultimately this has the ability to free up the everyday human's agenda to do exclusively tasks that robots cannot perform, such as innovative thinking.

In tables I & II we give the initial specifications for the two main systems of our project. Table I describes the requirements for the Optics system. These parameters were designed to maintain the integrity of the base Kinect performance, despite the wide variety of environments we expect it to be employed in. For this reason every one of the Kinect's original field of view and range requirements were maintained, if not improved upon. While the final system will improve upon the range requirements, it will also add distortion to the 3D generation. This leads to the specifications considering the distortion between an unmodified Kinect and the LESS. According to Konolige and Mehelich, the distortions from optical filters on the Kinect's IR camera are between 0.1 to 0.2 pixels [4]. Taking this into account we gave a tolerance on each pixel in the depth image of 0.1 m.

TABLE I
Kinect Specifications

Specification	Value
Min Render Distance	< 0.4 m
Max Render Distance	> 4 m
Horizontal Field of View	> 57 °
Vertical Field of View	> 43 °
Tolerance from Unmodified Kinect	< 0.1 m
IR laser projector temperature	< 102 °C

TABLE II
Rover Specifications

Specification	Value
Kinect Horizontal Range	> 0.65 m
Kinect Vertical Range	> 1.07 m
Rover Footprint Width	0.65 m
Rover Footprint Length	1 m
Rover Clearance Height	1.25 m
GPS Arrival Accuracy	3 m
GPS Travel Speed	0.5 m/s
Kinect Forward Range	0.5 < Range < 2.5 m
Obstacle Avoidance Travel Speed	0.5 m/s
Kinect Operational Frequency	0.25 Hz

The Kinect's IR projector also has a very important specification for the temperature. The overall temperature of the IR projector cannot exceed 102°C due to a built in safety temperature sensor. This sensor is programmed to shut off power to the projector in cases of extreme heat which could melt the surrounding parts.

Table II describes the requirements for the Rover system. Our requirements for the Rover system are designed with both the physical dimensions of the Rover and the Kinect hobbyist community in mind. The horizontal and vertical ranges are defined such that the Kinect should be able to visualize the entire height and width of the rover going forward. Additionally the footprint and clearance height define the shape of the rover inside of the software, so if we tweak the parameters in code, we do need to meet the given values.

Additional rover specific requirements include the GPS speed and accuracy, set by the previous work done by SDP14 team AIR. We can potentially increase these two performance parameters, but our goal is to not decrease them.

The remaining parameters are intended to meet the default settings for the Robot Operating System (ROS) navigation stack (described later). Since this package is used by many hobbyists, we want our hardware to be compatible with the standard setup. The forward range value describes the range in front of the rover where an object is considered an obstacle, so our Kinect must be able to see in that range. The object avoidance travel speed describes the maximum speed that the rover can travel at while still avoiding obstacles. The selected 0.5 m/s is both the standard speed of a navigation stack robot and the speed of our GPS travel. Lastly, the Kinect operational frequency is derived from the previous two requirements. If the rover travels at 0.5 m/s and the forward range has a 2 meter length, the Kinect must process an image at least once every 4 seconds or else an object can move through the operational range undetected. In reality, the Kinect operates at 30 Hz, so this requirement is very much satisfied.

II. Design

A. Overview

We took a modular approach to our design with the overall goal to simplify each module for an extremely adaptable solution. As a result our solution has two main components: the Kinect system and the Rover system. The Kinect system is the modified hardware we would market to hobbyists, while the Rover system represents any robot running the OpenNI software for Kinect. In our case, the Rover system is the ATRV-Jr rover which runs ROS with an OpenNI package [5].

The main innovation in the LESS hardware consists of modifying the 3D environment building from the Microsoft Kinect for outdoor use. This technology was selected because it has yet to be widely applied for 3D environment rendering. More expensive systems have used similar methods to achieve the same result, such as Light Detection and Ranging (LIDAR) systems. Like most LIDAR systems, the Kinect projects and observes the changes of an infrared signal. The Kinect's current system fails because it does not employ proper filtering systems for outdoor use, in order to maintain effective consumer pricing for the Microsoft Xbox.

