2016

Development of a mobile robotic phenotyping system for growth chamber-based studies of genotype x environment interactions

Dylan Shah
Iowa State University

Development of a mobile robotic phenotyping system for growth chamber-based studies of genotype x environment interactions

by

Dylan Shah

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Agricultural and Biosystems Engineering

Program of Study Committee:
Lie Tang, Major Professor
Stephen H. Howell
Steven James Hoff

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

Page

LIST OF FIGURES iv
LIST OF TABLES vi
ACKNOWLEDGMENTS vii
ABSTRACT viii
CHAPTER 1. INTRODUCTION AND THESIS FORMATTING 1
References 4
CHAPTER 2. DEVELOPMENT OF AN AUTONOMOUS INDOOR PHENOTYPING ROBOT 6
2.1. Abstract 6
2.2. Introduction 7
2.3. Materials 10
2.3.1. Background on hardware and design 10
2.3.2. System hardware 12
2.4. Method 13
2.4.1. Overview 13
2.4.2. A brief note on control strategy 14
2.4.3. Pre-Processing of Kinect depth data 17
2.4.4. Describing Kinect data in robot-base coordinates 19
2.4.5. Obtaining position and normal 23
2.4.6. Calculating desired robot pose 24
2.5. Results and Discussion 25
2.5.1. Mobile rover 25
2.5.2. Stationary rover, positional accuracy test 27
2.5.3. Stationary rover, surface normal test 29
2.6. General Discussion and Conclusion 32
2.6.1. Acknowledgements 34
2.6.2. Safety emphasis 34
References 35
CHAPTER 3. PATH PLANNING OF A ROBOTIC ARM IN A GROWTH CHAMBER 38
3.1. Abstract 38
3.2. Introduction 38
3.3. Related Work 40
3.4. Background on the Application 43
3.5. Materials and Methods 45
3.5.1. 3D PRM 46
3.5.2. PRM for 6-DOF robot arm 50
3.5.3. 6D PRM variants 53
3.5.4. Experiment, with motion 56

3.6. Results and Discussion 57
3.7. Conclusion 59
References 60
CHAPTER 4: GENERAL CONCLUSIONS 63
APPENDIX A: INSTRUCTIONS FOR BASIC USE OF ROVER BASE 65
APPENDIX B: BUILD OUTLINE 68

LIST OF FIGURES

Page

Figure 1. The chamber arrangement. 10

Figure 2. The Enviratron rover. 12

Figure 3. Software hierarchy. Line denotes a communication line, and arrows show direction packets are sent. 14

Figure 4. Psuedocode for track choice. 16

Figure 5. A sample fused image 19

Figure 6. Main UR coordinates. B: Robot base, E: end-effector, C: camera coordinates. 20

Figure 7. Error (Euclidian distance) [mm] for methods D, F, and M on a stationary rover. .. 28

Figure 8. Probing angled white paper. 30

Figure 9. Angle [deg] for methods D, F, and M on a stationary rover. 90 is perfectly normal. 32

Figure 10. Probing examples. Left: collision with a plant. Right: no collision. 44

Figure 11. Four artificial plants. 1: Pink Germanium, 2: Diffenbachia, 3: Dracaena, 4: Gold Guzmania. 47

Figure 12. Sample 3D path generation. Black dots: dilated obstacles (sparsely plotted), gray: KinectFusion mesh, green dots: sparse path travelled, yellow circles: nodes visited. 49

Figure 13. Sample graph. Black dots: dilated obstacles (sparsely plotted), blue dots: nodes, gray: KinectFusion mesh, green dots: sparse path travelled, yellow circles: nodes visited. 49

Figure 14. UR 10 modelled by spheres. 51

Figure 15. Default PRM 52

Figure 16. Adding nodes near plants 54

Figure 17. Adding nodes near end-effector at desired pose 55

Figure 18. Adding nodes near low-neighbor nodes 56

Figure 19. PRM example: the PRM framework generated paths approaching each goal location in succession. Starting with top left, left-to-right, top-to-bottom (in visitation order): right, top, and back of an artificial Pink Germanium; top and left of an artificial Diffenbachia; right and top of an artificial Dracaena; and back, top, and right of an artificial Gold Guzmania. 57

Figure 21. Roboteq Motor Control Utility for most MC/Tape sensor troubleshooting and settings. 65

Figure 22. C++ Roboteq test code uses RoboteqDevice class and sample.cpp. 66

Figure 23. Use green Serial/RS232 wires to communicate with computer. 67
Figure 24. Solidworks of rover. 68
Figure 25. Frame, motors, and LoveJoys. 69
Figure 26. Mecanum wheels, bearings, and collars. 69
Figure 27. Left: Some plating, bus bars, and electrical components added. Right: get creative. Sometimes taping a screw onto an Allen wrench aids in adding a difficult screw. 70
Figure 28. Shortening MagSensor cables. 70
Figure 29. Shorthand wiring diagram. 71
Figure 30. "Power Side" of the rover. 72
Figure 31. "Brain Side" of the rover. 73
Figure 32. Finished rover, without end-effector sensors (optional) 74

LIST OF TABLES

Page

Table 1. Euclidian distance error [mm]. “Avg.” means “average”. 27

Table 2. Means and Standard Deviations 28

Table 3. Student’s t Comparison of Means 28

Table 4. Means and standard deviations [deg]. 50 trials each level (method). 32

Table 5. Student’s t comparison of means [deg] 32

Table 6. 3D PRM results, averaged over 25 trials. $N = 150$ for each trial. 48

Table 7. Testing PRM variants. Ns is nodes for each trial’s initial roadmap (before enhancements, if any). Construction time [s], search time [s], error [deg], and Euclidian distance error (m) averaged over 25 trials. Each trial consisted of 10 separate goal locations. 59

ACKNOWLEDGMENTS

There are many people who influenced the content in this thesis, and who influenced my broader success in graduate school. I would like to thank those who encourage a pragmatic, adaptive, open-minded approach to problem solving. Specifically, these people include Dr. Tang, my fellow labmates, several of my Iowa State University professors, and my parents, Drs. Lori and Sanjay Shah.

I would like to thank my labmates for their contributions, whether this was something physically tangible such as helping assemble part of the rover, or something intellectual such as providing suggestions on the probing algorithm: Yin Bao, Jingyao Gai, Layne Goertz, Hang Lu, Austin Plotz, Rajesh Putta-Venkata, Caleb Stafford, and Taylor Wisgerhof. An additional thanks goes out to Yin Bao for answering many questions about programming on a Windows system, Zhanhong Jiang for academic career guidance, and my committee members for their guidance during this master's program and suggestions on this thesis.

This material is based upon work supported by the National Science Foundation under Grant No. 1428148.

ABSTRACT

In order to fully understand the interaction between phenotype and genotype x environment to improve crop performance, a large amount of phenotypic data is needed. Studying plants of a given strain under multiple environments can greatly help to reveal their interactions. This thesis presents two key portions of the development of the Enviratron rover, a robotic system that aims to autonomously collect the labor-intensive data required to perform experiments in this area. The rover is part of a larger project which will track plant growth in multiple environments.

The first aspects of the robot discussed in this thesis is the system hardware and main, or whole-chamber, imaging system. Semi-autonomous behavior is currently achieved, and the system performance in probing leaves is quantified and discussed. In contrast to existing systems, the rover can follow magnetic tape along all four directions (front, left, back, right), and uses a Microsoft Kinect V2 mounted on the end-effector of a robotic arm to position a threaded rod, simulating future sensors such as fluorimeter and Raman Spectrometer, at a desired position and orientation. Advantages of the tape following include being able to reliably move both between chambers and within a chamber regardless of dust and lighting conditions. The robot arm and Kinect system is unique in its speed at reconstructing an (filtered) environment when combined with its accuracy at positioning sensors. A comparison of using raw camera coordinates data and using KinectFusion data is presented. The results suggest that the KinectFusion pose estimation is fairly accurate, only decreasing accuracy by a few millimeters at distances of roughly 0.8 meter. The system can consistently position sensors to within 4 cm of the goal, and often within 3 cm. The system is shown to be accurate

enough to position sensors to ± 9 degrees of a desired orientation, although currently this accuracy requires human input to fully utilize the Kinect's feedback.

The second aspect of the robot presented in this thesis is a framework for generating collision-free robot arm motion within the chamber. This framework uses feedback from the Kinect sensor and is based on the Probabilistic Roadmaps (PRM) technique, which involves creating a graph of collision-free nodes and edges, and then searching for an acceptable path. The variant presented uses a dilated, down-sampled, KinectFusion as input for rapid collision checking, effectively representing the environment as a discretized grid and representing the robot arm as a collection of spheres. The approach combines many desirable characteristics of previous PRM methods and other collision-avoidance schemes, and is aimed at providing a reliable, rapidly-constructed, highly-connected roadmap which can be queried multiple times in a static environment, such as a growth chamber or a greenhouse. In a sample plant configuration with several of the most challenging practical goal poses, it is shown to create a roadmap in an average time of 32.5 seconds. One key feature is that nodes are added near the goal during each query, in order to increase accuracy at the expense of increased query time. A completed graph is searched for an optimal path connecting nodes near the starting pose and the desired end pose. The fastest graph search studied was an implementation of the A* algorithm. Queries using this framework took an average time of 0.46 seconds. The average distance between the attained pose and the desired location was 2.7 cm. Average distance C-space between the attained pose and the desired location was 3.65 degrees.

The research suggests that the robotic framework presented has the potential to fulfill the main hardware and motion requirements of an autonomous indoor phenotyping robot, and can generate desired collision-free robot arm motion.

CHAPTER 1. INTRODUCTION AND THESIS FORMATTING

The 21st century presents many challenges, many of which will inevitably involve food. Organizations such as the Royal Society of London (2009) and the FAO (Rai, Reeves, Pandey, Collette, & Food and Agriculture Organization of the United Nations, 2011) suggest a need for at least a 50% increase in food supply in the next half-century. Understanding how to sustainably improve yield in a changing climate is a complex task. This will require an intimate understanding of how existing and emerging crop species interact with their environment, especially if the genetic descendants of these crops are expected to survive climate change predictions as high as several degrees in many areas, and globally as high as 4.8 by some scenarios reported by the Intergovernmental Panel on Climate Change (IPCC, 2014). Studying crops requires large quantities of data under every environment studied, so manually gathering data for studying plants' response to multiple climate scenarios will become prohibitive for many research groups.

As many important traits, such as yield, are complex traits (Pieruschka & Poorter, 2012), there have been many studies and companies relating to phenotyping. Phenotyping is the study of plants' expressed traits, in contrast to genotyping, or studying plant's genetics. The recent development of plant phenotyping techniques measuring plant parameters is often done using imaging. Example approaches include Time-of-Flight (ToF) cameras (Alenyà, Dellen, Fox, & Torras, 2013; Chaivivatrakul, Tang, Dailey, & Nakarmi, 2014), stereo image sequences (Aksoy, Abramov, Wörgötter, Scharr, Fischbach, & Dellen, 2015), and point clouds from the Microsoft Kinect v1 (Azzari, Goulden, & Rusu, 2013). These technologies are generally used to measure leaf area, plant height, and stem diameter. As the techniques mature,

these parameters should become more reliable and algorithms to extract additional plant parameters will undoubtedly be developed. This will, in turn, allow researchers to compile great quantities of repeatable data on their plants' growth to correlate the various inputs, such as water practices or soil nitrogen treatments (Neilson, Edwards, Blomstedt, Berger, Møller, & Gleadow, 2015) with observed growth patterns. These, in turn, can be used to identify areas for improving plants' treatments or optimize biomass yield, for instance. The studies previously mentioned are part of the generation of studies which often either uses a fixed robot arm or a human observer. Since a similar set of measurements are typically taken for many plants at many time intervals, the area is ripe for automation. Some agricultural robots aim to fill this role, including those at many universities, and the relatively new Rowbot (sold by Rowbot Systems).

While operating a robot arm on a sparsely populated environment, often the robot arm can be commanded to approach a location without colliding with anything. For instance, the robot arm used by Aksoy (Aksoy, et al., 2015) for imaging tobacco seedlings likely never hit their small plants during imaging. Even during the initial tests presented in this thesis, Chapter 2, the robot arm was operating without serious collisions. However, the situation is more complicated when there are obstacles to either reach over or around. This prompted the robotics community to seek solutions to various path-planning problems. Commonly, a distinction is made between approaches involving knowledge of the environment and those involving local sensors (Shaffer, 1991). Most modern methods use knowledge of the robot's environment. Some operate in static environments, such as the probabilistic roadmaps (PRM) method (Kavraki, Svestka, Latombe, & Overmars, 1996). Others operate in dynamic environments, such as the depth space approach proposed by Flacco et al. (Flacco, Kröger, De Luca, &

Khatib, 2012). The desired end result is a collision-free path from a start position and orientation (pose) to a goal pose.

Current phenotyping systems, such as those available through Lemnatec and used at the Plant Accelerator at the University of Adelaide (Neilson, et al., 2015), enable researchers to study the traits they desire on many plants in one environment. As far as the author is aware, there is currently no solution available to autonomously study plant phenotypes using multiple environments. This is the long-term goal and broad current knowledge gap that this research is targeting. It is part of the broader Enviratron project at Iowa State, "a facility to test and evaluate the performance of plants under variable environmental conditions" (NSF, 2014).

All research objectives were focused on making this broad research project a working reality. The work presented in this thesis is snapshot of the work required on this path, and all work is completed with the goal of being integrated together into a seamless robotic plant phenotyping system. New solutions are presented, and novel packaging of existing ideas helps to fill in gaps in the current literature, linking previously-separated ideas to push the envelope on robotic applications. The chapters in this thesis are aimed at becoming two independent journal papers: 1) development and results of the robotic system hardware and Kinect-level imaging system, and 2) a framework for path planning in constrained environment using the Kinect feedback in quasi-real-time. What is meant by quasi-real-time is that the map and plans are generated in a short time period, and the environment is static, allowing the rest of the interactions to happen at true real time speeds.

There are two supplementary process documents in the appendices. Appendix A is a light instruction manual for the robot base, intended for use in the Enviratron facility during operation. Appendix B contains pictures and SolidWorks drawings from the build phase of the

project. These appendices are presented to illustrate the function and completeness of the robot base.

References

Alenyà, G., Dellen, B., Foix, S., & Torras, C. (2013). Robotized plant probing: leaf segmentation utilizing time-of-flight data. *Robotics & Automation Magazine, IEEE,* 20(3), 50-59.

Aksoy, E. E., Abramov, A., Wörgötter, F., Scharr, H., Fischbach, A., & Dellen, B. (2015). Modeling leaf growth of rosette plants using infrared stereo image sequences. *Computers and Electronics in Agriculture,* 110, 78-90.

Azzari, G., Goulden, M. L., & Rusu, R. B. (2013). Rapid characterization of vegetation structure with a Microsoft Kinect sensor. *Sensors* (Basel, Switzerland), 13(2), 2384.

Chaivivatrakul, S., Tang, L., Dailey, M. N., & Nakarmi, A. D. (2014). Automatic morphological trait characterization for corn plants via 3D holographic reconstruction. *Computers and Electronics in Agriculture,* 109, 109-123.

Flacco, F., Kroger, T., De Luca, A., & Khatib, O. (2012). A depth space approach to human-robot collision avoidance. *Robotics and Automation (ICRA), 2012 IEEE International Conference on,* 338-345.

IPCC (2014): Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. *IPCC,* Geneva, Switzerland, 151 pp.

Kavraki, L., Svestka, P., Latombe, J., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on,* 12(4), 566-580.

Neilson, E., Edwards, A., Blomstedt, C., Berger, B., Møller, B., & Gleadow, R. (2015). Utilization of a high-throughput shoot imaging system to examine the dynamic phenotypic responses of a C cereal crop plant to nitrogen and water deficiency over time. *Journal Of Experimental Botany*, 66(7), 1817-1832.

NSF (2014). "MRI: Development of an ENVIRATRON - an accelerator for climate change research." NSF Award Abstract #1428248. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1428148 on 8-2-2016.

Pieruschka, R., & Poorter, H. (2012). Phenotyping plants: genes, phenes and machines. *Functional Plant Biology*, 39(11), 813-820.

Rai, M., Pandey, S., Collette, L., Reeves, Timothy G, & Food and Agriculture Organization of the United Nations. (2011). *Save and grow: A policymaker's guide to sustainable*

intensification of smallholder crop production / [final technical editing, Mangala Rai, Timothy Reeves and Shivaji Pandey; lead authors: Linda Collette ... [et al.]]. Rome: Food and Agriculture Organization of the United Nations.

Royal Society of London. (2009). Reaping the benefits: science and the sustainable intensification of global agriculture. London: Royal Society.

Shaffer, C. A. (1991). Real- time robot arm collision detection for telerobotics. C*omputers and Electrical Engineering*, 17(3), 205-215.

