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 modifying a Kinect to expand its functionality to outdoor environments. A novel optical system is described and under testing in order to accomplish this. Implementation with an ATRV-Jr mobile robot is shown, and future plans for final implementation are illustrated.**

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 tasks that robots cannot perform, such as innovative thinking.

In Tables I & II we give the specifications for the two main systems of our project. Table I describes the requirements for the Optics systems. 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 are the GPS speed and accuracy which are 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 rest of the parameters are meant to meet the default settings for the 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 Nav 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 as much as possible for an extremely adaptable solution. As a result our solution has two main components: the Kinect System and the Robot System. The Kinect System is the modified hardware we would market to hobbyists, while the Robot System represents any robot running the OpenNI software for Kinect. In our case, the Robot 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, but much more expensive systems have used similar methods to achieve the same result. 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.

Two main alternatives were considered in the first iteration of the LESS hardware. 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 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 $8000 [8]. This would undermine the goal of our project which is to bring our solution to the hobbyist at a reasonable cost.

Another main competitor to the system was using 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, 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 to develop the two largest needs of the design. The Optics block strives to create a version of the Kinect which can work in direct sunlight outdoors. While the Rover block creates

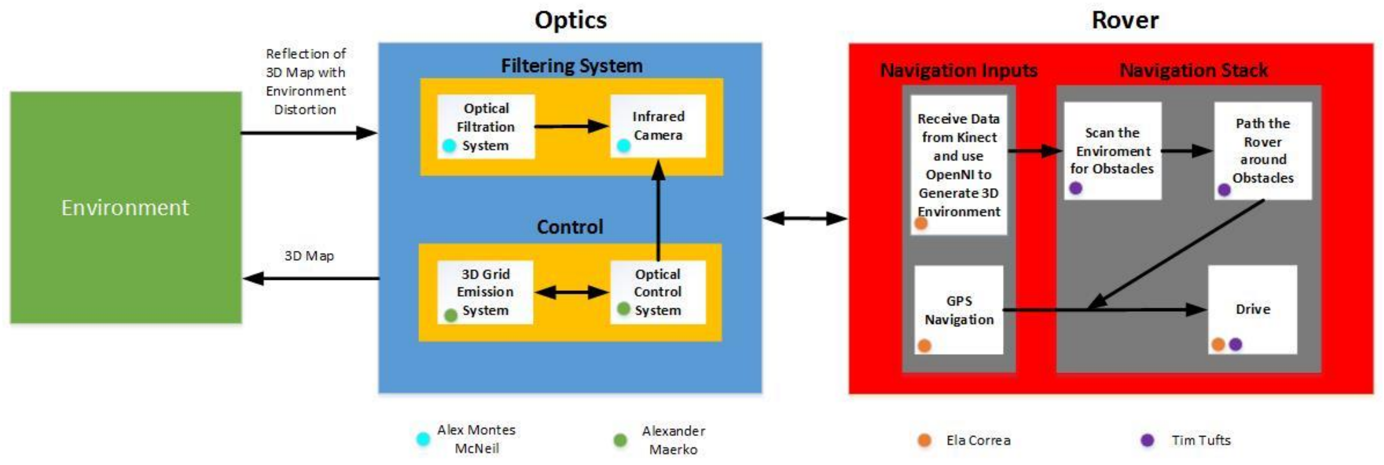

Fig. 1. Block Diagram showing the separation and internal workings of the two main systems.

a software system that can be easily replicated in a wide variety of robotics projects when implemented with the optics block.

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 using 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 image meaning the pixels were fully saturated. The background

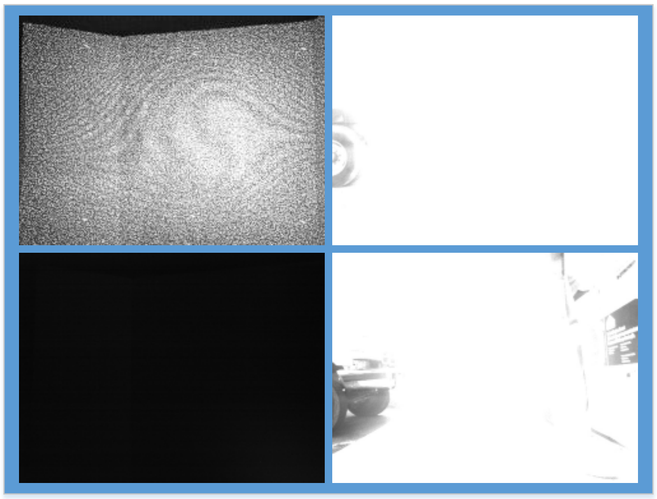

Fig. 2. Top Left: Indoors with Kinect IR; Top Right: Outdoors with direct sun; Bottom Left: Indoors without IR; Bottom Right: Outdoors Cloudy

difference between outdoors and indoors is almost the entire scale of the IR camera.