A couple alternatives were considered in the first iteration of the LESS hardware. Initially, SDP14 team AIR attempted some basic object detection on the Mars rover using stereoscopic cameras. We decided that for our project this was not a sufficient method due to the simplified geometry assumptions necessary for 3D object detection using this method [6].The second alternative was using a conventional 3D LIDAR system. These systems have been used for extremely effective real time environment sensing [7], however they are prohibitively expensive and thus not an option for most hobbyists. The cheapest LIDAR sensor from Velodyne is approximately $8,000 [8]. This would undermine the goal of our project which is to bring a solution to hobbyist at a reasonable cost.

Another main competitor to the system was the use of ultrasonic sensors [9]. These sensors have been demonstrated to be effective for object detection in many commercial systems but also have some major drawbacks for our design. While they have great range, they suffer from being less accurate than light based systems. For this reason they are normally part of more complex sensor systems and equally do not meet our requirements of creating a simple, adaptable solution [10].

Although similar to LIDAR, what separates our solution from the previous methods is the way it modifies light for 3D environment sensing. Instead of observing phase shifts such as LIDAR or sonar technologies, the Kinect uses an IR projector which passes its light through a diffraction grating. Once the pattern is projected onto the environment, the infrared camera on the Kinect observes and compares it to a reference pattern. Based on the difference in the patterns, the system can calculate how far away the point is.

Combining a modified Kinect with object detection and avoidance, the block diagram in figure 1 shows our final design solution. The two main blocks, Optics and Rover, display the strategy used to develop the two largest components of the

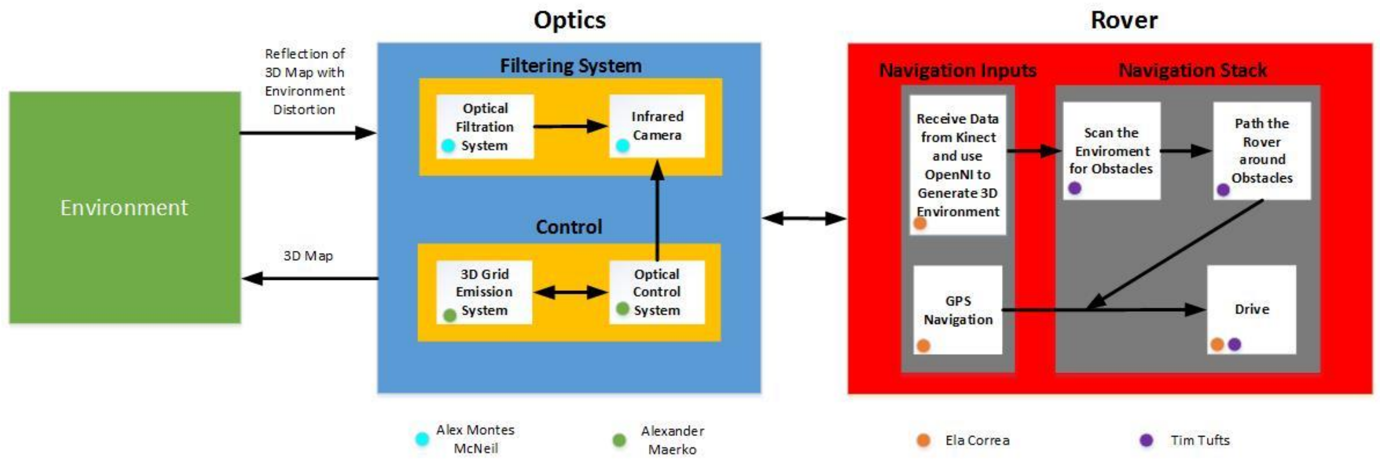

Fig. 1. Block Diagram showing the separation and internal workings of the two main systems.

design. The Optics block strives to create a version of the Kinect which can work outdoors in direct sunlight. While the Rover block creates a computer and software system that can be easily replicated in a wide variety of robotics projects, the Optics block achieves outdoor functionality by further breaking down into two subsystems. The Filtering block uses techniques to remove as much of the sun's interference as possible. The Control block then makes up this difference by increasing the power of the laser so it can be distinguished from the filtered sunlight.