CHAPTER 2. DEVELOPMENT OF AN AUTONOMOUS INDOOR PHENOTYPING ROBOT

2.1. Abstract

In order to fully understand the interaction between phenotype and genotype x environment to improve crop performance, a large amount of phenotypic data is needed. Studying plants of a given strain under multiple environments can greatly help to reveal their interactions. To collect the labor-intensive data required to perform experiments in this area, an indoor rover has been developed, which can accurately and autonomously move between and inside growth chambers. The system uses Mecanum wheels, magnetic tape guidance, a Universal Robots UR 10 robot manipulator, and a Microsoft Kinect v2 3D sensor to position a threaded rod near a user-chosen point on the leaves of a Ficus plant and an artificial Dracaena, simulating the future use of sensors such as a fluorimeter and Raman spectrometer. Integration of the motor controllers, robot arm, and a Microsoft Kinect (v2) 3D sensor was achieved in a customized C++ program. Three-dimensional meshes representing plants inside the chamber were reconstructed using the Kinect SDK's KinectFusion. Image-processing functions were implemented to filter the depth image to remove undesired surfaces and noise, reducing the memory requirement and allowing the plant to be reconstructed at a higher resolution. This paper shows the system architecture and some preliminary results of the system, as tested using a setup which simulates a growth chamber. A comparison of using raw camera coordinates data and using KinectFusion data is presented. The results suggest that the KinectFusion pose estimation is fairly accurate, only decreasing accuracy by a few millimeters at distances of roughly 0.8 meters.

Keywords: growth chambers, mechatronics, robotics, software development

2.2. Introduction

In a world of changing climate and increasing world population, there is a great need to understand the interaction between genotypes and phenotypes in order to produce enough crop yield. Organizations such as the Royal Society of London (2009) and the (Rai, Reeves, Pandey, Collette, & Food and Agriculture Organization of the United Nations, 2011) suggest a need for at least a 50% increase in food supply in the next half-century. This will not be achieved without a drastic change in the way we grow our food. Sustainable intensification, involving increasing the productivity of existing farmland while reducing negative environmental impacts, is promoted as one of the best ways – and some would say the only way - to achieve this.

One of the main methods to increase crop yield without increasing chemical use involves plant breeding techniques. Effective plant breeding requires in-depth data on plants' health and growth patterns, which are part of their broader phenotype, or physical characteristics. Many traits relating to growth, performance, and yield are complex traits under polygenic control (Pieruschka & Poorter, 2012). Studying these traits using plants' phenotypes to understand how each strain behaves under various growing conditions is an important step toward improving the characteristics of a crop stock. This paper presents a technical solution to several issues relating to current phenotyping techniques.

Due to the large amount of manual labor required in traditional by-hand phenotyping methods, numerous studies and experiments have been exploring phenotyping methods that are based on images, often either RGB (red, green, and blue channels) or RGB-D (red, green, blue, and depth channels). For instance, one phenotyping technique uses infrared and depth images acquired with a CamCube time-of-Flight (ToF) camera (Alenyà, Dellen, Foix, &

Torras, 2013). The method involves taking a general view of the plant, segmenting to find leaves which are suitable for probing, and then moving the cameras closer to the suitable leaf using a Barret WAM arm. This leaf is then probed with a sample cutting tool. This points to the possibility for an application of a wide variety of sensors. For instance, fluorescence imaging sensors could be placed on the end of the robot arm for investigating the fluorochrome chlorophyll, which is involved in crop yield (Chaerle & Van Der Straeten, 2001). Numerous other sensors, such as near-infrared spectroscopy, can be applied to extract phenotypic data (Montes, Melchinger, & Reif, 2007).

Other image-based phenotyping techniques use infrared stereo image sequences to extract depth and then segment the resulting data to extract parameters such as leaf area and number of leaves (Aksoy, Abramov, Wörgötter, Scharr, Fischbach, & Dellen, 2015). Tobacco plants can be stereo-imaged periodically, using a KUKA robot arm. The image pairs are then run through an OpenCV block-matching algorithm to extract depth information. Next, the images are segmented to distinguish each leaf. Leaf area is found by ellipse-fitting each leaf, and the number of leaves is compared to the ground truth obtained via human measurement. Since these methods used fixed plant imaging positions, the methods are mainly suitable for stationary plants and a stationary robot arm, limiting the experiment to one growth environment. In a growing area larger than the reach of the robot arm, this approach also requires conveyance of the plants out of their growth environment, introducing other stress factors outside of the designed environment. Azzari et al. (2013) fused several point clouds, sometimes as many as 2000 point clouds per plant, to reconstruct the plant for extraction several pieces of information, including volume and allometric relationships. Point clouds were attained through manually moving their first-generation Kinect (v1). Chaivivatrakul et al.

(2014) used a plant rotating table to obtain and fuse several point clouds of corn plants into one 3D reconstruction. They then extracted traits such as leaf area, leaf length, and stem diameter. Phenotyping can also be done in the field, as demonstrated by Klodt, Herzog, Töpfer, and Cremers (2015). This method involves taking a pair of images of the same grapevine plant, on several different days. Once image pairs are acquired, they are rectified to extract depth information, and finally segmented to find leaf and stem areas. This method did not provide an automated image capturing technique, and did not provide a framework for integration with other sensors for monitoring plant growth.

The solution presented in this paper required a minimal amount of labor during run-time, and can be extended, by adapting established phenotyping and plant-breeding to the techniques, to track plants that are growing in multiple growth environments concurrently. This solution involved a semi-autonomous rover equipped with a Universal Robots UR 10 (UR10) robot arm, an industrial computer, a Microsoft Kinect (v2) sensor, and a rover base. The system was self-powered and required no wires to the outside world. The rover was equipped with a 120 volt power supply capable of powering numerous auxiliary sensors. This system allowed for attachment of plant monitoring equipment, as illustrated in three proof-of-concept experiments. In the first, the rover autonomously moved to the region representing the desired growth chamber in a setup mimicking Iowa State University's new Enviratron plant growth facility. This facility will contain several growth chambers with a robot vestibule in front of each chamber door. Once at the destination, the system probed several plants in the chamber with a rigidly-mounted steel rod that simulates other sensors that need to be placed at a certain distance and with a specific orientation to plant leaves. In the second experiment, the robot probed leaves on one plant using three related but different data sources, to compare the

accuracy of each source. The third experiment involved the robot probing (perpendicularly) a flat surface which was at several orientations.

2.3. Materials

2.3.1. Background on hardware and design

The Enviratron Rover aims to be a tool which can be used to autonomously gather feedback on plants which are simultaneously growing in multiple environments, from sensors as varied as traditional cameras, Raman spectrometers, thermal imagers, and fluorescence monitoring systems. The final application consists of eight growth chambers, arranged in a grid pattern (Figure 1). To reach these goals, the rover needs to be mobile, accurate at positioning sensors, and able to navigate the environment it is placed in (roughly 120 cm in the direction perpendicular to travel), all with extremely high reliability and repeatability. Making the rover autonomous will allow data to be collected at precise intervals of time, with significantly lowered per-data labor requirements for the researchers involved. The system should run without human input for a whole day, or 8 hours. Clearly, the application requires the system to adapt to varied sensors, and a powerful onboard PC to process that information.

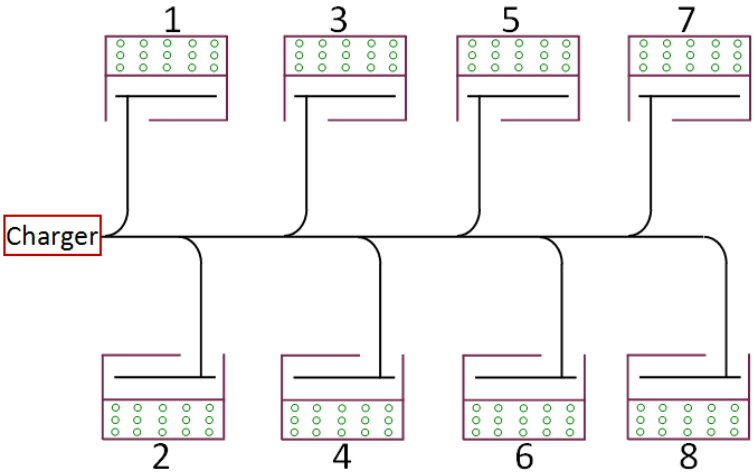

Figure 1. The chamber arrangement.

The scope of this paper is to present the system in an intermediate stage where automation is achieved for each individual task. Our setup is equivalent to assuming the growth chamber vestibule is opened, and the material separating the vestibule from the growth area is already removed. Human input was only required to select, using a PC, a point on the plant to image, and the human told the rover when to shift sideways to view another plant. The procedures required to get Raman spectrometer readings share very similar sub-tasks, such as locating the plant and calculation of the surface normal.

Several commercial systems could be repurposed and programmed for this task. However, each has significant drawbacks. For instance, the Segway 440 Flex Omni has omnidirectional control, can interface with external hardware, and is specified to carry payloads as high as 450 kg. Another system, the Neobotix MMO-550, is omnidirectional and has support for the UR10, sensors and sensors such as SICK NAV350, and can be controlled using the open-sourced Robotic Operating System (ROS). However, its uptime is quoted at 3 hours and there is limited additional space for sensors and electronics. Finally, the Ridgeback, sold by Clearpath Robotics, nearly fits the design requirements, but has low obstacle clearance, mediocre battery, and limited support for high-power-consumption external computer, arm and sensor configurations.

It should be noted that, to the best of our knowledge, no use of a rover for autonomous imaging of plants in multiple environments has been reported in the literature. These previously-mentioned commercial systems are only collections of multi-purpose hardware which we merely claim could be repurposed, i.e. modified and put to use, for the current novel task.

2.3.2. System hardware

The rover proposed in this paper, shown in Figure 2, combines many features which are desirable in research applications similar to this, and a brief overview of these features will be explained in this section.

Figure 2. The Enviratron rover.

The main frame of the rover uses the T-slotted aluminum building system sold by 80/20 Inc. (Columbia City, Indiana, USA). This makes the overall system architecture extremely modular and adaptive, and also provides natural ease of maintenance. The system was modeled entirely in SolidWorks prior to build. There was one Roboteq FBL2360 motor controller for each side (Left and Right) each controlling two Midwest Motion Products MMP BL58-412F-48V GRA60-032 brushless DC motors. Each motor drove one 6” Mecanum Wheel HD, purchased through AndyMark. Four dust and water-resistant Roboteq MG1600 magnetic tape sensors were used for guidance and simple localization. A Spektrum DX6i Transmitter was used for remote control during development. One lighted hard-wired emergency stop was placed on each side for safety. A Meanwell TS1000 DC/AC inverter powered external 120 V

outlets as well as the three core components of the imaging and probing system: a Logic Supply ML400G-30 industrial computer, Microsoft Kinect for Windows V2, and Universal Robots UR 10 robot arm. Finally, the whole system is powered by a 100Ah battery from AA Portable Power Corp (also known as batteryspace.com). This battery was chosen for its high energy storage capacity and appropriate balance of features, safety, and pricing. The overall system, excluding the UR10 arm to allow comparison with previously-mentioned commercial systems, had roughly half the cost of the least-expensive commercial system.

The Mecanum wheels used are similar to those analyzed by Gfrerrer (2008). In the ideal Mecanum wheel, force between the wheel and the ground only occurs along a vector parallel to the axis of the single roller which is in contact with the ground at that instant. Each roller's axis is rotated 45 degrees from the motor's axis. The wheels make contact with the ground in a shape described as an "O", a configuration often termed "O from below" in the robotics community. If the wheels are installed such that this does not occur, the stability was found to be poor, especially for lateral motion.

2.4. Method

2.4.1. Overview

Our task mainly involves: localization of the rover, control of the motion of the rover, building knowledge of the growth chamber's contents, and positioning the sensor. The way we integrated our hardware allowed compartmentalization of these tasks, all of which can be controlled by the "brain", which is the PC (Figure 3).

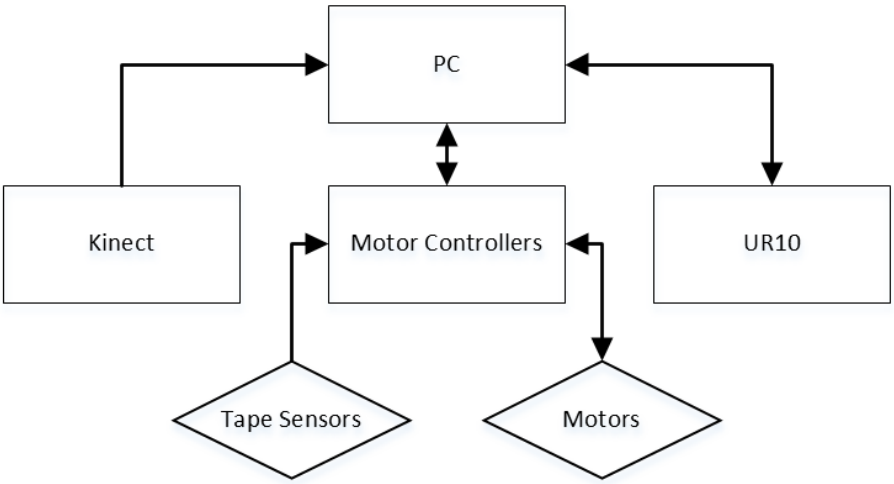

Figure 3. Software hierarchy. Line denotes a communication line, and arrows show direction packets are sent.

Various communication protocols and technologies were leveraged. Three tape-sensor data streams were sent to the motor controllers using Pulse Width Modulation (PWM), and one magnetic sensor used Roboteq's proprietary RoboMag protocol, allowing marker information to be transmitted easily. Motor position was implicitly sent to the motor controller via the built-in Hall-effect sensor. Roboteq's proprietary implementation of a PID closed-loop speed controller was leveraged by our custom "master" motor controller script to control each motor during each scan cycle. The two motor controllers ("master" and "slave") communicated using Roboteq's proprietary implementation of the Controller Area Network (CAN) protocol. Our custom PC code was all written in C++. The PC modifies and reads the motor controllers' internal variables using serial (RS232) communication. The PC gets current pose (position and orientation) information and sends desired poses to the UR10 via TCP/IP (Ethernet) communication. Kinect information is read using a USB 3.0 connection.

2.4.2. A brief note on control strategy

This paper aims to present a functional, intuitive, and adaptable framework for mobile manipulator implementation. As such, the control scheme is kept simple with a minimal

amount of tunable parameters. The ideas presented apply to all holonomic robots equipped with a guidance sensor on each side. Although it is dynamically-loaded, the system requires the controller to know relatively little about its system in order to function.

All inputs and real-time commands are fed into the “master” motor controller which then commands the other “slave” motor controller. The motor controllers’ internal scripting have a large number of internal functions, but can only process Booleans and integers and have no ability to add libraries. This is generally not a limitation if clever coding practices are used. To provide the system with knowledge of its location, we have implemented a concept equivalent to tracking the rising edge of strategically-placed markers (any “upside-down” magnetic tape is interpreted as a marker by the MG1600’s). A marker on one side of the track makes the controller “look” for rising edges on the other side of the track. The number of rising edges found during one “look” interval is stored as the last known location in the motor controller. To determine when the rover should turn off of the main track, we checked whether we counted an even or odd number of markers and compared with the current desired chamber. An example of this logic, for an odd chamber number, is presented in Figure 4. In addition to information on the most recent marker count, representing the current magnetic-tape branch, the PC can recognize the four-way intersection inside each individual chamber that corresponds to the leftmost plant. This provides requisite low-level localization. Further localization within the chamber will result from feedback from the Kinect sensor.

```
MarkerCopy = Marker
While MarkerCopy > 2
        MarkerCopy = MarkerCopy – 2
If MarkerCopy = 1 & Marker = DesiredChamber
        Follow Left Track
Else
        Follow Right Track
```

Figure 4. Psuedocode for track choice.

The rover has two modes for guidance: one-sensor proportional control and two-sensor proportional control. The strategy can be summarized with the following formulas.

1-sensor control:

$$V_{DOT} = V_{DEF} * F \tag{1}$$

$$F = \begin{cases} 0.5 \; if \; on \; turns \\ 0.75 \; if \; left \; tape \; detected \; (i.e. "look") \\ 1 \; else \end{cases} \tag{2}$$

$$V_{STE} = K_1 * T_1 \tag{3}$$

2-sensor control:

$$V_{DOT} = V_{DEF} * F \tag{4}$$

$$F = \begin{cases} 0.5 \; if \; on \; turns \\ 0.75 \; if \; left \; tape \; detected \; (i.e. "look") \\ 1 \; else \end{cases} \tag{5}$$

$$V_{STE} = K_2 * (T_1 + T_2) \tag{6}$$

$$V_{PERP} = K_2 * (-T_1 + T_2) \tag{7}$$

In the formulas, $V_{DEF}$is the default velocity, V_{DOT}, V_{STE}, V_{PERP} terms denote velocity in the direction of travel (DOT), steering, and perpendicular to the DOT, respectively. F is a

scaling factor, K_1 and K_2 are proportional gains, T_1 is the reading from the tape sensor on the leading edge of the robot (i.e. the DOT), and T_2 is for the trailing edge. Equations 1-3 are standard proportional control. Equations 4-7 apply proportional control in two directions. Equation 7 makes the robot correct the difference in the Tape readings by moving perpendicular to DOT; if the robot is moved forward both sensors will read opposite signs and strengthen the feedback. As seen in equation 6, an improperly-oriented robot, i.e. one that is not "pointing along the track", will have both sensors have the same sign and the steer command will be strengthened. The three velocity terms are summed appropriately for each wheel (c.f. AndyMark), and the results are sent to the individual motors.