Once it was clear that the Kinect was failing because its pixels were fully saturated, a variable IR projector was created that could mimic these conditions. This source 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.

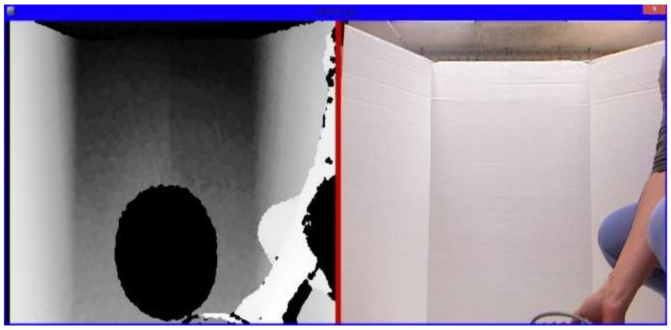

Fig. 3. IR Interference with Kinect 3D Depth map

The amount of total emitted power necessary to interfere with the Kinect 3D generation was 74 mW. This is validated because the total amount of power emitted by the Kinects IR projector is approximately 60 mW. 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, is greater than the total amount of power the Kinect is emitting on the environment.

From this result the design process for the optical filtering system was to remove as much power from the sun as possible and then to 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 3.

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 830nm +/- 100nm filter on it meaning normally it could experience up to 544W from the sun. This is orders of magnitude above the Kinects 60mW laser which is why its 3D generation technique does not work outside.

This leaves two design options for the Kinect with the other two band pass filters. The specification for the 10nm filter requires an instantaneous power of 5.8W out of the laser. If a more accurate IR projector is used with the 2nm filter, the IR projector needs an instantaneous power emitted of 316mW.

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

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 60mW 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 [11].

The design chosen increases the instantaneous power of the laser. This accomplishes the goal of increasing the intensity of the projector from the Kinect's perspective while maintaining the same average power and temperature. We chose a 10% duty cycle for the 1W laser diode we will be using in our IR projector. The pulsing circuit topology can be viewed in Figure 4.

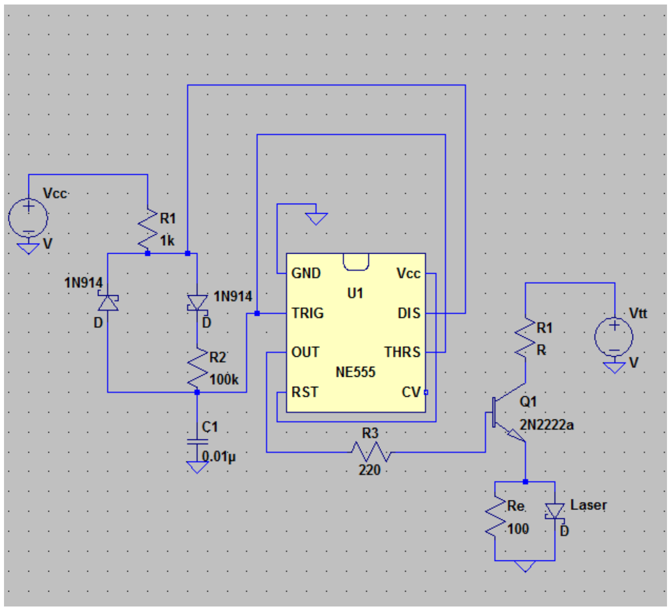

Fig. 4. Create a pulse with a 555 Timer

The 555 timer and 1N914 switching diodes will be used to enable complete control of the on/off time for the pulse. The decoupling capacitors of 0.1 µF and 5 µF control the period of the pulse. A 220 ohm series resistor in the 2N2222a base circuit is used to set the base drive level. A 10 ohm collector resistor provides suitable current limiting. The LED in the emitter circuit is in parallel with 100 ohm resistor to speed up the fall time of the charge drain. The 555 and transistor are biased at the same adjustable level with a common power supply. This circuit is designed to operate at frequency ~ 1.4 kHz and Tp ~ 7 µs, requires 1 Amp at 12V which will be easily provided by the two 12V, 35 Ah batteries on the rover.