The Rover block uses a similar sub block system in order to ensure the complete functionality of the system. The Navigation Inputs block builds the 3D environment with the OpenNI software provided by Primesense (the same company behind the Kinect). This information is then passed to the navigation stack where it is analyzed in ROS and object detection and avoidance is calculated. The GPS navigation is specific to the Mars rover prototype. This demonstrates a full system application of LESS where a robot is given a particular goal, and uses LESS to ensure it can achieve it without any complications.

B. *Optics: Filtering System*

While there is an abundance of information about the Kinect, the specific design parameters which cause it to fail in direct sunlight are not released. This created motivation to generate a series of experiments which characterized the Kinect. While it would have been ideal for a complete understanding of the Kinect's 3D generation system, complete specifications about the IR projector, diffraction grating, and how OpenNI interprets the information from the IR Camera to create a 3D environment are unknown. This led the design of the experiments to understand the conditions in which the 3D generation system fails. Once this was understood, the second step was to design a filtering system that will prevent the Kinect 3D generation system from failing.

The first set of experiments were designed to examine the different conditions the Kinect IR Camera faces indoors and outdoors. The unaltered IR camera output is displayed in figure 2. The conclusion was that in the worst case conditions outdoors the IR camera was recording the maximum value for the IR

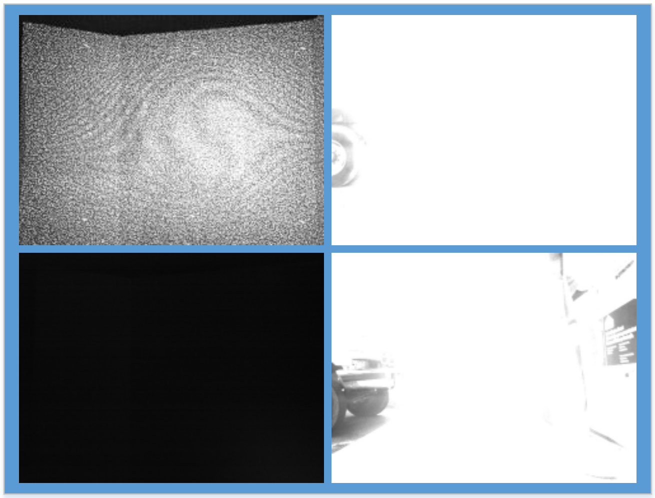

Fig. 2. Top Left: Indoors with Kinect IR; Top Right: Outdoors with direct sun; Bottom Left: Indoors without IR; Bottom Right: Outdoors Cloudy

image meaning the pixels were fully saturated. The background difference between outdoors and indoors is almost the entire scale of the IR camera.

Once it was clear that the Kinect was failing due to its pixels being fully saturated, a variable IR projector was created in order to mimic these conditions. This projector was used to determine the total amount of emitted power necessary to break the 3D generation performed by the Kinect and OpenNI software. This can be observed by the black circle in figure 3.

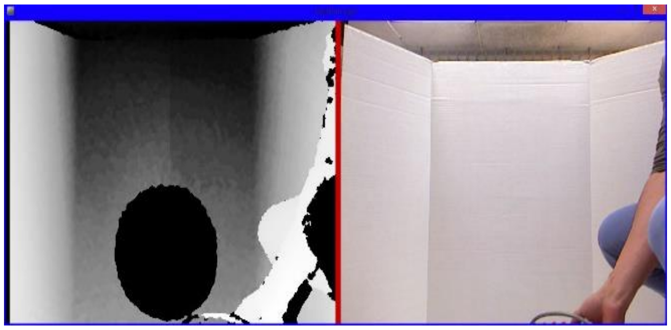

Fig. 3. IR Interference with Kinect 3D Depth map

The amount of total emitted power necessary to interfere with the Kinect 3D generation was 74mW. This is validated by the total amount of power emitted by the Kinects IR projector is approximately 100mW. From this we concluded that Kinect's 3D generation technique fails when the total amount of power on the environment—from the camera's perspective—approaches the total amount of power the Kinect is emitting on the environment.

From this result the design process for the optical filtering system aimed to remove as much power from the sun as possible, and then increase the instantaneous power of the laser to make up the difference. Based off of the ASTM standards for Irradiance with an air mass of 1.5, the power reduction of several band pass filters around the wavelength of the Kinect's laser is shown in table III.