2.4.3. Pre-Processing of Kinect depth data

The imaging and probing system has to be inherently robust to varying leaf size, stalk height, and plant type. Two approaches have been considered: 1) the trivial case of hard-coding robot-arm positions to several generally-desired poses, such as "front view" and "top view". 2) The general case of calculating desired pose based on knowledge of the sensor being used and the current arrangement of plants. For this study, we focused on the second, more general case.

Researchers have proposed using many 3D reconstruction methods on plants including structured light (Nguyen, Slaughter, Max, Maloof, & Sinha, 2015), ToF (Alenyá et al., 2013), and stereo reconstruction (Biskup, Scharr, Schurr, & Rascher, 2007). The Kinect V2 sensor was chosen for this study due to its affordable price, useful features and specifications (c.f., Butkiewicz, 2014), and well-documented application program interface (API). An additional advantage of the Kinect V2 is that its Software Development Kit (SDK) provides reasonably

accurate reconstruction sample code, termed KinectFusion, that integrates easily into custom applications (Izadi, Kim, Hilliges, Molynequx, Newcombe, Kohli, 2011, and Microsoft, 2016).

Sending an unfiltered depth image to the KinectFusion algorithm was found to lead to a gradual erosion of the leaves, stalk, and stems of the plant, resulting in unusable meshes. However, this off-the-shelf algorithm was found to be very effective if an appropriately-filtered depth image was instead passed to the algorithm. The number of voxels that are tracked are limited, and unstable voxels are filtered out, to allow the algorithm to run in real-time. In our application the table and walls surrounding the plant are far more stable than the pixels corresponding to the thin-stemmed, thin-leaved plants.

Several details were found to be important for this application. First, algorithms for real-time tasks such as plant probing need to be computationally efficient enough to allow quick reconstructions. Additionally, the Kinect V2 API has an accurate mapping between the color camera and the depth camera. However, not every depth pixel has a corresponding color pixel due to physical properties of the sensors. We set depth pixels without an RGB counterpart to zero. Incoming images from the Kinect sensor were put into a data structure that is effectively an RGB image. This representation is known to be sensitive to lighting changes, and there are methods for dealing with this issue. Luckily, the Kinect V2 had quite effective auto-brightness capabilities so lighting changes were not a significant issue. The challenge was finding an appropriate color space that allowed for easy identification of plant matter. Yang, Lu, and Waibel (1997), for instance, found that human faces were clustered in what they term chromatic color space. We are mainly interested in plants, which are generally diffuse green and brown objects, so we converted each RGB input image into HSV. After conversion to HSV, an experimentally-determined threshold was applied to each depth pixel. Let D_{ij} (T_{ij})

represent the pixel in row i and column j of the R x C depth image (thresholded depth image), corresponding to a region of the HSV image as determined by the Kinect API's mapping, and let V_{ij} be the corresponding pixel in the V image of the HSV color space. We have, for $1 \leq i \leq R \; and \; 1 \leq j \leq C$,

$$T_{ij} = \begin{cases} D_{ij} \; if \; V_{ij} < 140 \\ 0 \; else \end{cases} \quad (8)$$

The value of 140 conservatively thresholds out many extremely bright pixels, such as the walls of the growth chamber. Next, a 2x2 rectangular structuring element was used to morphologically dilate the image (Jain, Kasturi, & Schunck, 1995), conservatively removing many noisy "bright" elements. The resulting thresholded and dilated depth image, which contains the plant pixels plus some other pixels, was passed to the KinectFusion algorithm. Most settings were found to have minimal effect on the results. We left every setting default except increased the amount of voxels tracked per meter from 256 to 512. An example fusion image is shown in Figure 5.

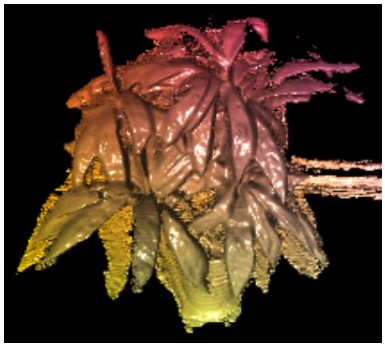

Figure 5. A sample fused image

2.4.4. Describing Kinect data in robot-base coordinates

Before being able to use the Kinect's 3D information to accurately position a robot arm, the Kinect data must be described in robot base coordinates (see Figure 6). This was achieved by first describing camera coordinates P_c in end-effector coordinates and then

describing that data in robot base coordinates, via homogeneous coordinate transforms. To convert user-selected point P_c [m], from camera coordinates into base coordinates P_B [m] we used:

$$P_B = H_{EB} * H_{CE} * P_C \quad (9)$$

Where H_{EB} and H_{CE} denote transformation matrices between end-effector and base, and camera and end-effector, respectively. A transformation matrix is given by

$$H = \begin{bmatrix} R & P \\ \mathbf{0} & 1 \end{bmatrix} \quad (10)$$

where $\mathbf{0} = [0, 0, 0]$, $R \in \mathbb{R}^{3x3}$ is a rotation matrix, and $P \in \mathbb{R}^3$ is the "translation vector describing the origin of the original coordinates by using the final coordinate system.

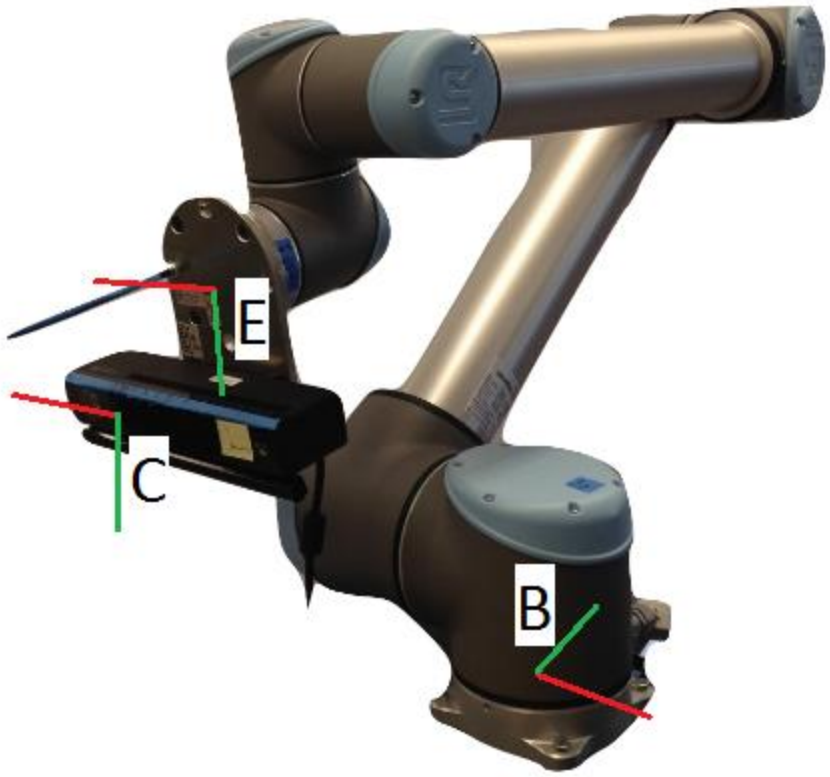

Figure 6. Main UR coordinates. B: Robot base, E: end-effector, C: camera coordinates.

The mapping between the camera's coordinate system and the robot arm's end-effector's coordinate system must be determined. An overview of camera "hand-eye"

calibration can be found in the seminal Tsai and Lenz paper (1989). This mapping can be completely defined by a rotation followed by a translation, which can be represented by a homogeneous transformation:

$$P_E = H_{CE} * P_C \tag{11}$$

where P_E denotes a point in robot end-effector coordinates [m].

The Kinect SDK presented a few details that needed to be worked out. For instance, for a given point in the space, different functions return coordinates in different reference frames. Careful reading of the SDK's various projects' source code and experimentation with the sensor yielded a thorough understanding of the setup.

Transforming from a given frame to the robot base follows the same process of applying successive transformation matrices to coordinates until the point is described in robot coordinates. Since all three coordinate systems used by Kinect shared the same origin, converting form these coordinates to the camera coordinates was trivial. The main transformation matrix was from camera coordinates to the robot end-effector. First, a rough "theoretical" estimate of H_{CE} was obtained using calipers, for validating the final calibration.

In absence of the detailed calibration found in high-priced sensors, a more accurate calibration matrix was found using Christian Wengert's add-on (Wengert) to the Camera Calibration Toolbox (Bouguet, 2015) for MATLAB. The rotation was found to be small (off-diagonal entries $|h_{ij}| < 0.03$). Since the camera gets bumped occasionally and will therefore change orientation by small amounts that will likely negate these rotation entries, we mostly care about translation. The HCE for our setup was approximately (translation in [mm]):

$$H_{CE} = \begin{bmatrix} 1 & 0 & 0 & 54.0 \\ 0 & 1 & 0 & 102.6 \\ 0 & 0 & 1 & 82.5 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} \tag{12}$$

The final calibration procedure involved techniques found in Wengert's online documentation, with a few modifications. The Kinect V2 produced washed-up images of the calibration pattern when the camera was within roughly 0.5 meters of the calibration pattern. A single layer of standard white paper was taped over the Kinect IR emitters, with a piece of tape securing the edge closest to the receiver firmly against the Kinect front face. This decreased the IR emission in a clean enough fashion to collect images that weren't over-saturated. The calibration pattern was scaled up such that the spacing between two adjacent circles' centers was 16mm. Finally, the Kinect was flipping images left to right internally; we un-did that flip prior to calibration. The system was successfully calibrated using between 30 and 40 images; several calibrations were performed to perfect the process. Doing rapid calibration enables the end system to be modified to accommodate additional sensors without delaying the rest of the experiments' schedule significantly.

After the points are described in end-effector coordinates, we desire to describe them in robot base coordinates. Given a pose of the robot, defined by position $P = \left[p_x, p_y, p_z\right]^T$ and rotation vector $r = [r_x, r_y, r_z]$ in the robot base coordinates, we desire the transformation matrix from the end effector to the base. Using the notation in equation 10, the rotation matrix R is defined as

$$R = \left[A_x, A_y, A_z\right] = \begin{bmatrix} \mathrm{c} + \mathrm{r_x^2 v} & \mathrm{r_x r_y v} - \mathrm{r_z s} & \mathrm{r_x r_z v} + \mathrm{r_y s} \\ \mathrm{r_y r_x v} + \mathrm{r_z s} & \mathrm{c} + \mathrm{r_y^2 v} & \mathrm{r_y r_z v} - \mathrm{r_x\, s} \\ \mathrm{r_z r_x v} - \mathrm{r_y s} & \mathrm{r_z r_y v} + \mathrm{r_x s} & \mathrm{c} + \mathrm{r_z^2\, v} \end{bmatrix} \tag{13}$$

where $c = cos(\theta)$, $s = sin(\theta)$ and $v = 1 - cos(\theta)$. See Craig (2009) for discussion.

In the experiment presented in this paper, described later, all coordinates were transformed to "camera" coordinates immediately after their variable's initialization. The pose used for H_{EB} depended on where the data was obtained from, as explained later.

2.4.5. Obtaining position and normal

To aid in planning for the next phase of the project, three different methods of probing were implemented. All three used the same Kinect camera and the same probing algorithm. However, the source of the normal and position information was different.

The Kinect for Windows SDK (K4W) has functions for mapping between the color frame (RGB) to the camera space ("K2"). Using OpenCV, K4W, and custom methods (functions), the first method involved using the Kinect's internally-estimated, real-time position and normal of the location in the internal KinectFusion reconstruction corresponding to an RGB point clicked on our OpenCV window (termed method F). The pose of the robot at the initialization of that instance of KinectFusion was used to calculate H_{EB}, since data is returned after being transformed to initial-pose "camera coordinates" using Kinect's estimate of the current pose relative to the initial pose.

The second method used three points in the Kinect's real-time depth space corresponding to RGB points (termed method D). The point clicked on our OpenCV window gave us the desired position. The surface normal was obtained using the K4W CameraSpace coordinates associated with the clicked pixel (P_1), a pixel five pixels to the right (P_2; since the RGB images are flipped this is equivalent to -x in camera coordinates), and a pixel five pixels below (P_3) in the RGB image. The pose used to calculate H_{EB} was the pose of the robot at the instant the probing button was clicked. This data is real-time, and not dependent on the estimated relative pose. The surface normal pointing out of the leaf was the cross product of the vector from P_1 to P_2 and the vector from P_1 to P_3. Quantitatively: $V_1 = P_2 - P_1, V_2 = P_3 - P_1, and\ N = V_1 \times V_2$. V_1 and V_2 are vectors.

The third method involved manually opening the mesh generated by KinectFusion with the software MeshLab and saving the coordinates of three points (minimum necessary to understand the leaf's normal and position), to .txt files (termed method M). Logic used to find the normal in this method was the same as in the method D above. This data uses the same H_{EB} as method F.

For methods F and D, the user simply clicked the desired point and clicked probe. In method M, there was roughly a minute of required interaction between the user and the PC, per probe.

2.4.6. Calculating desired robot pose

Next, we calculated the desired end-effector coordinate system axes $A_x, A_y, and\ A_z$. Two logical constraints were added to our system: A_z should align with the surface normal N and the Kinect should be level with the ground. Since our probe end-effector is orthogonal to our end-effector's XY plane, i.e., it extends in the A_z direction, we solved the following equations for $A_x, A_y, and\ A_z$:

$$A_z = -N, A_x * N = 0, A_{xz} = 0, and\ A_y = A_z \times A_x \quad (14\ a, b, c, d)$$

This can be solved by two directions of A_x. To disambiguate, two checks were implemented. First, if A_y is pointing up and the normal is not pointing high, rotate 180 degrees so A_y is downward. Else, if A_x points left, rotate 180 degrees so the Kinect doesn't hit the plant.

The position the end-effector moved to, P_{C1}, is a translation from the actual leaf position P_1. Our probe stick was offset from the tool center point in the direction of the tool's x-axis, and was orthogonal to the tool's XY plane. Thus, the end-effector position, in robot base coordinates, was defined by:

$$P = P_1 + L * N - r * \mathrm{A_X} \quad (15)$$

Where L is the length of the probe stick and r is the radius from the tool center point to the thin probe. Next, we determined the rotation vector to send to the arm by solving equation 13 for $r_x, r_y, and\ r_z$. Finally, the calculated coordinate values were sent to the UR10 to probe the leaf.

2.5. Results and Discussion

This section presents three experiments to validate the rover's design and the imaging system. The first uses the rover to position the UR10 appropriately, and uses method M to probe two plants. The second experiment uses a stationary rover to position the UR10 appropriately, and uses all three described probing methods to probe one plant. For all probing's, the goal location was the bottom-left corner, from the perspective of the crouched robot, of a small piece of painters' tape, roughly 5mm square. For easy identification, one leaf had two additional pieces placed away from the goal location, and one leaf had one additional piece. The third experiment involved probing a piece of flat acrylic which was oriented at specific angles, to gain insight into the KinectFusion's surface normal estimation. An experiment testing the KinectFusion normal or position accuracy has not been reported in the literature, and this should prove to be valuable to researchers looking to use the KinectFusion in other systems. As far as the author is aware, only depth information has been quantifiably analyzed (Butkiewicz, 2014).

2.5.1. Mobile rover

This section presents an experiment demonstrating current effectiveness of the proposed system. The UR 10 arm was commanded to probe two plants (one artificial Silk

Dracaena plant - VCK8023, from artificialplantsandtrees.com - and one real Ficus plant), on three separate leaves, five times each (i.e. 30 separate probing's). Both plants were placed on a 73 cm-high table (approximately the height that the UR base is at), around 1m apart. During normal conditions, the UR arm would extend approximately 80 centimeters in its -Y direction for its end-effector to hit a desired leaf.

Before each trial, the rover was set up at the edge of a track with complete magnetic tape as would be in a setup with one chamber. The rover entered the "chamber" and stopped at the intersection. A user clicked a button on the PC's custom user interface (UI), commanding the rover to shift sideways. There was only one issue with the tape-following navigation. After trial 3, one sensor reported magnetic tape in an un-taped portion of concrete. The sensors were calibrated with the Roboteq utility and the experiment proceeded as planned. The tape-following was accurate enough at our low speed that it has not been formally tested; the steady-state oscillations about a straight line are negligible, and the rover never went off the tape.