A series of experiments will be designed in order to test the integrity of the final system. The main concern is heat dissipation as the IR projector we are building will be utilizing much more power than the original IR projector and therefore will require a much better cooling system. Our task is to make sure the temperature of the laser diode remains constant and does not exceed the limit of 102°C. This is our primary requirement, because the Kinect has a temperature sensor built in for safety measures, which is programed to shut off the power to the IR projector in case of overheating [12].

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 app called ROS Sensor [13] and input the ROS core's IP address. Back on the rover, the GPS goals needs 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.

To test and fine-tune the GPS system for accuracy we plan to continue test runs in the engineering quad after fine-tuning the system from our last run where we observed the rover circle around its final destination. After establishing the rover will stay at rest after arriving at its destination, we will measure the accuracy of arrival by using a landmark GPS coordinate we can acquire from overlaying Google Maps, and measure the rover's distance from the final destination. This will allow us to confirm the arrival accuracy is under three meters, as per subblock requirements.

E. Rover: Navigation Stack

The rover navigation system will be 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. This block takes the goal computed by the GPS system and computes a 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 path and updates the 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 are using the ROS navigation stack [14] and the RFLEX driver [15]. 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[15].

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 [16], 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.

The trial and error that we used to learn the parameters of the navigation stack involved driving the rover straight at obstacles as well as giving it long distance goals and seeing how it reacts. This experimentation was meant to get the system into a rough working state for MDR. There are many different orientations we needed to and continue to try in order to characterize the pathing of our configuration. For example, we can test the horizontal field of view and obstacle buffer zone parameters by placing obstacles off center in front of the rover until the rover no longer recognized them. These experiments are very inefficient in nature because we are not actually monitoring the software output of the pathing algorithm, we are watching the hardware output which could introduce another layer of bugs. We cannot monitor the software output because the rover's computer crashes when put under a moderate amount of computational stress. We know this is a bug because we have plenty of artifacts showing SDP14 team AIR using programs that crash the computer now.

Once we can fix the crashing problem, we will be able to actually monitor the path as it's made in real time and see any disconnect between software and hardware. We will also be able to see decisions that are being made in advance instead of just the decisions that are currently being executed. With this better insight we will be 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. Then after we find that we can repeat the configuration tuning experiments, except with more informative results, until we reach a configuration that we are 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[link] was invaluable for this block as well.

III. Project Management

TABLE IV
MDR Goals

Goal	Progress
Indoor Obstacle Avoidance	Tuning Stages
GPS Functional	Tuning Stages
Filtration System Design	Experimentation Stages
Control System Design	Experimentation Stages

So far in our project, we have accomplished a balance of practical and theoretical tasks. The Rover team has very concrete goals based around creating systems on the rover and tuning their performance. On the other hand the Kinect team has 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 benefit from independence because they have two distinct work styles. While the Rover team has a very iterative and functionality driven project, the Kinect team has a project that requires intensive measurements before committing to hardware. There is a high amount of precision involved when working with these optics, and our budget would not be able to handle many purchases that don't make it into the final project. While the Kinect team runs their experiments, the Rover team is not held back waiting for the modified Kinect to be finished and so parallel progress is achieved.

That being said, the Rover team has implemented the systems it set out to create, but we still need to tune and improve those systems to a passable level. Specifically we have successfully run GPS navigation and indoor obstacle detection and avoidance with the Kinect. However the GPS interface is rather clunky and a better mounting position for the GPS device could be created to reduce noise. Also, the indoor pathing algorithms have trouble with the field of view of the Kinect, so we need to

	Sept.				Oct.				Nov.			
Task	1	2	3	4	1	2	3	4	1	2	3	4
Preliminary Research	X	X	X	X	X							
Project Design		X	X	X	X	X						
PDR Preparation					X	X	X					
Optics - Experiments						X	X	X	X			
Project Redesign							X	X	X			
Rover - GPS							X	X	X	X	X	
Rover - Obstacle Avoidance								X	X	X	X	X
Rover - System Maintenance							X	X	X	X		
Optics - Final Filter Design Solution										X	X	X
MDR										X	X	X

Fig. 5. Fall Gantt Chart of what has been done.

redesign the positioning of the Kinect as well as tune the pathing parameters to avoid visual deadzones.