TABLE III
REDUCTION OF SUN BY BAND PASS FILTER

Band Pass Filter (nm)	Power from the Sun (W)
830 +/- 100	544.000
830 +/- 10	5.800
830 +/- 2	0.316

The total power is the amount of power from sunlight that floods the Kinects camera when emitted on the same environment it is trying to observe. The stock Kinect has an 830 nm +/- 100 nm filter on it meaning it could experience up to 544 W from the sun. This is orders of magnitude above the Kinects 100 mW laser which is why its 3D generation technique does not work outside.

This left two design options for the Kinect with the other two band pass filters. Due to complications that arose when trying to use other lasers on the Kinect (discussed in section III) it became apparent we had to use the laser diode that came with the stock Kinect. This forced us to choose the +/- 2 nm option while increasing the instantaneous power by a factor of 4. It was acceptable for us to use such a restrictive filter since in using the Kinects laser, we could also maintain the thermal regulation system. This ensured that the laser diode would maintain a very strict wavelength of 830 nm.

In addition to the choice of filter, an external shutter was added outside the filtration system. This was chosen in order to prevent extra saturation from the sun on the camera that observes the laser grid on the environment while the laser is off when it is being pulsed. The shutter follows the design in a paper by Scholten *et al.* which uses the voice coil actuator from a mechanical hard drive for an affordable, easily manipulated shutter [11] [12].

C. *Optics: Control*

The main goal of the optics control block is to modify the 3D grid emission system in order to raise the total power radiated from the system. The original Kinect 360 uses a 100 mW laser diode located in the projector assembly. It passes through a focusing lens and, as it passes through the diffraction grating, it creates a dot pattern projected on the environment. The task of this block is to increase the output power of the laser diode so that it has enough power to be detected by the IR camera in direct sunlight and make it work with the rest of the system [13].

This goal was achieved with three designs. Shown in figure 4, main design creates a trigger that goes to the driver for the shutter and laser diode. The driver for the laser diode is simple and therefor lumped in this design. The frequency and duty cycle that appear in the trigger are very sensitive to the power supply so the second part of the design converted the power from a car battery to a stable 9.1 V. Lastly the driver for the shutter was replicated and applied to our design as specified in Scholten [12].

The trigger generation shown in figure 4 has three main parts. The first is a 555 timer which generates the trigger. This circuit uses a 555 timer to create a pulse which is set by the charging and discharging of a capacitor. The duty cycle is set by the diode/resistor network directly beneath the top timer in figure 4. After the trigger is created, it is passed to the shutter driver and passed through a buffer to the driver for the diode. The lower 555 timer creates a negative voltage in order to pull the buffered signal down to 0V when the laser is off. The Laser driver uses a power mosfet and resistor to create a constant current to pulse the laser diode. This current is then pulsed on and off by observing the trigger at the gate of the diode.

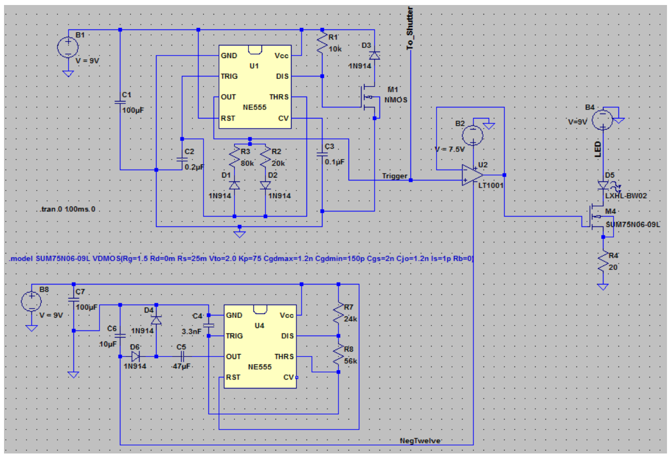

Fig. 4. Create a pulse with a 555 Timer

The trigger ultimately creates a frequency of 60 Hz which was chosen to be twice the cameras shutter speed. This ensures when the camera is open it will always observe the environment with at least one pulse of the grid on it. To maintain normal functionality of the laser while increasing its continuous wave power output we chose a duty cycle of 20% and increased the current through it by a factor of four to 400 mA. This brought the continuous wave power output of the laser diode to approximately 390 mW which satisfies the criteria for the Kinect 360 to work outdoors under any conditions if it utilize the +/- 2 nm filter.