Once the desired plant was in view, the user initiated our filtered KinectFusion algorithm. After several seconds of data acquisition, the mesh was saved. Three points near the bottom-left corner of the tape markers were located in the MeshLab software, and stored in .txt file. The user clicked on our UI to probe the plant, the coordinates were sent through our "mesh probe algorithm", and the UR 10 approached the leaf with the probe. Using the UR 10's touchscreen user interface, we found ground truth by translating the end-effector until the probe hit the bottom-left corner of the tape markers. A summary of results is presented in Table 1. The mean error was 26.5 [mm] (95% CI [24.2 28.7]).

Table 1. Euclidian distance error [mm]. “Avg.” means “average”.

		Trial					
Plant	Leaf	1	2	3	4	5	Avg.
1	1	29.7	27.7	27.4	27.9	26.6	27.9
1	2	22.9	23.5	22.0	24.5	26.9	24.0
1	3	20.7	33.9	20.9	25.7	30.3	26.3
2	1	18.6	16.9	8.5	24.0	19.6	17.5
2	2	27.4	34.3	29.5	30.3	32.6	30.8
2	3	34.7	33.0	27.1	35.1	31.9	32.4
						Avg.	26.5

2.5.2. Stationary rover, positional accuracy test

For the second experiment, the rover was kept stationary near the artificial Silk Dracaena, the UR10 was placed in a low “crouching” pose, and the program was initialized. The UR arm slowly (0.07 m/s) moved in an arc up to a higher pose, looking at the plant. The mesh was saved and integration was paused to ensure as much similarity between method F and M. Method F was used, the ending coordinates recorded, and the ground truth was found as before. The UR resumed its high pose, and if tracking was still successful, method F was used to probe another leaf. Again, the UR resumed its high pose, and if tracking was still successful, method F was used to probe another leaf. Tracking was occasionally lost while moving close to the plant, likely due to the proximity to the plant, which would result in an entire depth image of points below the Kinect’s minimum threshold. When tracking wasn’t successful on the second or third leaf, the Fusion was started at the bottom again before the next probing. (Since the KinectFusion can be reconstructed in real time, losing tracking is only a minor issue. For applications where tracking continuously is important, a future work could be replacing the transformation matrix with one obtained using feedback from the robot arm, and stopping reconstruction while approaching the plant.) However, a new mesh was not saved. Once all three leaves and their ground truth were found for method F, method P and M were used to probe the same three leaves. So, each trial had 9 probing’s and 9 corresponding

ground truths. (Since orientation wasn't exactly the same for each probing, the probe would contact the tape at a different end-effector position. Thus, ground truth could not be shared between methods or trials.) Five trials were conducted, resulting in 15 samples for each of the three methods. A standard comparison of means (student's t and LSD) test was performed in JMP Pro 12. Key results are presented below.

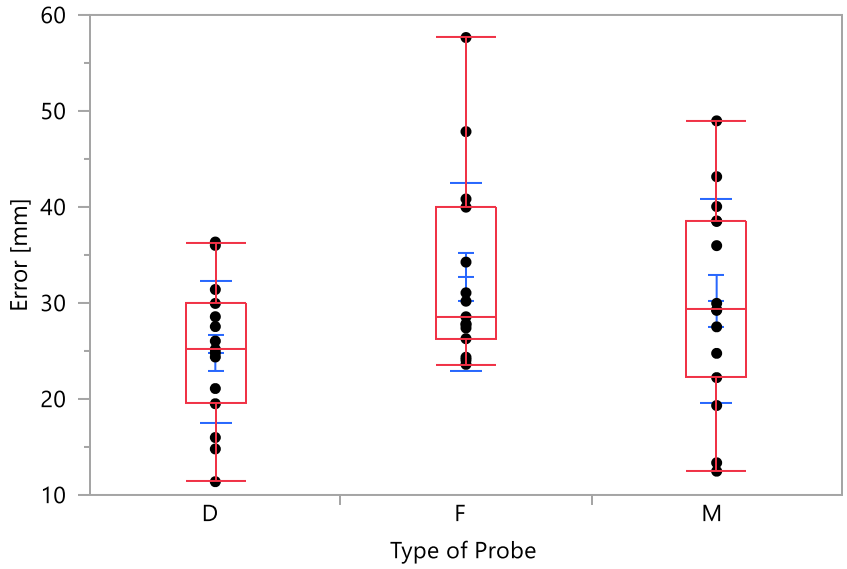

Figure 7. Error (Euclidian distance) [mm] for methods D, F, and M on a stationary rover.

Table 2. Means and Standard Deviations

Level	Mean [mm]	Std Dev [mm]	Std Err Mean [mm]	Lower 95% [mm]	Upper 95% [mm]
D	25	7.3	1.9	20.8	28.9
F	33	9.8	2.5	27.3	38.2
M	30	10.7	2.8	24.3	36.2

Table 3. Student's t Comparison of Means

Level	- Level	Difference [mm]	Std Err Dif [mm]	Lower CL [mm]	Upper CL [mm]	p-Value [mm]
F	D	8	3.4	1.0	14.8	0.0260*
M	D	5	3.4	-1.6	12.3	0.1257
F	M	3	3.4	-4.4	9.5	0.4606

As seen in Table 2 and Table 3, only one difference of means was statistically significant (at $p<0.05$). Method D only had a statistically significantly lower mean error ($p<0.05$) mean than method F. However, the Lower CL is 1, suggesting that the methods,

although having statistically different means, could have *practically* similar means. This is in line with what was qualitatively observed; method D appeared to be the best method, but not by a lot. This suggests that the default Kinect Fusion pose estimation, used by method F (real-time KinectFusion) but not method D (real-time Depth) is reasonably accurate but not perfect.

More data could be collected to increase confidence in the true means. However, this will likely be unnecessary because in practical terms, they all are good enough for "rough" probing, and none of them were accurate enough to be used, un-corrected, with a traditional fluorimeter or Raman spectrometer sensor. Regardless of the plant leaf location data source, the probing accuracy still needs to be improved for the system to meet the autonomous data-collection objective. This will be elaborated in the "general discussion and conclusion" section.

2.5.3. Stationary rover, surface normal test

This experiment tested the accuracy of the surface normal estimates that can be obtained using KinectFusion. A standard white paper was fixed to the surface of a piece of standard grade extruded acrylic (sold by Grainger, Inc.), as shown in Figure 8. The acrylic was fixed to pieces of 8020 extruded aluminum arranged with hinges for one degree of freedom. This hinge lined up with the robot's x axis. A piece of paper with lines intersecting at a common point in angles in increments of 30 degrees was placed below the setup. These intersections helped measure rotation about the robot's z axis. For several combinations of x and z rotation ([0,0], [-30,0], [-60,0], [-90,0], [-90,30] with [0, 0] meaning flat acrylic and x axes aligned), all three previously-described methods (real-time Kinect Fusion, depth, and mesh) were used to probe the bottom-left corner of a small piece of blue painters' tape, ten times per combination. The range of angles tested is limited mainly due to self-collisions of the robot and collisions of the Kinect with the Environment.

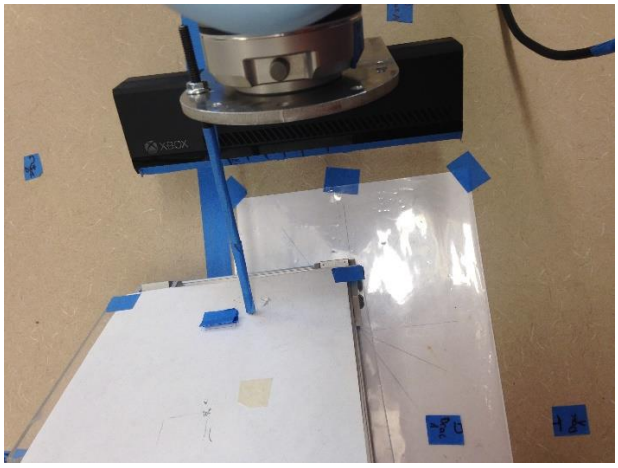

Figure 8. Probing angled white paper.

As the surface normal defines one vector, and the probe defines another vector with a common point where the probe intersects the paper, these two vectors define a plane. Within this plane, the two vectors will make an angle. This angle is easy to standardize and is a suitable metric for comparing various systems' ability to interact with the surface normals in their environment.

In a perfect trial, the probe would just barely contact the paper, at an exact point. As shown in the previous experiments, there is translational error in the system. To deal with this error, the acrylic was translated along the table's angle lines when the probe appeared to be near the acrylic. This maintained the angle of the acrylic's normal relative to the robot's coordinate system, preserving the ground truth for measurement. Note that a perfect probing would yield an angle of 90, meaning the probe was perfectly normal.

Results are presented in Table 4 and Table 5. First, a comparison of the means of the three methods was conducted. Method M was seen to have statistically significantly better mean compared with the other two methods, while the depth (D) and KinectFusion (F) methods failed to show statistically significant different means. This is in agreement with what was observed and what was expected. An estimate of the surface normal is far more accurate when

using neighboring data to find an average normal estimate. Looking at each method individually, the results show that method F (real-time KinectFusion) has a mean between 63 and 67 degrees (95% confidence). Method D (depth) shows a mean between 65 and71 degrees (95% confidence). Neither of these ranges is accurate enough to use fluorimeters, which are sensitive enough that they are often are used with an attachment to guide the sensor to the proper angle. The error for method D can be attributed to the local noise in the Kinect data and possibly the hand-eye calibration, as the KinectFusion pose estimation was not used. Error in method M (manually-selected KinectFusion), with the mean angle between 81 and 84 degrees (95% confidence) can be attributed to the KinectFusion's estimation of pose and possibly the hand-eye calibration. Since a human entered points manually, and used points that were separated by roughly a centimeter, there was little room for other errors. Error in method F is more complicated, however. This error will be a combination of the KinectFusion's estimation of pose as well as the local imperfections caused by local Kinect noise. Comparing the means for method F and D shows that the two methods' means are not statistically different. This reinforces what the previous probing experiment showed: the KinectFusion's estimation of pose is probably pretty accurate. This suggests that much of the error was from the Kinect's local surface normal estimate.

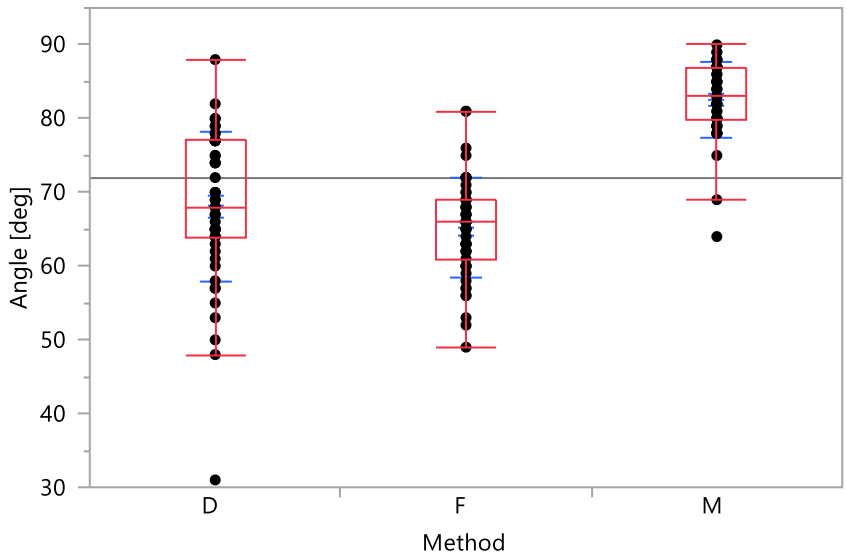

Figure 9. Angle [deg] for methods D, F, and M on a stationary rover. 90 is perfectly normal.

Table 4. Means and standard deviations [deg]. 50 trials each level (method).

Level	Mean [deg]	Std Dev [deg]	Std Err Mean [deg]	Lower 95% [deg]	Upper 95% [deg]
D	68	10.2	1.4	65.2	71.0
F	65	6.7	0.9	63.3	67.1
M	83	5.0	0.7	81.1	84.0

Table 5. Student's t comparison of means [deg]

Level	- Level	Difference [deg]	Std Err Dif [deg]	Lower CL [deg]	Upper CL [deg]	p-Value [deg]
M	F	17	1.5	14.4	20.4	<.0001*
M	D	14	1.5	11.5	17.5	<.0001*
D	F	3	1.5	-0.1	5.9	0.0580

2.6. General Discussion and Conclusion

As presented in this paper, a novel system for autonomous indoor phenotyping was created. The system's design, hardware, software layout, and a concept-level view of the software were presented. The system was tested in a one-chamber case, and the effectiveness of the navigation system was informally demonstrated. Data is presented demonstrating the validity of the plant probing sub-system. This system used the Kinect V2 ToF sensor, which is rapidly being adopted by various research communities. The reliability of the Kinect's surface normal estimate and an investigation into the accuracy of the KinectFusion algorithm are

presented. This has previously been a gap in the literature, even though this application of the Kinect has many potential uses for integration with robotic systems such as that presented in this paper. The result from probing using the same algorithm on three related but different data structures were compared. One general conclusion is that the Kinect's internal estimate of orientation is generally accurate to better than 9 degrees. The three data sources discussed – real-time depth (D), real-time local Kinect Fusion (F), and offline Kinect Fusion (M) – can be used to consistently position sensors within 4 cm of a desired position, and if used properly, within 3 centimeters reliably. In ideal cases, the error was as low as 1.5 cm. Before installing other sensors, some refinement of the probing sub-system should be done, improving location accuracy and surface normal estimation.

Future work includes reducing probing position and surface normal errors. Efforts to this end include modifying the Kinect housing for as rigid as possible mounting, and doing a camera calibration with more pictures, and using optimized locations for these pictures. Other improvements include more sophisticated pre-processing of the depth data and using the UR10 feedback for calculation of H_{EB}, the transformation from end-effector coordinates to robot base. The surface normal estimate can be improved using, for instance, principal components analysis to find the surface normal (technically, an eigenvector) corresponding to the plane which is fitted to a cluster of points near the goal location. Use of another highly accurate distance sensor to augment the rough, real-time Kinect algorithm is another general solution. Additionally, we aim to program the system to autonomously position sensors near a desired leaf with a desired angle. This will likely involve using traditional RGB image processing methods to detect the desired leaf location, and incorporation of an appropriate collision-free path planner.

2.6.1. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1428148. The authors would like to thank Jingyao Gai and Rajesh Putta-Venkata for help during the early development of the probing algorithm, and undergraduate students Caleb Stafford, Taylor Wisgerhof, Layne Goertz, and Austin Plotz for help during the mechanical design and build phases.

2.6.2. Safety emphasis

In order to make the rover safe for interaction with humans, several features have been implemented. First, fuses were inserted which minimizes risk of short-circuiting of contacts and risk of drawing current that is higher than components' specifications. The battery consists of four sets of four 3.2.V (nominal) lithium-ion batteries (total 51.2 nominal). One fuse is inserted in between the 2 middle cell-sets, and one fuse is connected directly to the battery's 48V contact. Both fuses are currently 25A. Secondly, one emergency stop, which cuts power to the motors, is on each side of the rover base. Third, the wheels are designed to have a "moderate" level of traction, so the rover is physically limited in its accelerations and in its ability to push objects. This reduces likelihood that the rover would trap someone against a wall, whether the robot or the human was at fault. Fourth, according to the UR10 user manual (2015), the robot arm is suitable for "Collaborative operation according to ISO 10218-1:2011" If the robot exceeds a safety setting (force, torque, velocity, etc.) the UR10 can go into a slow or stopped mode according to the user-defined settings.

Further internal electrical precautions were taken. High-current relays (arbitrarily chosen as 200A) implement the emergency-stop and main switch power control, allowing reliable operation. Some electrical precautions similar to those recommended in the motor

controller manufacturer's user manual (Roboteq) were implemented. These included a path from motor voltage terminal through a pull-down resistor to ground while the system is turned off (implemented using a five-pole single-throw industrial switch), and a path from the motor voltage terminal through a diode and a 10A fuse to the battery's positive terminal to prevent over-voltage while the system is turned on. Finally, all significant wiring inside the controller has been sized appropriately, labelled, shielded with heat-shrinkable tubing and cable wrap, crimped with appropriately-rated wire crimps, and guided using cable ties and mounting pads. Bus bars were used to organize voltage contacts of the same voltage (24V, 48V, ground). This enhances maintenance and decreases the risk of electrical accidents. No unified standard was applied or monitored, however.

References

Aksoy, E. E., Abramov, A., Wörgötter, F., Scharr, H., Fischbach, A., & Dellen, B. (2015). Modeling leaf growth of rosette plants using infrared stereo image sequences. *Computers and Electronics in Agriculture*, 110, 78-90.

Alenyà, G., Dellen, B., Foix, S., & Torras, C. (2013). Robotized plant probing: leaf segmentation utilizing time-of-flight data. *Robotics & Automation Magazine, IEEE*, 20(3), 50-59.

AndyMark. MecanumWheelSpecSheet. Retrieved from: http://files.andymark.com/MecanumWheelSpecSheet.pdf on 8-2-2016.