The Kinect team has designed and ran experiments to define many important constraints on their design, however we need to run some more specific experiments over the break to get ready for the implementation of the design in the spring. For example, we will study the possibility of adding polarization into the modified system. After some key design choices are made, some experiments will be run to test the thermal and intensity integrity of the control block. After the design is implemented in the spring, more experimentation will be done to ensure we stay within the system operating constraints. While the design is not complete for either system, it is an iterative process that has keeps getting more advanced and refined.

In both cases, we still need to design what exactly our demo day deliverables will be in order to properly showcase the work that we have done.

Our team has been working very well since we divided our 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 has an appropriate spread of expertise for this project. Alex McNeil is an Electrical Engineer with a minor in Physics which makes 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 is 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, is crucial in getting rover software created and running properly.

	Dec.				Jan.				Feb.			
Task	1	2	3	4	1	2	3	4	1	2	3	4
MDR Report	X	X										
Rover - GPS Design Update	X	X										
Optics - Shutter Design	X	X	X	X								
Rover - GPS Fine-tuning		X	X	X								
Rover - System Maintenance		X	X	X	X	X						
Optics - Polarization Design			X	X	X							
Rover - Avoidance Fine-tuning		X	X	X	X	X						
Optics - Modify Kinect					X	X	X	X	X			
Documentation						X	X	X				
Optics - Characterize Kinect									X	X	X	
Prepare for CDR										X	X	X

Fig. 6. Spring Gantt Chart of what will be done.

There is healthy communication between each sub-team in the form of contacting individuals from the other team to help with specific tasks. Notable among these are 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 have two weekly meetings as well as shared Google Drive / Dropbox storage and a Facebook message thread for immediate contact. Besides that, the team has really pulled through with helping struggling members and putting in the work hours when they are needed.

Figures 5 and 6 show the Gantt chart split between what has been done and what is planned. All of the experimentation and design that has been discussed can be seen in Figure 5, as well as the clear shift from October into November when the sub teams were established. In Figure 6 we see there is risk mitigation as we plan to have our modified hardware finished well before CDR. We have tasks such as tuning and documentation planned before CDR so we have a comfortable window of time in case the implementation does not progress as planned.

IV. Conclusion

Currently, the GPS system and the Obstacle avoidance systems are functional, however much fine-tuning needs to be done. The preliminary optical design is complete, however some design options such as the use of a polarizer still need to be chosen. At the beginning of the semester, we were faced with an engineering challenge posed by Professor Parente: to build an obstacle avoidance and navigation system for MIRSL's Mars Rover.

We first approached the problem by researching previously implemented solutions. All the best solutions were expensive, this led us to reshape our challenge: to build a low-cost 3d environment sensing system for outdoor use. After exploring many design alternatives, we settled on creating a new optical system for the Kinect and integrating it with the rover system.

The first challenges were to build our optical system, and integrate a Kinect with the rover systems. Now we have our optical design and a prototype of Kinect-rover integration, our future contains the task of building our optical system, and fine-tuning the rover navigation and Kinect integration. Final tests of the device will include the use of a specifically designed obstacle course in the engineering quad.

We expect to face technical difficulties with the optical system, and we will have to modify it in order to meet our system requirements. We have yet to overcome the challenge with the rover's onboard computer crashing as well. Once the Kinect's optical system is modified, we will also have to recalibrate the Kinect-rover integration. This winter we will continue our work, and by spring we hope to be mostly debugging and establishing that system requirements are met through specific system tests, such as the obstacle course.

Acknowledgment

The authors would like to thank Professor Jun Yan and Professor Bill Leonard for their assistance with the optical design, 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] 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].
[12] "Hardware info," *OpenKinect*. [Online]. Available: http://openkinect.org/wiki/Hardware_info. [Accessed: 15-Dec-2014].
[13] Chad Rockey, *ROS Android Sensors Driver*. 2013.
[14] Eitan Marder-Eppstein, "Navigation," *ROS Wiki*, 10-Jul-2014. [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 15-Dec-2014].
[15] Mikhail Medvedev and David V. Lu, "RFLEX," *ROS Wiki*. [Online]. Available: http://wiki.ros.org/rflex. [Accessed: 15-Dec-2014].
[16] Tully Foote, Eitan Marder-Eppstein, and Wim Meeussen, "tf," *ROS Wiki*. [Online]. Available: http://wiki.ros.org/tf. [Accessed: 15-Dec-2014].