The power module shown in figure 5 uses the LMZ35003 to ensure that the pulsing circuit is reliable with a car battery no matter what the actual voltage might be. A typical car battery can vary in voltage from 14 to11 V. From this the circuit was created to accept an input from 10.5 to 15 V and have the consistent output of 9.1 V to power the trigger. The design was created using a tool that TI offers which selects a part and suggests a design with the necessary input/output specifications.

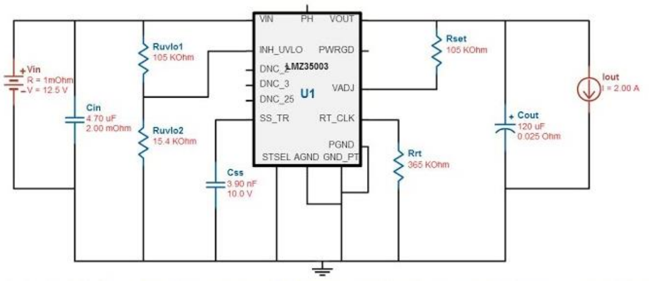

Fig. 5. DC to DC converter with power regulation

D. *Rover: Navigation Input*

The navigation system relies on GPS in order to get from one location to another. While separate from the obstacle avoidance system, the GPS is an integral part to knowing the rover's location within an accuracy of three meters. Documentation on this component of the rover was severely lacking, so the system had to be reverse engineered from snippets of code distributed throughout the rover's onboard computer.

To perform these tasks, skills from previous experience with Linux had to be used. In addition, ROS had to be learned. This was done mainly by reading the tutorials found on the ROS wiki [4] as well as using the ROS answers forum. The ROS answers forum was invaluable in debugging pieces of ROS code because the creators of ROS have answered many questions that involve the intricacies of the software system. Basics of code understanding taught in introductory programming were also employed in order to make this subblock functional. Communication skills learned in team experiences from iCons were also used in order to effectively contact previous students with experience on the rover.

In order to launch the GPS system, first one must connect the main computer to the RFLEX computer and initialize the ROS core. Next, the WiFi needs to be broadcasted and configured for external device connection. For our GPS, we used an Android phone and connected it to the rover's configured WiFi. On the phone, we installed an application called ROS Sensor [14] and input the ROS core's IP address. Back on the rover, the GPS goals need to be modified, establishing a destination. Next, a program needs to be run in order to publish the distance of the rover from the destination. Lastly, a separate program is run to send the rover to its destination.

The GPS has an accuracy of approximately three meters, so this tolerance is taken into account while sending the rover to its destination. As previously demonstrated, if the rover has a tolerance of less than three meters to its location, it will continue to circle in the approximate area of the destination since the GPS cannot accurately establish that it is within three meters to the destination. This is where some fine-tuning needs to be done, such that the rover will arrive at the destination and not circle it.

Unfortunately due to the Rover team's setback described in section III, GPS tuning was not completed for our project. Instead we did all our final testing and demoing with a keyboard command function that we wrote. This function takes WASD keyboard input and places a goal 1 m away in the corresponding direction so the rover can display its local path planning and object avoidance capabilities. The GPS navigation works with the same mechanic by placing goals at the specified locations, so it follows that our modified Kinect project will be usable with the GPS navigation.

We have left the GPS in a working state, so the next group working on the rover project can utilize either or both of the GPS and Kinect technologies. We recommend that they first confirm GPS accuracy against a landmark GPS coordinate and then do all their tuning with that accuracy in mind.

E. Rover: Navigation Stack

The rover navigation system is in charge of driving the rover to a given destination while avoiding all obstacles on the way. It takes inputs from the previous block that tell it where to go and what the environment looks like in front of it. With goals generated by either the GPS or keyboard navigation, this block computes the best path to get there. This path takes into account the environment data and avoids getting too close to any points it determines are obstacles. While this is happening, it also monitors the environment for any changes that might disrupt the global path and updates the local path accordingly. Additionally there are protocols that will rotate the rover in order to collect data about the immediate area, that is, it makes movements with observational motivation rather than goal driven motivation.