Azzari, G., Goulden, M. L., & Rusu, R. B. (2013). Rapid characterization of vegetation structure with a Microsoft Kinect sensor. *Sensors* (Basel, Switzerland), 13(2), 2384.

Biskup, B., Scharr, H., Schurr, U., & Rascher, U. (2007). A stereo imaging system for measuring structural parameters of plant canopies. *Plant, Cell & Environment*, 30(10), 1299-1308.

Bouguet, J.-Y. (2015). Camera calibration toolbox for MATLAB [computer software]. Retrieved from http://www.vision.caltech.edu/bouguetj/calib_doc/ on 4-6-2016.

Butkiewicz, T. "Low-cost coastal mapping using Kinect v2 time-of-flight cameras," *2014 Oceans - St. John's, St. John's, NL*, 2014, pp. 1-9.

Chaerle, L., & Van Der Straeten, D. (2001). Seeing is believing: imaging techniques to monitor plant health. *BBA - Gene Structure and Expression*, 1519(3), 153-166.

Chaivivatrakul, S., Tang, L., Dailey, M. N., & Nakarmi, A. D. (2014). Automatic morphological trait characterization for corn plants via 3D holographic reconstruction. *Computers and Electronics in Agriculture*, 109, 109-123.

Craig, J. J. (2005). *Introduction to robotics: mechanics and control / John J. Craig (3rd ed.)*. Upper Saddle River, N.J.: Upper Saddle River, N.J.: Pearson Education.

Gfrerrer, A. (2008). Geometry and kinematics of the Mecanum wheel. *Computer Aided Geometric Design*, 25(9), 784-791.

Izadi, S.; Kim, D.; Hilliges, O., Molynequx, D., Newcombe, R., Kohli, P. (2011). KinectFusion: real-rime 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*.

Jain, R., Kasturi, R., & Schunck, Brian G. (1995). Chapter 2: Binary Image Processing. In *Machine vision / Ramesh Jain, Rangachar Kasturi, Brian G. Schunck*. New York: McGraw-Hill.

Klodt, M., Herzog, K., Töpfer, R., & Cremers, D. (2015). Field phenotyping of grapevine growth using dense stereo reconstruction. *BMC bioinformatics*, 16, 143.

Microsoft, (2016). Kinect Fusion: Kinect for Windows 1.7, 1.8. Microsoft.com. Retrieved from https://msdn.microsoft.com/en-us/library/dn188670.aspx on 8-2-2016.

Montes, J. M., Melchinger, A. E., & Reif, J. C. (2007). Novel throughput phenotyping platforms in plant genetic studies. *Trends in Plant Science*, 12(10), 433-436.

Nguyen, T. T., Slaughter, D. C., Max, N., Maloof, J. N., & Sinha, N. (2015). Structured light-Based 3D reconstruction System for Plants. *Sensors* (Basel, Switzerland), 15(8), 18587.

Pieruschka, R., & Poorter, H. (2012). Phenotyping plants: genes, phenes and machines. *Functional Plant Biology*, 39(11), 813-820.

Rai, M., Pandey, S., Collette, L., Reeves, Timothy G, & Food Agriculture Organization of the United Nations. (2011). *Save and grow: A policymaker's guide to sustainable intensification of smallholder crop production / [final technical editing, Mangala Rai, Timothy Reeves and Shivaji Pandey; lead authors: Linda Collette ... [et al.]]*. Rome: Food and Agriculture Organization of the United Nations.

Robots, U. (2015). User Manual: UR10/CB3, Version 3.1. Odense, Denmark: Universal Robots

Royal Society of London. (2009). *Reaping the benefits: science and the sustainable intensification of global agriculture*. London: Royal Society.

Tsai, R., & Lenz, R. (1989). A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *Robotics and Automation, IEEE Transactions on*, 5(3), 345-358.

Wengert, C. calib_toolbox_addon [computer software] Retrieved from http://www2.vision.ee.ethz.ch/software/calibration_toolbox//calibration_toolbox.php on 4-6-2016.

Yang, J., Lu, W., and Waibel, A. (1997). Skin-color modeling and adaptation. Technical Report CMU-CS-97-146, CS department, CMU, 1997.

CHAPTER 3. PATH PLANNING OF A ROBOTIC ARM IN A GROWTH CHAMBER

3.1. Abstract

To aid in collection of phenotypic data, some researchers are proposing robotic systems. Operating a robotic arm in a growth chamber with multiple plants occupying space in the robot arm's used workspace can result in unwanted collisions. Current phenotyping systems ignore this aspect of robotic phenotyping. A novel implementation of the Probabilistic Roadmaps technique is applied to feedback from a Time-of-Flight sensor, the Kinect v2. A standard roadmap is created and further enhanced by adding nodes near the plants and near nodes with few neighbors. During the query phase, the algorithm attempts to add the goal to the roadmap. If unsuccessful, nodes are added near this goal. Collision-free motion is developed, considering the joints, end-effector, and Kinect sensor. Planning times are approximately 32 seconds, and query times are approximately 0.5 second. The approach is flexible and can easily be adapted to multiple setups with different sensors.

Keywords: Path Planning, Robot Arm, Growth Chambers

3.2. Introduction

In a world of changing climate and increasing world population, there is a great need to understand the interaction between genotype and phenotype in order to produce enough crop yield. Organizations such as the Royal Society of London (2009) and the FAO (Rai, Reeves, Pandey, Collette, & Food and Agriculture Organization of the United Nations, 2011) suggest a need for at least a 50% increase in food supply in the next half-century. Sustainable intensification, involving increasing the productivity of existing farmland while reducing

negative environmental impacts, is promoted as one of the best ways – and some would say the only way – to achieve this.

One of the main methods to increase crop yield without increasing chemical use involves plant breeding techniques. Effective plant breeding requires in-depth data on plants' health and growth patterns, which are part of their broader phenotype, or physical characteristics. Many traits relating to growth, performance, and yield are complex traits under polygenic control (Pieruschka & Poorter, 2012). Studying these traits using plants' phenotypes to understand how each strain behaves under various growing conditions is an important step toward improving the characteristics of a crop stock. This requires large quantities of data from sensors. One promising solution to obtain this data is to use robotic systems. However, robot arms are not programmed for collision-free motion, and the robotic arm may collide with a plant. Slight collisions are generally not an issue, but a robot arm may sweep a volume that goes straight through the middle of a plant, knocking it over, for instance.

This paper discusses available path planning solutions as they could be applied to operation in a plant growth environment, and proposes a novel solution which can be considered a variant of the probabilistic roadmaps (PRM) technique. Emphasis is on the application of a mobile manipulator interacting with plants in a growth chamber, but the method is general enough to work with many setups. Importantly, the framework provided allows easy re-configuration to adapt to different sensor arrangements.

The framework is tested using simulations and a physical plant setup. The framework uses input from a Kinect v2 Time-of-Flight (ToF) sensor. Both the simulations and the physical setup use a Universal Robots UR10 robot arm, and processing is done in MATLAB on a 64-bit Dell T1700 running Windows 10 with an Intel Xeon 3.4 GHz CPU and 16.0 GB RAM.

3.3. Related Work

Two related classic problems in robotics are path planning and collision avoidance. Path planning involves generating a set of poses which the robot should follow in succession. Collision avoidance describes methods which aim to help the robot prevent itself from hitting un-desired objects. This paper is aimed at collision-free path planning, a branch of robotics research which combines these two problems: these methods aim to provide sets of poses which the robot should follow, where the robot will not hit un-desired objects.

The collision-free path planning problem has been around for several decades, and is aimed around the general task of moving a robot arm from a starting position to a goal position while minimizing or eliminating collisions. Most approaches can be placed into one of two categories: approaches based on knowledge or simulation of the robot workspace and those based on local sensors (Shaffer, 1991).

Sensor methods include rings of infrared sensors (Gandhi & Cervera, 2003), capacitance-based sensors (Feddema & Novak, 1994), and a whole-arm skin of infrared proximity sensors (sensors described in Cheung and Lumelsky, 1992; successful implementation described in Lumelsky and Cheung, 1993). Sensor methods are often promoted for their relatively lower computational requirements, low sensor noise, and ability to operate in the absence of a perfect world model. However, many modern advances in computing power and algorithm maturity have led to successful implementation of real-time collision avoidance using knowledge of the robot workspace.

Some simulation-based approaches use robot simulators such as V-Rep (Rohmer, Singh, & Freese, 2013) to simulate the currently-running robot environment, and use the feedback to slow down robots as appropriate to lower risk of collision (Fenucci, Indri, &

Romanelli, 2014). In Fennucci's case, unfortunately, results have not been reported on dynamically-updated obstacles, although it appears possible given their framework. Another interesting real-time approach involves using a statically-placed, external Microsoft Kinect to compute distances between the robot arm, as modeled by a sequence of spheres, and obstacles of arbitrary shape (Flacco, Kröger, De Luca, & Khatib, 2012). These distances, and a rough measure of obstacle velocity, are sent to a collision-avoidance algorithm which computes target joint position and velocity. Finally, this target joint position and velocity is sent to the one of the Reflexxes libraries for jerk-limited, continuous control (consult, for instance, Kröger & Wahl, 2010 and Kröger, 2011). The approach has demonstrated the ability to avoid a human's arm in real-time. Another real-time collision-free robot control system limits the robot so that it avoids colliding with its environment except for a specific hand of a user, when the user enters the interaction mode (De Luca and Flacco, 2012). The observing PC uses the distance between the end-effector and the nearest obstacle to modify the task velocity. The smallest distance from an obstacle to other points of interest on the robot is used to slow down the joint motions which result in a smaller distance, as determined using the Jacobian associated with the robot's current joint positions. De Luca and Flacco point out that these vectors can be used in more traditional collision avoidance algorithms such as potential fields (Khatib 1986), circular fields (Haddadin, Belder, & Albu-Schaeffer, 2011; Singh, Stephanou, & Wen, 1996), and elastic strips (Brock & Khatib, 2002).

In general terms, the potential field method uses the goal position as an attractive force, and obstacles as a repulsive force, in operational space (the coordinates of the task i.e. position and orientation). This approach helps a robot arm avoid obstacles. As the method does not explicitly require global planning, avoiding getting stuck in local minima must be

accomplished by augmenting the method with other techniques. Circular fields generates a repulsive force perpendicular to the direction of travel. This conserves the kinetic energy of the system and generally helps the robot to move around obstacles, rather than getting stuck in local minima. Elastic strips represent a planned collision-free path as a swept volume in the workspace (XYZ, for instance) which is expanded by a Minkowski sum. When obstacles enter the vicinity of the original path, an external force is applied which creates a modified path which remains within the expanded swept volume. The approach allows robots with many degrees of freedom to operate in dynamic environments while maintaining task constraints.

One approach which has been shown to apply to holonomic and non-holonomic robots is the Rapidly-Exploring Random Trees (RRT) method (LaValle, 1998). The initial location is added to the graph, and subsequent nodes are generated while considering the system's geometry (and, optionally, dynamics). The method has been applied, sometimes with variations, to many classes of robots. This is the method used by V-Rep (Rohmer, 2013). Although the method could probably be modified for use in multi-query applications, the method is generally suited for single-query collision-free path planning problems. A related approach grows one RRT at the start and one at the goal, and attempts to connect the two in an efficient way (Kuffner & LaValle, 2000).

Some methods use a combination of offline map-building with online graph-search. One example is probabilistic roadmaps (PRM), which involves generating a graph of reachable configurations (nodes) and rapidly-computable local paths (edges) between these nodes (Kavraki, Svestka, Latombe, & Overmars, 1996, Kavraki, Kolountzakis, & Latombe, 1998, and Song, Thomas, & Amato, 2003). This graph can be searched using many different graph-search methods. Another related approach, Dynamic Road Maps (DRM), uses knowledge of

the robot forward kinematics, geometry, and sensor arrangement to generation a roadmap offline. A mapping between the operational space and the C-space is also generated offline. During runtime, the algorithm modifies the roadmap before performing a graph search for the final path (Leven & Hutchinson, 2002). This has been used to perform collision-free path planning in under 100ms in a dynamic environment using a point cloud of the environment and the A* search algorithm (Kunz, Reiser, Stilman, & Verl, 2010). Kunz et al. reported pre-processing time on the order of several hours for many of their setups. The time increased as the number of roadmap nodes increased, in a manner which appears quadratic but is not specified. The graph-search (online) time wasn't affected much by the number of roadmap nodes, likely because they use the A* search algorithm which only searches part of the graph in most situations (Hart, Nilsson and Raphael, 1968 and 1972).

The main idea behind A* search is that for a given node n_i, the cost, $f(n_i)$, is a sum of the cost of reaching that location, $g(n_i)$, and the estimated cost of reaching the goal from that location, $h(n_i)$. ("Cost" depends on the application. For instance, traveling across water could be more "expensive" than travelling across land.) In fact, Dijkstra's algorithm (Dijkstra, 1959), which only takes into account the path taken to a node, is a special case of the A* algorithm where no estimate $h(n_i)$ is used. Many heuristics could be used for $h(n_i)$, and for heuristics which do not overestimate the cost of reaching the goal, the algorithm is guaranteed to converge on the optimal solution, if any is available (Hart et al., 1968 and 1972).

3.4. Background on the Application

In order to gather data on indoor plants during their growth cycles, one promising option is to use a robot arm in conjunction with a robot base to position sensors to a proper

position and orientation. This paper focuses on building collision-free path plans with a ToF sensor (Microsoft Kinect v2) mounted on the end-effector of a robot arm (UR10). As shipped from the factory, the robot arm has no awareness of when it will collide with a plant, as shown in Figure 10.

Figure 10. Probing examples. Left: collision with a plant. Right: no collision.

Situations like this can easily be avoided when only one plant is in front of the robot arm. However, in plant-breeding applications, often dozens of plants will be in each growth chamber, which means that situations where the instruments (or possibly even the arm) collide with the plants will likely occur quite often. As such, the robot needs a method to avoid hitting plants while still allowing the instrument (in this case, the probe) to reach the goal. Additionally, when a collision-free path is not possible, the system should detect this, document it, and have a programmed response. In the current implementation, a message is displayed on the PC's screen and the robot arm approaches as close as it can in configuration space (C-space), also termed joint space. A future work is to decide how to deal with this as part of the overall robotic imaging system. The program could pick a different leaf, e-mail the researcher, or even choose an "optimal" collision-containing path or "optimal" collision-free approximate pose, using metrics such as a weighting of Euclidian distance and orientation offsets.

In applications such as plant breeding, where the robot is intended to interact with specific aspects of its environment, using various sizes and quantities of sensors, the robot will require a representation of the environment, in order to understand where to place sensors. This could be time-invariant and relatively complete as in the real-time-updating 3D models created by KinectFusion (Izadi, Kim, Hilliges, Molynequx, Newcombe, & Kohli, 2011) implementations, or incomplete as in the case of using a single frame of depth information or a laser scanner. Since information about the environment is therefore already available before the robot attempts collision-free path planning, adding a sensor-based collision avoidance scheme doesn't yield much benefit, and approaches using knowledge of the robot workspace have been considered.

Analytic approaches are infeasible on noisy plant-shaped Kinect feedback, and would require pre-processing to find geometric approximations. In contrast, the PRM and DRM families of algorithms can be configured or modified to operate in virtually any environment. Due to its ease of modification, and related ability to rapidly adopt to new configurations, such as in-field sensor changes, PRM is well-suited for applications with sensor requirements which may change often, such as autonomous plant phenotyping.

3.5. Materials and Methods

In this section, the framework proposed is developed. First, the basic PRM method is explained while only considering collisions with the end-effector, which is a 3D case. The approach is extended to the 6D configuration space, and enhancements are discussed.

3.5.1. 3D PRM

To illustrate the basic PRM method, we first considered only collisions with the end-effector. A sample plant environment was set up with four artificial plants (from artificialplantsandtrees.com) with varying leaf size and shape, and arranged with the tall plants in front. This represents the toughest case. Reaching over and around tall plants to approach a short plant is more difficult than: a) reaching over and around short plants, or b) reaching over and around tall plants to approach a tall plant. A sample start position and several goal positions were chosen. The positions were, in order: right, top, and back of an artificial Pink Germanium; top and left of an artificial Diffenbachia; right and top of an artificial Dracaena; and back, top, and right of an artificial Gold Guzmania. The plant arrangement is shown in Figure 11. The table is roughly the same height as the $z = 0$ location on the robot arm. The arm must reach over plants 1 and 4 in order to reach the locations attempted near plants 2 and 3. This case reduces to a three-dimensional path planning problem. First, an occupancy grid $Occ \subseteq \mathbb{R}^3$ (an array in memory, where 1 represents objects and 0 represents free space) was constructed by down-sampling an approximate mesh model of the Kinect's environment, as obtained using the default KinectFusion program, discussed in Chapter 2. The resolution of the occupancy grid is application-dependent, and for this paper, 1cm was chosen. This occupancy grid was morphologically dilated by 6cm, which is equivalent to modelling the end-effector as a ball with radius 6 cm. This will be elaborated in the whole-arm collision avoidance case. $N = 150$ randomly-generated nodes n were generated in the open space around the plants (in this case, $Y < 0, Z > 0$). Nodes n_i and n_j were connected by edge E_{ij} using a local planner if the collision-free line between them was in the free space, and if $d = |n_i - n_j| < 0.3 * \max(size(Occ)) = t$, the maximum allowable distance.

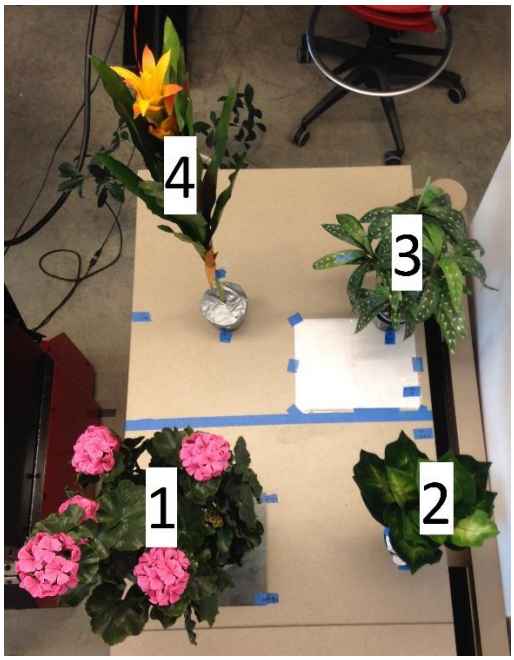

Figure 11. Four artificial plants. 1: Pink Germanium, 2: Diffenbachia, 3: Dracaena, 4: Gold Guzmania.

The resulting graph G was searched for the optimal solution using either depth-first search (DFS) or A* search. In the A* implementation presented in this paper, the distance between two nodes is measured by the 2-norm of the vector connecting them, representing straight-line edges. As a result, $g(n_i)$ is the sum of all edge distances on the shortest path from the start node to n_i. Hence, a 2-norm of the vector from the current node and the goal node is used for $h(n_i)$. This is a lower bound on the distance to the goal. At a given search step, neighbors of the current node are evaluated, and the node n with the lowest known function value $f(n)$ is chosen as the node for the next step. The lowest-cost successor node visited on the way to a node is stored in the data structure. The final node will to point to its successor, which will point to its successor, all the way back to the start node. The result is a sequence of sets of XYZ locations (joint values in the 6D case) which, when visited by the UR10 arm in its linear (joint-space in the 6D case) trajectory setting, will result in a collision-free path from the start pose to the goal pose. Further, this sequence is optimal given a graph and our cost function.

A summary of results is presented in Table 6, as averaged over 25 trials. A sample path in the environment is shown in Figure 12; the graph of nodes corresponding to this path is shown (without edges) in Figure 13. Key observations include: depth first search (DFS) is significantly slower than A*, and the search time for A* is significantly smaller than the construction time. This shows the need for using a limited-search method, such as A*, rather than an exhaustive-search method such as depth-first search, which is more suited for querying the same graph with the same goal, from multiple start locations. Additionally, this points to the validity of PRM-based approaches in this type of application: once the roadmap was constructed from the Kinect feedback, querying the roadmap for traversal between locations is computationally inexpensive. The average error column reflects the high probability that the nearest node can be connected to the goal via the PRM graph. The algorithm found a safe path to the goal for most queries, and only occasionally couldn't reach the goal, returning a destination which had error usually on the order of 15-20 cm. This suggests a need for roadmap enhancements or increased number of nodes, both of which are discussed in the next section. Finally, note how the sample path consists of 10 nodes. This is not always possible and is certainly not possible for many whole-arm-collision avoidance cases, including the one presented in this paper. Links of the arm would collide with the tall plants (1 and 2 in our case).

Table 6. 3D PRM results, averaged over 25 trials. $N = 150$ for each trial.

Method	N	Construction Time [s]	Search Time [s]	Error [cm]
A*	150	1.024	0.110	0.148
DFS	150	1.163	25.401	0.170

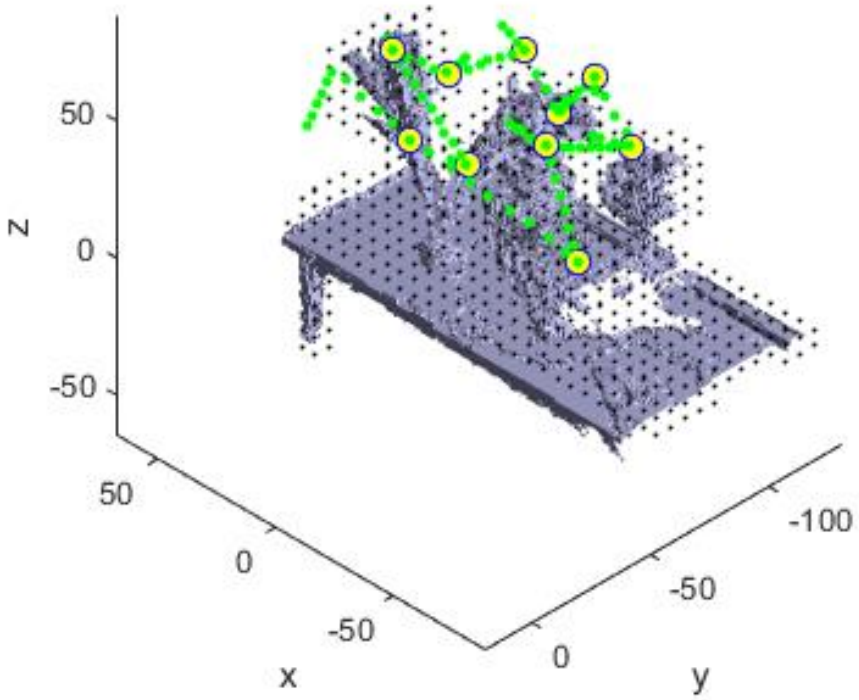

Figure 12. Sample 3D path generation. Black dots: dilated obstacles (sparsely plotted), gray: KinectFusion mesh, green dots: sparse path travelled, yellow circles: nodes visited.

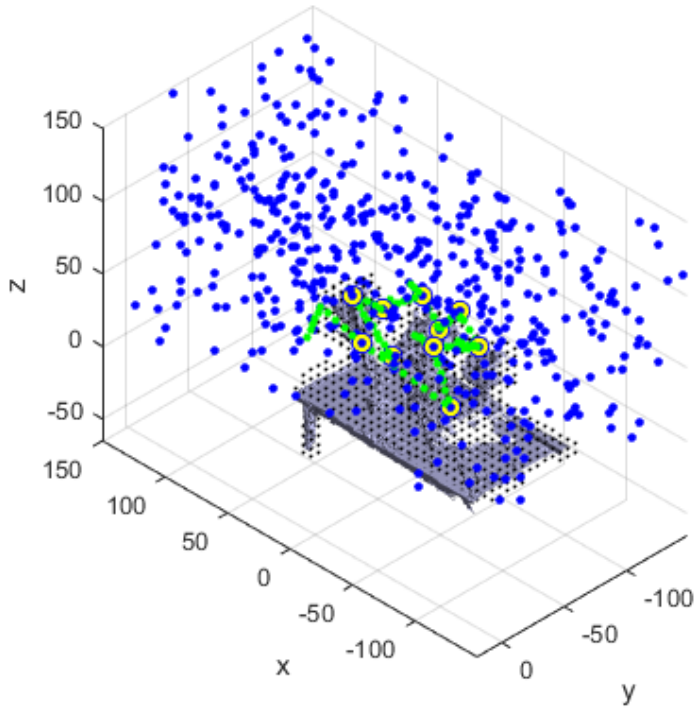

Figure 13. Sample graph. Black dots: dilated obstacles (sparsely plotted), blue dots: nodes, gray: KinectFusion mesh, green dots: sparse path travelled, yellow circles: nodes visited.

3.5.2. PRM for 6-DOF robot arm

For the case of planning for collision-free motion of the whole robot arm, the PRM method was still used. The presented approach consists of a roadmap construction phase and a query phase. During the roadmap construction phase, the first step is creating a 3D occupancy grid, as in the previous case. However, collision checking was carried out in a way that accounted for the robot arm's links and the sensors on the end effector, which will be collectively called "the robot". In this case, the only sensor was a Kinect v2, but the approach would extend to arbitrary configurations. To aid in computation, the robot was modeled as several spheres of radius 6 cm as shown in Figure 14. This is similar to the modeling used by Flacco et al. (2012). One major advantage in modeling the UR as several spheres is that checking for collision of these spheres with the environment is computationally inexpensive. This is equivalent to checking for collision of the spheres' center point with an expanded (dilated) occupancy grid. In other words, checking whether an object collides with a sphere is equivalent to checking whether that obstacle is one radius away from the sphere's center. Hence, expanding the obstacle by one radius has the same effect. To accommodate spheres of different sizes, a new occupancy grid could have been made for each unique radius, by dilating the original environment by the unique radius.

To avoid self-collisions on the system (Kinect-robot and robot-robot collisions), and to avoid twisting sensor wires unnecessarily, the joints were restricted. For reference, in the current implementation they were (in order from base to end-effector): [-180, 0; -180, 0; -180, 90; -180, 15; -180, 180; 125, 235].

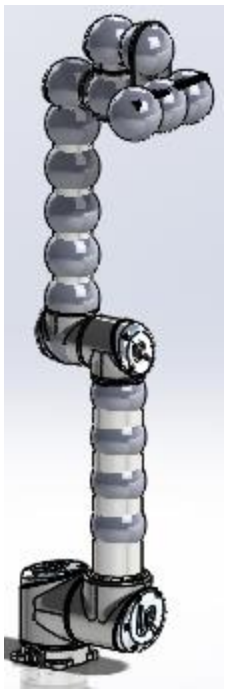

Figure 14. UR 10 modelled by spheres.

Motion in Cartesian space results in unpredictable robot motion, without complete knowledge of the joint-level motion controller being used. Leveraging the UR's internal processor and pre-built software, the only trajectory generation scheme that allowed our remote PC complete knowledge of the robot's swept path was a linear trajectory in C-space. To move the robot with C-space motion required a desired angle for all six robot joints, so this is a 6-dimensional space. The robot moved each joint linearly to the temporary goal value, such that every joint on the robot reached its final value at the same control cycle. As a result, the robot traced a path that is fully defined at each value of joint angle.

The start and goal joint positions were assumed to be known, since the UR10 arm is sending its current pose over TCP/IP constantly, and the imaging system can determine its goal once a leaf is identified, using the Kinect's mapping from RGB to depth space. Inverse Kinematics can be determined using the Robotic Operating System (ROS) package for the UR10, which is easily ported from C++ to MATLAB or virtually any other language

supporting floating-point arithmetic. The algorithm planned a path from a start through several goals, which were the C-space equivalent poses to the previous 3D case's XYZ goals.

The algorithm for a basic 6D PRM implementation is outlined below[1]. Sets of joints, termed a node, within the joint limits, $\theta_r \in \mathbb{R}^6$, were added if zero of the tracked points, including the end-effector (p_e) from the tracked set $P(n_i) = \{p_0\ p_1 \dots p_e\}$ collided with the obstacles, using forward kinematics. Edges between nodes were added if the distance between nodes was below a threshold (for the 6D case, if $d = |n_i - n_j| < k * |\theta_r| = t$, k chosen as 0.2 in these examples) and the 6D line between nodes (sparsely calculated every 3 degrees, to reduce computation), as defined by an extension of the Bresenham algorithm (Bresenham, 1965), resulted in zero collisions.

$N \leftarrow Number\ of\ Nodes$
$i = 0$
While $i < N$
    $n_i \leftarrow$ Random set of valid joint angles
    If $\forall\ p \in P(n_i), p \notin Occ$
        Add node
        $i = i + 1$
        For all n_j s.t. $\|n_i - n_j\| < t$
            If $\forall\ p \in \{\cup\ P(n_l) \mid n_l\ along\ line\ between\ n_i\ and\ n_j\}, p \notin Occ$
                Add edge e_{ij}
            End if
        End for
    End If
End while

Figure 15. Default PRM

Now that the construction phase was completed, the graph was queried with a start and goal position. In all variants discussed, the query and search is only carried out using the largest connected component. A discussion on a simple method to increase connectedness is presented

[1] I say "a basic 6D PRM" because some other basic PRM planners only allow a node to add edges to its k nearest neighbors (Song, Thomas, & Amato, 2003).

later on. The node nearest to the start and goal were labeled as the start and goal nodes, respectively. In theory, any 6D graph search method could then be used. However, due to the graph size and hardware limitations, breadth-first search was found to crash the MATLAB environment. Other related recursive and exhaustive-search methods would likely result in similar results. Additionally, these methods generally are slower than limited-search method such as an algorithm from the A* search family, including the uninformed Dijkstra's algorithm. Dijkstra's algorithm was shown by Kunz et al. (2010) to generally take longer than a more-informed A* search on PRM planners.

In this implementation, an A* implementation was used. As in the 3D case, distance between nodes is measured by the 2-norm of their difference (now a 6D vector), and the 2-norm of the vector from the current node and the goal node is used for $h(n_i)$. Again, this sequence is optimal given a graph and our cost function.

3.5.3. 6D PRM variants

The basic procedure outlined above works well for general cases, and can be adapted to achieve a desired balance between computation time and various motion characteristics, such as minimum swept Cartesian volume, or increase the nearness of the final pose to the true goal pose. A good discussion of several variants was presented by Song (Song, Thomas, & Amato, 2013). Variants explored in this paper are aimed at applications where the robot arm is known to operate in a subset section of its workspace, with additional joint constraints imposed by, for instance, wires attached to the arm.

The three variants presented are: adding nodes with an end-effector near the plants (in front of the robot, above the table); adding nodes with the end-effector, $p_e(n_i) = [p_{ex}(n_i) \quad p_{ey}(n_i) \quad p_{ez}(n_i)]$ (cm), near the goal (including attempting to directly connect the

goal); and adding nodes near nodes with a low number of neighbors (these nodes are likely in difficult-to-reach locations, and potentially part of a disconnected part of the graph). These variants are outlined and compared.

Although it is possible that another research team studied similar enhancements, the author has not seen these published elsewhere. These enhancements are intuitive, require a minimal amount of additional code, and are shown to be effective at producing highly useful PRM Graphs with a low number of nodes compared to other published results, which can range on the order of several thousand nodes for a 6-DOF robot.

The "Near Plants" variant adds nodes with end-effector location in the operational space (XYZ) near the plants. This is very similar to basic node-addition phase.

$N_p = floor(N/2)$
$i = 0$
While $i < N_p$
    $n_i \leftarrow$ Random set of valid joint angles
    If $(\forall\, p \in P(n_i), p \notin Occ)\ \&\ p_{ey}(n_i) < 0\ \&\ p_{ez}(n_i) > 0\ \&\ p_{ez}(n_i) < 0.80$
        Add node
        $i = i + 1$
        For all n_j s.t. $\|n_i - n_j\| < t$
            If $\forall\, p \in \{\cup\, P(n_l) \mid n_l\ along\ line\ between\ n_i\ and\ n_j\}, p \notin Occ$
                Add edge e_{ij}
            End if
        End for
    End If
End while

Figure 16. Adding nodes near plants

Adding nodes near the goal introduces d_{eMin}, a minimum distance threshold. This can be tuned to the accuracy required by the application, and could be extended to include orientation. Position and orientation are important for most applications, including plant-probing.

```
startTime ← Current Time
i = 0
G′←Largest connected component on G
Attempt to connect goal q_g to G′
        Success→Break
        Fail→Continue
While Current Time < startTime + 5
        n_i←Random set of valid joint angles
        If ∀ p ∈ P(n_i), p ∉ Occ & ‖p_e(q_g) − p_e(n_i)‖ < d_eMin
                i = i + 1
                For all n_j s.t. ‖n_i − n_j‖ < t
                        If ∀ p ∈ {∪ P(n_l) | n_l along line between n_i and n_j}, p ∉ Occ
                                Add edge e_ij
                                If n_i ∉ G
                                        Add n_i
                                        If ‖q_g − n_i‖ < 0.15
                                                Quit Adding
                                        End If
                                End If
                        End If
                End For
        End If
End While
```

Figure 17. Adding nodes near end-effector at desired pose

The "low neighbors" routine ("k neighbors") is called to add nodes near low-neighbor nodes, until every node n_i has more neighbors $b(n_i)$ than some minimum number b_m. The method is repeated multiple times (3 in these examples) to raise the number of neighbors of newly-added points. In this context, "near" has a very specific meaning, which will now be defined.