In order to create these functionalities, we used the ROS navigation stack [15] and the RFLEX driver [16]. The ROS navigation stack is a standard package for controlling robot movement. It is designed to be fully configurable and then fully automatic. Configuration options include which sensors to subscribe to, how those sensors physically relate to the body of the robot, and different parameters to control the behavior of the pathing. Once all of the required configurations have been made, navigation is as simple as publishing goals to a topic. Our implementation of the navigation stack is designed around the speed of the rover and the limitations of the Kinect such that we don't get so close to an obstacle that we can no longer see it. Lastly the RFLEX drivers are specific for the ATRV-Jr and allow for ROS to communicate with the rover's physical systems without any configuration on our end [16].

The main difficulty of this block was learning the entirely new ROS environment and more specifically the navigation stack. Just like with the previous block we had to learn how ROS pieces together and the functions of all of the files involved. When adding the navigation stack on top of that, we had to learn about tf transforms [17], goal publishing, and all of the configuration parameters. Tf transforms are how the navigation stack relates sensors to the robot in order to interpret the geometry of the sensor data. This was particularly easy to learn as we managed to capture the relationship in a one line, static, xyz relationship. Goal publishing was also easy to learn as it is just a more specific type of ROS topic and we were able to learn from examples we found on the ROS wiki. Configuration parameters were much harder to learn because it's one thing to read the description of a parameter in the documentation, and it's another to see it driving around the room.

Before MDR, we were stuck using a sort of blind trial and error to learn the parameters for the navigation stack. We would drive the rover straight at obstacles as well as give it long distance goals and see how it reacts. This experimentation was meant to get the system into a rough working state for MDR due to a computer crashing problem that prohibited us from using the visualization software. These experiments were very inefficient in nature because we were not actually monitoring the software output of the pathing algorithm, we were watching the hardware output which could introduce another layer of bugs.

Once we fixed the crashing problem (discussed in section III), we were able to actually monitor the path as it's made in real time and see any disconnect between software and hardware. We were also able to see decisions that were being made in advance, instead of just the decisions that were currently being executed. With this better insight we were able to experiment with different orientations of the Kinect in order to find the position that lets us see both the closest and widest areas in front of the rover. It turned out that the best place for the Kinect was on a shelf attached to the main tower. By placing the Kinect back on the rover body and aiming it downwards, we were able to reduce the amount of blind area in front of the rover. After we mounted the Kinect up high, we repeated the configuration tuning experiments, except with more informative results, until we reached a configuration that we were satisfied with.

Debugging the software that drives a physical system is very natural after having taken Computer System Lab. It is certainly a different task than debugging a purely software system because all of the outputs are analog and some of the symptoms can be disguised in that sense, however the hours spent in Duda have prepared us for this. Beyond that, just general good coding practice picked up across different courses have helped this technical block a fair deal and, of course, ROS answers was invaluable for this block as well.

III. Project Management

As our project concludes, it is clear that we have accomplished a balance of practical and theoretical tasks. The Rover team had very concrete goals based around creating systems on the rover and tuning their performance. On the other hand the Kinect team had much more physics and electronics oriented goals aimed at fully characterizing the functionality of the Kinect in order to improve its capabilities. The two subteams also benefitted from independence because they have two distinct work styles. While the Rover team had a very iterative and functionality driven project, the Kinect team had a project that required intensive measurements before committing to hardware. There is a high amount of precision involved when working with these optics, and our budget would not have been able to handle many purchases that didn't make it into the final project. While the Kinect team ran their experiments, the Rover team was not held back waiting for the modified Kinect to be finished and so parallel progress was achieved.

That being said, the Rover team successfully implemented the systems it set out to create. We ran into some difficulty at the beginning of the semester as we tried to fix the previously mentioned crashing problem. As a brief overview, the rover contains two computers: one which drives the proprietary rover hardware and one which runs the ROS software. The one running the ROS software was the one with the crashing issue. While trying to find the cause of the crashing, the motherboard stopped working and then soon after we found burn marks on the board supplying the motherboard with power. We came to the conclusion that this faulty power board was damaging the motherboard and causing the crashing. As a result we had to construct an entirely new computer using recycled parts found around campus and find a new way to deliver power to it. By the end of the semester we restored full functionality and were able to successfully tune the obstacle avoidance such that the rover avoided all obstacles we tested on indoors.