The tolerance T can be defined by a vector which is larger in high-freedom joints, and which is guaranteed to have a magnitude of $c * t$. For $c < 1$, therefore, points generated with all joints within this tolerance will be closer to the low-neighbor node than $c * t$.

$$T = c * \frac{\theta_r}{\|\theta_r\|} * t \tag{15}$$

Valid means the joint values are within the overall joint limits *and* within $n_i \pm T$.

```
N_T ← Number of nodes on G
i = 0
b_m ← Minimum number of Neighbors
While i < N_T
    startTime ← Current Time [s]
    While b(n_i) < b_m && Current Time < startTime + 5
        n_i ← Random set of valid joint angles
        If (∀ p ∈ P(n_i), p ∉ Occ)
            For all n_j s.t. ||n_i − n_j|| < Threshold
                If ∀ p ∈ {∪ P(n_l) | n_l along line between n_i and n_j}, p ∉ Occ
                    If n_i ∉ G
                        Add n_i
                    End If
                    Add edge e_ij
                End if
            End for
        End If
    End While
    i = i + 1
End while
```

Figure 18. Adding nodes near low-neighbor nodes

3.5.4. Experiment, with motion

Qualitatively, it was found that doing the roadmap construction and enhancement in a particular order produced paths which appeared smoothest, while keeping construction time and search time low. The arrangement proposed is as follows.

1. Start with an initial, Default roadmap construction. ($N = 500$)
2. Add nodes near plants. ($N_p = 250$)
3. Increase connectivity by adding nodes near low-neighbor nodes ($b_m = 5$, 3 iterations)
4. Add nodes near the goal during the search phase. ($d_{eMin} = 0.1$)

With $N = 500$, and the same goal positions as in the 3D and 6D cases presented, during a sample path, the robot achieved poses near the goal locations, as shown in Figure 19.

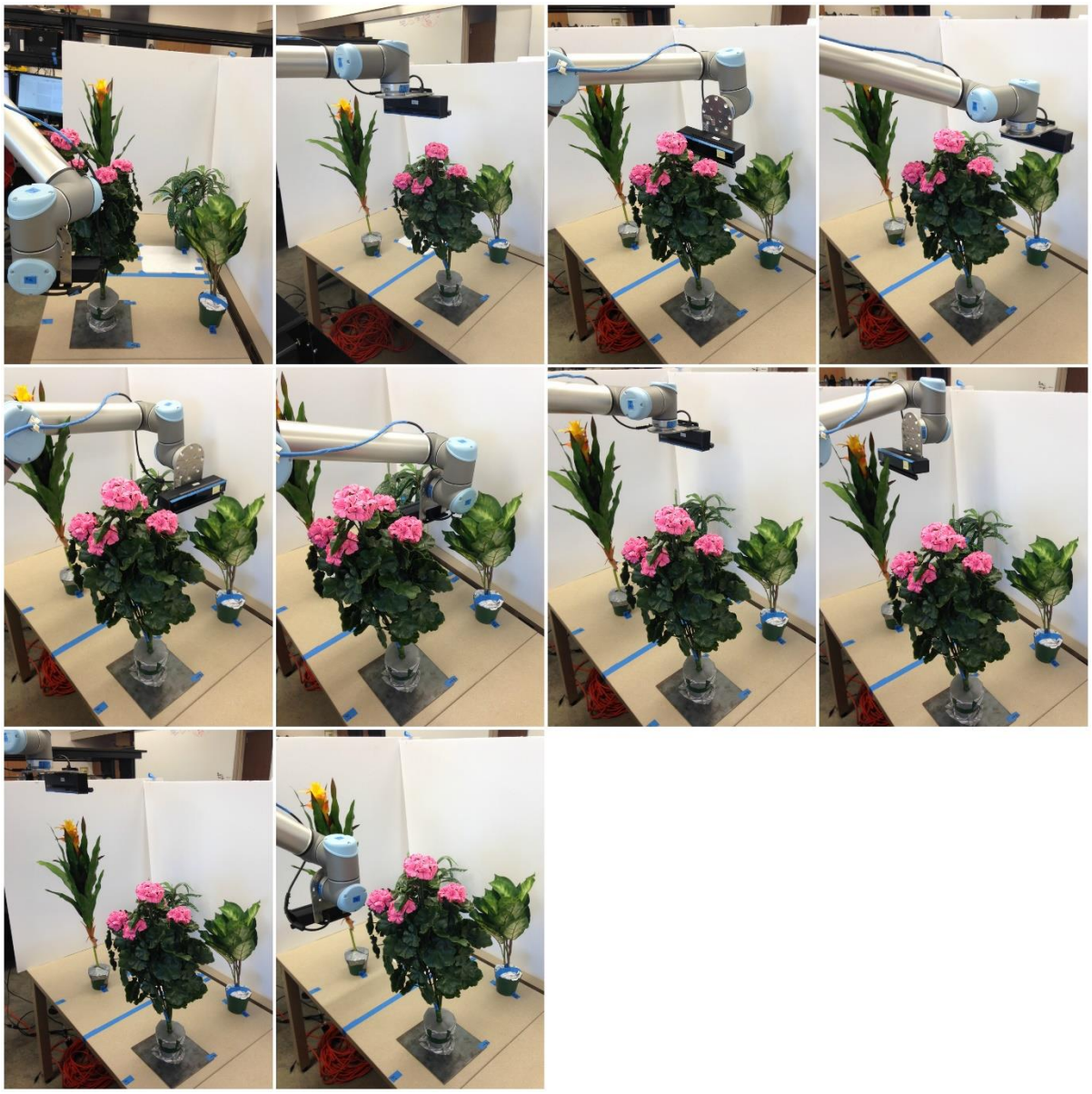

Figure 19. PRM example: the PRM framework generated paths approaching each goal location in succession. Starting with top left, left-to-right, top-to-bottom (in visitation order): right, top, and back of an artificial Pink Germanium; top and left of an artificial Diffenbachia; right and top of an artificial Dracaena; and back, top, and right of an artificial Gold Guzmania.

3.6. Results and Discussion

These methods were studied one at a time, with key results presented in Table 7. The construction time does not include the time required to load and down-sample the mesh to

create the occupancy grid, as this is the same for all methods, and will likely require negligible computation time in programs that are already generating models of the environment in real time, such as KinectFusion. N_s is the number of nodes added before roadmap enhancements, and was kept at 150 for all trials (except the second "default" method, with $N_s = 225$ to allow fair comparison with the "add near plants" enhancement). Keeping N_s this low helps illustrate the weaknesses and strengths of each method. With high N_s, the error decreases and the improvements of each enhancement are mainly seen qualitatively by looking at the distribution of the set of end-effector locations, $S = \{\cup\, p_e(n_i) | n_i \in G\}$. 25 trials were simulated, each with the same 10 goal locations, which were the C-space configurations resulting in the same $p_e(\cdot)$'s as the XYZ's in the 3D case.

As can be seen, adding nodes near the goal is an effective enhancement for reducing error, but adds significant search time. Adding nodes near plants reduces error more effectively than simply increasing N to the same number that adding nodes near plants results in. As pointed out earlier, one major benefit of this enhancement is that the paths generated are usually closer to the plants, resulting in a smoother-looking motion. As side effects, this extended construction time, kept search time low, and reduced errors slightly. Finally, adding nodes near nodes with few neighbors was found to reduce connected components by an average 38.8% (from 36.5 before enhancement to 22.32 after enhancement). The hidden benefit of increasing graph connectivity, thereby increasing number of searchable nodes and generally smoothing the paths, can be seen qualitatively when the algorithm was used on the real robot.

As a fair disclaimer, it is possible that these relationships and benefits of enhancement don't hold at every N_s value. Due to the high number of constants used in just the three enhancements described in this paper, which were kept as simple as possible, the number of

parameters and the range of possibilities which would need to be explored to optimize the PRM frameworks is enormous, and is beyond the scope of this paper. This is one disadvantage to using the highly-effective probabilistic or heuristic methods for problem solving.

Table 7. Testing PRM variants. N_s is nodes for each trial's initial roadmap (before enhancements, if any). Construction time [s], search time [s], error [deg], and Euclidian distance error (m) averaged over 25 trials. Each trial consisted of 10 separate goal locations.

Method	Ns	Construction [s]	Search [s]	Error [deg]	Error [m]
Default	150	1.198	0.269	20.606	0.180
K Neighbors	150	4.759	0.429	16.764	0.158
Add Near Plants	150	2.684	0.342	14.769	0.144
Default	225	2.550	0.349	15.874	0.157
Add Near Goal	150	1.278	1.191	12.771	0.098
Proposed Method	500	32.540	0.461	3.652	0.027

3.7. Conclusion

As demonstrated in section 3.6, the generic PRM can generate roadmaps rapidly, and the A* algorithm can return paths of acceptable poses rapidly. The roadmap enhancements presented allow the robot to approach closer to the goal locations in exchange for computation time. The PRM generated reasonable paths which attained the desired positions, reliably. In contrast to previous approaches, the current implementation uses a low-cost end-effector-mounted sensor and a novel implementation of the PRM algorithm which is shown to work well in a real-world application. In future work, the framework can be further optimized for speed and achieving specific path objectives, such as balancing position and orientation when a collision-free path is unattainable. Specific objectives can be achieved by changing the cost functions used in finding the "nearest node to the goal" and in the A* algorithm (currently Euclidian distance in C-space). Paths can be smoothed by, for instance, attempting to skip nodes on the returned path by checking for a collision-free path between non-adjacent nodes. Additionally, the framework can be integrated with the KinectFusion algorithm for near-real-

time collision-free path planning, without the need for an external sensor. This would involve porting the code to C++ and interacting with the mesh directly, rather than saving it as a separate file.

References

Bresenham JE. Algorithm for computer control of a digital plotter. *IBM Systems Journal* 1965; 4(1):25–30.

Brock, & Khatib. (2002). Elastic Strips: A Framework for Motion Generation in Human Environments. The *International Journal of Robotics Research*, 21(12), 1031-1052.

Cheung, E., & Lumelsky, V. (1992). A sensitive skin system for motion control of robot arm manipulators. *Robotics and Autonomous Systems*, 10(1), 9-32.

De Luca, A., & Flacco, F. (2012). Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration. *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*, 288-295.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik,1*(1), 269-271.

Feddema, J.T., & Novak, J.L. (1994). Whole Arm Obstacle Avoidance for Teleoperated Robots. *IEEE International Conference on Robotics and Automation*.

A. Fenucci, M. Indri & F. Romanelli, "A real time distributed approach to collision avoidance for industrial manipulators," *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Barcelona, 2014, pp. 1-8. doi: 10.1109/ETFA.2014.7005160

Flacco, F., Kroger, T., De Luca, A., & Khatib, O. (2012). A depth space approach to human-robot collision avoidance. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 338-345.

Gandhi, D., & Cervera, E. (2003). Sensor covering of a robot arm for collision avoidance. *Systems, Man and Cybernetics, 2003. IEEE International Conference on, 5*, 4951-4955.

Haddadin, S., Belder, S., & Albu-Schaeffer, A. (2011). Dynamic motion planning for robots in partially unknown environments. *IFAC World Congress*.

Hart, P., Nilsson, N., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2), 100-107.

Hart, P., Nilsson, N., & Raphael, B. (1972). Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *SIGART Bull*. 37 (December 1972), 28-29. DOI=http://dx.doi.org/10.1145/1056777.1056779

Izadi, S., Kim, D., Hilliges, O., Molynequx, D., Newcombe, R., & Kohli, P. (2011). KinectFusion: real-rime 3D reconstruction and interaction using a moving depth camera. *In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*.

Kavraki, L., Kolountzakis, M., & Latombe, J. (1998). Analysis of probabilistic roadmaps for path planning. *Robotics and Automation, IEEE Transactions on*, 14(1), 166-171.

Kavraki, L., Svestka, P., Latombe, J., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4), 566-580.

Khatib, O. (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *International Journal of Robotics Research*. 5(1). Pp. 90-98.

Kroger, T. (2011). Online Trajectory Generation: Straight-Line Trajectories. *Robotics, IEEE Transactions on*, 27(5), 1010-1016.

Kroger, T., & Wahl, F. (2010). Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events. *Robotics, IEEE Transactions on*, 26(1), 94-111.

Kunz, T., Reiser, U., Stilman, M., & Verl, A. (2010). Real-time path planning for a robot arm in changing environments. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 5906-5911.

Kuffner, J. J. & LaValle, S. M. (2000). "RRT-connect: An efficient approach to single-query path planning," *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, San Francisco, CA, 2000, pp. 995-1001 vol.2. doi: 10.1109/ROBOT.2000.844730

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University.

Leven, P., & Hutchinson, S. (2002). A Framework for Real-time Path Planning in Changing Environments. *The International Journal of Robotics Research*, 21(12), 999-1030.

Lumelsky, V. J., & Cheung, E. (1993). Real-time collision avoidance in teleoperated whole-sensitive robot arm manipulators. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(1), 194-203.

Pieruschka, R., & Poorter, H. (2012). Phenotyping plants: genes, phenes and machines. *Functional Plant Biology*, 39(11), 813-820.

Rohmer, E., Singh, S., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 1321-1326.

Royal Society of London. (2009). *Reaping the benefits: science and the sustainable intensification of global agriculture*. London: Royal Society.

Rai, M., Reeves, T. G., Pandey, S., Collette, L., & Food and Agriculture Organization of the United, N. (2011). Save and grow: a policymaker's guide to sustainable intensification of smallholder crop production / [final technical editing, Mangala Rai, Timothy Reeves and Shivaji Pandey; lead authors: Linda Collette ... [et al.]]. Rome: Rome: Food and Agriculture Organization of the United Nations.

Shaffer, C. A. (1991). Real- time robot arm collision detection for telerobotics. *Computers and Electrical Engineering*, 17(3), 205-215.

Singh, L., Stephanou, H., & Wen, J. (1996). Real-time robot motion control with circulatory fields. *Robotics and Automation, 1996. Proceedings. 1996 IEEE International Conference on,3*, 2737-2742.

Song, G., Thomas, S., & Amato, N.M. (2003). A general framework for PRM motion planning. *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 3, 4445-4450.

CHAPTER 4: GENERAL CONCLUSIONS

As seen in chapter 2, a rover was developed which met the design criteria for operating in a growth chamber for a whole day. The Kinect sensor was shown to be accurate enough to position sensors within a few centimeters (often within 3 cm) of the desired location. This is accurate enough to enable the system to use a more accurate short-range sensor, such as a line scanner, to get a highly-accurate reconstruction of the desired leaf. Local orientation estimates were discovered to be one weakness of the sensor, often producing results 20 to 30 degrees away from the desired orientation. If this error is not fixed, it may make sense to assume a fixed angle while going near the plants. Then, the line scanner's result could be used for highly accurate probing. The source of this error was shown to be mainly a result of the local noise, rather than the KinectFusion's estimation of pose. This suggests that, with additional work (such as using principal components analysis to find the surface normal of the plane which fits a cluster of points near the goal location), the sensor may be able to position sensors at the desired angle as well.

During the surface normal and accuracy tests, it became apparent that the arm may collide with plants if no path planning procedures were implemented. Off-the-shelf algorithms could not be found for use in our system, and as far as the author is aware, research on collision-free robot arm motion in plant spaces is virtually non-existent. To enable the robot arm to approach multiple plant leaves in the same environment without colliding with the plants, a variant of the Probabilistic Roadmaps (PRM) technique was implemented. The system modeled the robot arm by an overlapping set of spheres, allowing quick collision checks. The proposed variant took 32.5 seconds for roadmap construction, and allowed the robot to query

a collision-free path in 0.46 seconds. The robot reached locations 3.65 degrees away from the desired pose, which was (using data from each sample, not converting 3.65 degrees) 2.7 cm.

Future work for the path planning portion of this work should include smoothing of the generated paths (by, for instance, attempting to skip nodes on the returned path by checking for a collision-free path between non-adjacent nodes), and implementing the MATLAB code in C++ alongside the modified KinectFusion application described in Chapter 2. The framework can be further optimized for speed and achieving specific path objectives, such as balancing position and orientation when a collision-free path is unattainable. This could be achieved by changing the cost functions used in finding the "nearest node to the goal" and in the A* algorithm (currently Euclidian distance in C-space). The overall end result would be near-real-time collision-free path planning, without the need for an external sensor.

Finally, leaf-segmentation algorithms, a redundant localization system (such as QR codes in the facility or a 2D SLAM algorithm), and data management systems should be added to the system to complete the robotic package.

The work presented in this thesis can be used as the foundation for implementing a fully-autonomous data-acquisition system for use in plant phenotyping. Autonomously acquiring traits such as estimated biomass, plant height, number of leaves, chlorophyll fluorescence, and various spectral responses, can allow researchers to conduct experiments for a relatively low cost. Additionally, having a mobile rover, such as the one presented, acquire the data in multiple growth chambers allows a researcher to conduct multiple related studies in different environments, enabling the study of the interaction between genotype and the environment. This, in turn, allows the researchers to determine desirable strains of crops.