The Kinect team successfully designed and ran experiments to define many important constraints on their design. Throughout the semester we discovered some new complications that we ultimately couldn't solve. When we first constructed the modified Kinect we found that it crashed the OpenNI software after a short period of use. Considering that we had to remove the cooling system to install our new laser, we figured this was a heat issue and dedicated a large amount of time trying to fix that problem. After some experimentation with a new Kinect we found that the crashing happens whenever we are using a different laser, even if we attach a stock Kinect laser as well.

This indicated that the cooling system removal was not the fundamental problem; instead a component of the OpenNI code was realizing the changes made in the system, and reacting by shutting down the system. By the time we discovered this, we did not have enough time left to try and modify the OpenNI software to allow for our pulsing circuit. In the end the Kinect team provided two versions of the final product. One Kinect was fully modified with a pulsing laser, shutter, and bandpass filter while the other was simply the bandpass filter. While the fully modified Kinect has better performance outside, the filter version worked well enough that we did not hit anyone during our outside demonstrations. We are leaving both versions with Professor Parente for the next rover team in case they can solve the crashing issue.

Our team dynamics worked very well due to our divided efforts into two sub-teams. We had a problem where the goals of the group were very clear, but not so much for the individual. After splitting our goals in two, it was a lot easier for members to claim responsibility for aspects of the project.

The team had an appropriate spread of expertise for this project. Alex McNeil is an Electrical Engineer with a minor in Physics which made him perfect for the Optics design on the modified Kinect. Alex Maerko is by far the most hands on member of the team and has excellent circuit design skills, which was incredibly helpful with hardware design and implementation for the modified Kinect, and other hardware on the rover. Gabriela Correa has a strong Matlab and Linux background, which has proved useful in data analysis and rover system debugging/maintenance respectively. Timothy Tufts, being the only Computer Systems Engineer on the team, was crucial in getting rover software created and running properly.

There was healthy communication between each sub-team in the form of contacting individuals from the other team to help with specific tasks. Notable among these were Alex Maerko helping the Rover team by creating the power supply for the indoor Kinect and Gabriela Correa helping with image analysis on the Kinect team. To communicate, we had two weekly meetings as well as shared Google Drive / Dropbox storage and a Facebook message thread for immediate contact. Besides that, the team really pulled through with helping struggling members and putting in the work hours when they were needed.

IV. Conclusion

At the conclusion of the senior design project course, we would call our project a success. Throughout the year we were plagued by bad specification design, for example the specifications from tables I and II turned out to be unnecessarily specific for what we were trying to accomplish. As a result we simplified our specifications to be measurable field of view and then a barrage of tests comparing the performance with the unmodified and the modified Kinect to ensure that we preserve indoor performance and improve outdoor performance. Of the field of view specifications presented in table IV we managed to meet all but the height which was arguably the least important.

TABLE IV
Field of View specifications

Field of View	Goal (m)	Unmodified Kinect (m)	Modified Kinect (m)
Min Range	.55	.47	.52
Max Range	2.5	4	3
Height	1.07	.87	.84
Width	.65	1.31	.8

As for the performance tests, we saw that the modified Kinect performed almost exactly as well as the unmodified Kinect indoors even with the field of view loss. Most importantly our modified Kinect was able to successfully visualize obstacles outdoors where the unmodified Kinect would not see anything reliable. On SDP demo day, we were able to run live tests where guests would stand in front of the rover and the rover would sense them and drive around.

When we first approached the problem by researching previously implemented solutions, we found that all the best solutions were highly expensive. This led us to reshape our project and aim to build a low-cost 3d environment sensing system for outdoor use. Where the most inexpensive solution cost around $8,000, our cost estimates in table V show that we could market this product for around $150. So it is safe to say that we very comfortably met the low cost objective of our project.

TABLE V
PRODUCTION COST FOR 1000 UNITS

Part	Price
Microsoft Kinect 360	$ 15,000.00
+/- 2 nm Bandpass Filter	$ 81,283.20
Hard Drive	$ 19,000.00
LMZ35003	$ 10,282.50
LMD18200	$ 9,112.50
NE555	$ 230.00
Diodes	$ 777.00
Resistors	$ 30.29
Capacitors	$ 3,913.00
TOTAL	$139,628.49

Lastly, our own experience with the rover project has prompted us to prepare documentation for the future rover projects. Most importantly we have provided Professor Parente with an addition to his rover booklet that explains the current state of the rover and how to use our new hardware. We have also consulted with Professor Parente so he knows needs work and what future groups could work on. Inevitably this will not be enough so they will also have our contact information and we can help get them started as the SDP14 team helped us.

ACKNOWLEDGMENT

The authors would like to thank Professors Jun Yan, Bill Leonard, and Christopher Salthouse for their assistance with the optical design, Professor Robert Jackson for his assistance with circuit design, Fran Caron for all around help, Professor Mario Parente and MIRSL for providing us with the rover and lab space, and lastly SDP14 Team AIR and Keval Patel for their assistance in reviving old functionality in the rover.

REFERENCES

[1] W. Isaacson, “How Google Went from School Project to Global Phenom | Inc.com,” 17-Oct-2014. [Online]. Available: http://www.inc.com/walter-isaacson/how-google-got-its-start.html. [Accessed: 15-Dec-2014].
[2] D. Tam, “Meet Amazon’s busiest employee -- the Kiva robot - CNET.” [Online]. Available: http://www.cnet.com/news/meet-amazons-busiest-employee-the-kiva-robot/. [Accessed: 15-Dec-2014].
[3] J. Layton, “How Robotic Vacuums Work - HowStuffWorks.” [Online]. Available: http://electronics.howstuffworks.com/gadgets/home/robotic-vacuum.htm. [Accessed: 15-Dec-2014].
[4] Kurt Konolige and Patrick Mihelich, “Technical description of Kinect calibration,” *ROS Wiki*. [Online]. Available: http://wiki.ros.org/kinect_calibration/technical. [Accessed: 15-Dec-2014].
[5] “ATRV-Jr^{TM} MOBILE ROBOT TECH SHEET.” Real World Interface.
[6] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool, “3d urban scene modeling integrating recognition and reconstruction,” *Int. J. Comput. Vis.*, vol. 78, no. 2–3, pp. 121–141, 2008.
[7] J. Levinson and S. Thrun, “Robust Vehicle Localization in Urban Environments Using Probabilistic Maps,” Stanford Artificial Intelligence Laboratory.
[8] “Velodyne Lidar.” [Online]. Available: http://velodynelidar.com/lidar/lidar.aspx. [Accessed: 15-Dec-2014].
[9] P. E. Ross, “Tesla’s Model S Will Offer 360-degree Sonar,” *IEEE Spectrum*, 10-Oct-2014.
[10] D. A. Schoenwald, J. T. Feddema, and F. J. Oppel, “Decentralized control of a collective of autonomous robotic vehicles,” in *American Control Conference, 2001. Proceedings of the 2001*, 2001, vol. 3, pp. 2087–2092.
[11] Maguire, L. P., S. Szilagyi, and R. E. Scholten. "High Performance Laser Shutter Using a Hard Disk Drive Voice-coil Actuator." Review of Scientific Instruments 75.9 (2004): 3077. Web.
[12] Scholten, R. E. "Enhanced Laser Shutter Using a Hard Disk Drive Rotary Voice-coil Actuator." Review of Scientific Instruments 78.2 (2007): 026101. Web.
[13] Tim Carmody, “How Motion Detection Works in Xbox Kinect | WIRED,” 03-Nov-2010. [Online]. Available: http://www.wired.com/2010/11/tonights-release-xbox-kinect-how-does-it-work/all/. [Accessed: 15-Dec-2014].
[14] Chad Rockey, *ROS Android Sensors Driver*. 2013.
[15] Eitan Marder-Eppstein, “Navigation,” *ROS Wiki*, 10-Jul-2014. [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 15-Dec-2014].
[16] Mikhail Medvedev and David V. Lu, “RFLEX,” *ROS Wiki*. [Online]. Available: http://wiki.ros.org/rflex. [Accessed: 15-Dec-2014].
[17] Tully Foote, Eitan Marder-Eppstein, and Wim Meeussen, “tf,” *ROS Wiki*. [Online]. Available: http://wiki.ros.org/tf. [Accessed: 15-Dec-2014].