APPENDIX A: INSTRUCTIONS FOR BASIC USE OF ROVER BASE

RoAd/Enviratron Testing Instructions

(2016-08-03 YMD)
Abbreviations: MC Motor Controller, RC Remote Control, S_R_[M]_[D] script RoAd [month] [date] (S_E script Enviratron),

1. Front is the side with the robot arm (side with the bump)
2. Turn on rover's RC controller. Flip the appropriate upper-left switch to "1 F Mode" (RC. 0 is Script/tape)
3. Press one E-Stop in (such that current is blocked)
4. Turn on rover with 30mm main switch.
5. TeamViewer into the rover's PC.
6. Open Roborun+ (allows interface with MC).
7. **Configuration** has settings. **Run** has real-time status of MC. **Console** has places to send commands. **Scripting** allows you to edit one "MicroBasic" script at a time.

a. In Scripting, click the "Open" Icon and find most recent script (e.g. S_R_2_8) in "TestingOrganization" (Shortcut on Desktop). MC's only have 8 remote-editable variables, so they were given high-level "functions". These MC commands are in the first few lines of every script.

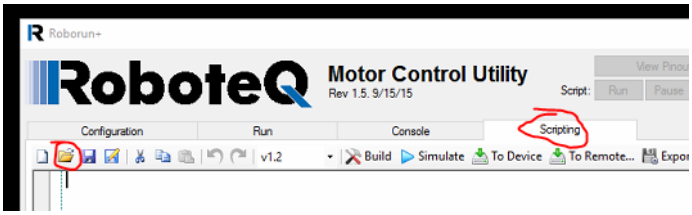

Figure 20. Roboteq Motor Control Utility for most MC/Tape sensor troubleshooting and settings.

8. Click "Yes" on popup saying roughly "Controller Found. Read Configuration?" (Your MC settings are now in "Configuration"). If COM2 is not open (Bottom of screen), switch COM Port drop-down (upper-right) to COM2 (it was probably on Auto and detected the secondary MC).
9. Safely test with the rover. Try:
    a. RC control
    b. Tape-following (upper-left switch to "0 Gear")
        i. General (no MC Commands required)
        ii. Track Left at a fork (MC Commands required)
        iii. Track Right at a fork (MC Commands required)
        iv. Go to a specific simulated chamber (use long cardboard simulator) (MC Commands required)
        v. Into and inside the chamber in High Bay (MC Commands required)
    c. Computer Control (MC Commands required)

d. Making Rover stop moving, only using Console commands (MC Commands required)

Shutting down Robot:
1. Press one E-Stop in (such that current is blocked).
2. Turn off rover's pc.
3. Turn off rover with 30mm main switch.
4. Turn off RC controller.
5. (During testing only) Verify that everything powered down properly. Remove red alligator clip from the battery + terminal, and clip onto an 8020 bar.

C++ Roboteq Class:

- Uses Windows API (VS2012) from http://www.roboteq.com/index.php/support/downloads
- OLD API: See sample.cpp. This file contains a C++ console program script similar to the MicroBasic script on the main MC. This was eventually abandoned, as the response rate was too slow. The MC should run its script, and the user can modify User Variables 1-8 in a similar manner as the Roborun+ procedure. Use an instance of the RoboteqDevice class, and use its member function "SetCommand". All standard Roboteq runtime commands have a macro, except _R (MicroBasic Run, equivalent to "12") and one other (_C, CAN, is likely 25.).

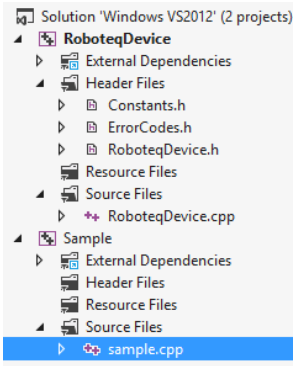

Figure 21. C++ Roboteq test code uses RoboteqDevice class and sample.cpp.

- Basic procedure: initialize MC1 and MC2 with device.Connect(\\\\.\\com1) etc.. Set the watchdog timer to 0 (see MC manual) device.Setconfig(_RWD, 0) for both. From there onward, use only MC1 since MC1 is "master" and runs the script. Use device.SetCommand(12, 2)) to run the onboard script. (This can be done efficiently using the customized device.SetVar method I added to the class in EnviratronFusion projects.) If you want to pause the program, Roboteq added sleepms(int milliseconds); similar to Windows C++ function sleep(). Example: set VAR 8 to 3 ("go to chamber 3") with device.SetCommand(_VAR, 8, 3);

A few notes:
1. Console commands are equivalent to what the C++ program can send to the MC

2. **E-Stops** are physically connected to VMOT (the voltage source for the motors). Hit an E-Stop, and there is extremely high probability that the motors will stop running. **Use these in panic situations.**
3. As a last resort, pull the **red alligator clip** off of the battery. This will **GAURANTEE** the rover motors **stop** being driven by the battery. However, damage to computer/data/UR10 may result. MC's will not be affected.
4. Don't run over wires. Mecanum wheels will severely damage the wires.
5. Procedure is very similar for Enviratron and RoAd rovers. Some in-script variables should be changed (search in script file for "Env" or "RoAd"; C++/Roborun+ commands to track Right / Left are flipped (1→2, 2→ 1).
6. Sensors may need to be zeroed (adjust pulse input wires as necessary. Connect MagSensor's 2 pin to Computer's RS232 2 pin, and Mag 3 – PC 3. Open MagSensor Control Utility, verify sensor is correct using small scrap of magnetic tape, move robot so sensor is over non-ferrous material, and click "Calibrate Zero".

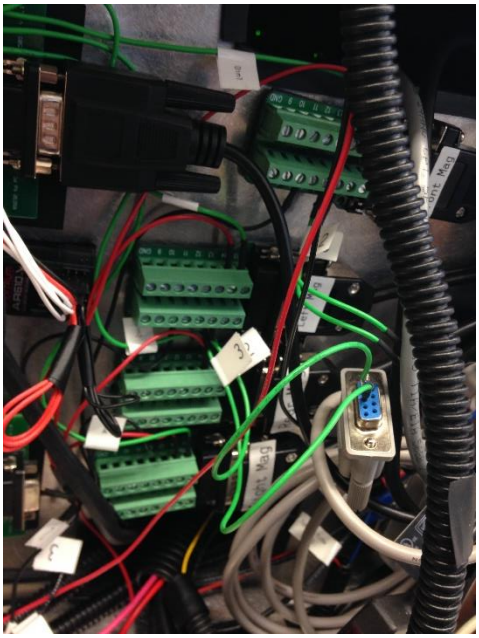

Figure 22. Use green Serial/RS232 wires to communicate with computer.

Feedback:

RC Control notes/tips/issues:

Tape notes/tips/issues:

Computer Control notes/tips/issues:

C++ Program Control notes/tips/issues

Other (organization, variable assignments, Mechanics/design of rover, etc.):

APPENDIX B: BUILD OUTLINE

Building a Rover Similar to the RoAd or Enviratron Rovers
A quick walk-through by Dylan Shah

This section explains key aspects of our robot build process, in a compact bullet-point format. This does not contain detailed (micromanaged) instructions, nor does it contain instructions on common skills such as soldering. The intimate details are specific to a particular build, and can either be further customized or can be copied from the existing implementation.

Abbreviations for following pictures: CAN controller area network communication standard, E-stop emergency stop, FL front left, FR front right, MC motor controller (MC.1.1 means motor controller 1, motor 1), MS main switch, NC normally closed, NO normally open, PC personal computer, PWM pulse-width modulation, R resistor, RC remote control, RL rear left, RR rear right, RS232 RS-232 communication standard, TS tape sensor VMOT voltage for motor (Roboteq labels this on their MC's),

1. Start with SolidWorks Drawings of a robot that fits design specifications and has all critical dimensions (hole placements, part sizes) as accurate as possible. Sketch or write out key steps not included in this manual. If holes aren't correct, you will have to manually make some with a drill, Dremel, or band saw, etc.

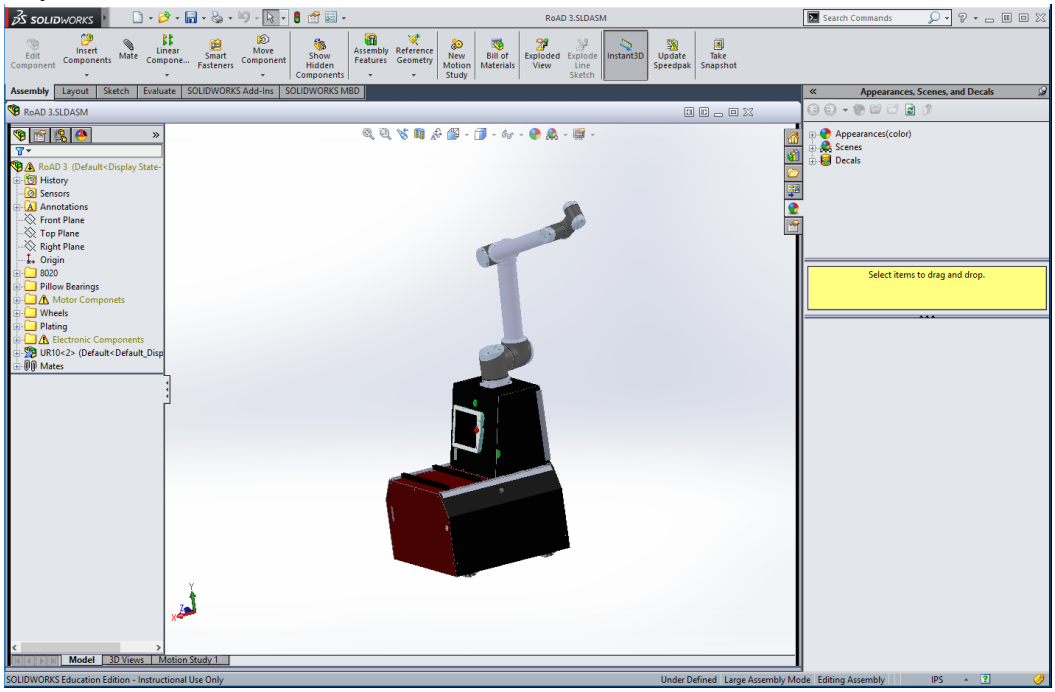

Figure 23. SolidWorks of rover.

2. Begin assembly of the main frame, motors and LoveJoys.

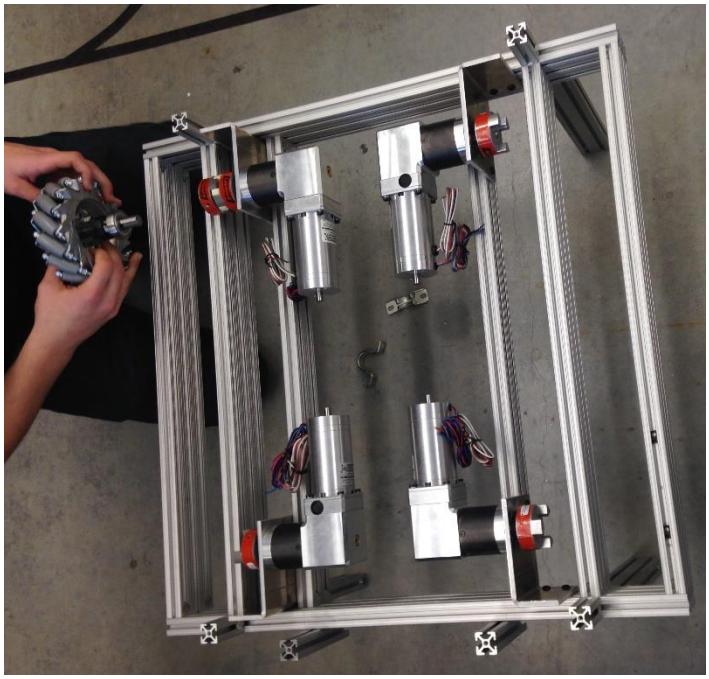

Figure 24. Frame, motors, and LoveJoys.

3. Add bearings, wheels, and collars. Verify that all components added so far, including the set screws on bearings and wheel collars, are properly tightened (tighten by hand, use lock washers wherever possible). These will be difficult to adjust later (likely requiring a forklift or specialized repair bay).

Figure 25. Mecanum wheels, bearings, and collars.

4. Add plating, bus bars, and electrical boxes (picture shows step partly completed). Some plating can only go on in a specific order.

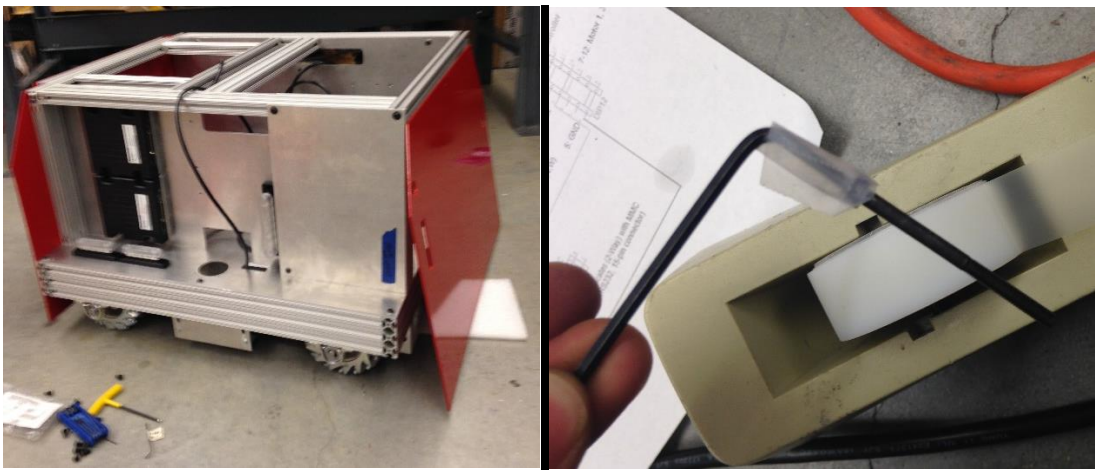

Figure 26. Left: Some plating, bus bars, and electrical components added. Right: get creative. Sometimes taping a screw onto an Allen wrench aids in adding a difficult screw.

5. (Optional) Shorten the MS cables. First, cut them, leaving them a few inches (I recommend 4) longer than you think you need. Strip the outer black sheath back approximately 2 cm. Keep as much of the foil and uninsulated wires intact as possible. Strip ~2mm of wire on the tip of each insulated sub-wire. Remove the case from the old DB15 connector, exposing the old wires. Solder the new stripped wires to the corresponding connection point on a new DB15 connector. The stray uninsulated wires and foil should be soldered with the GND pin (black wire) like the old DB15 had. This grounded sheathing helps eliminate noise. Basically mimic the Roboteq "old DB15". Extra Optional: use heat-shrink wrap to insulate connections after soldering.

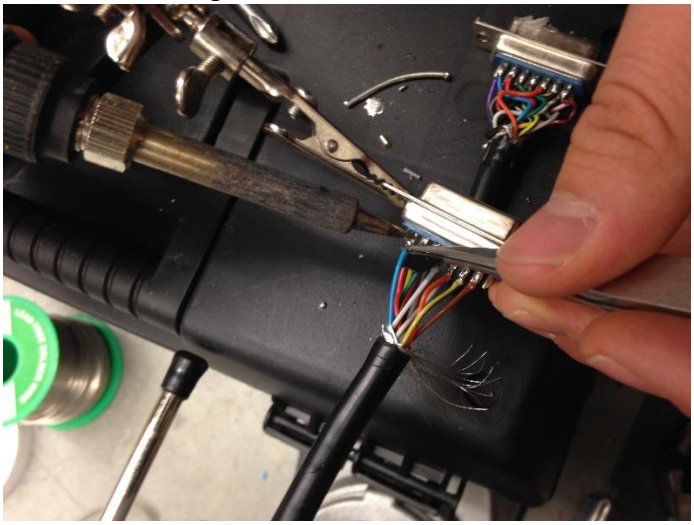

Figure 27. Shortening MagSensor cables.

6. Connect wiring.
   a. Use: cable guides, insulating heat-shrinkable tubing, cable wrap, nylon-insulated crimp connectors, and properly-gauged wiring. All non-professionally-purchased wiring was DC, so it can be sized according to charts available online based on expected current and wire length.

b. Follow the wiring diagram. If the diagram doesn't make sense, either there is an error in the diagram or you are misinterpreting – discuss with group members. Note that the diagram does not mention: motors (MC.1.1 FL MC.1.2 RL, MC.2.1 FR, MC.2.2 RR), AC components (connect +48 terminal to 48V bus bar, and GND terminal to GND Bus Bar, plug in components such as PC and UR10), the 48/24V converter has two black GND wires – connect both to the GND bus bar
c. Use the fuses and diode where recommended.
d. Label each wire as they are put on. "Brother" brand label makers work nicely.
e. Resistors are pull-down resistors. Fuses & e-stops are for safety. Diode allows high voltage from regeneration (stopping, or being externally pushed) to safely flow to the battery.
f. The specific MC input port for each MagSensor and RC PWM output is arbitrary.
g. From experience, grounding RS232 lines isn't required – this is likely because every component shares a common ground. However, there is little harm in using the RS232 "ground line" recommendation.
h. When in doubt, read the spec sheets and search internet forums etc. on the relevant topic.

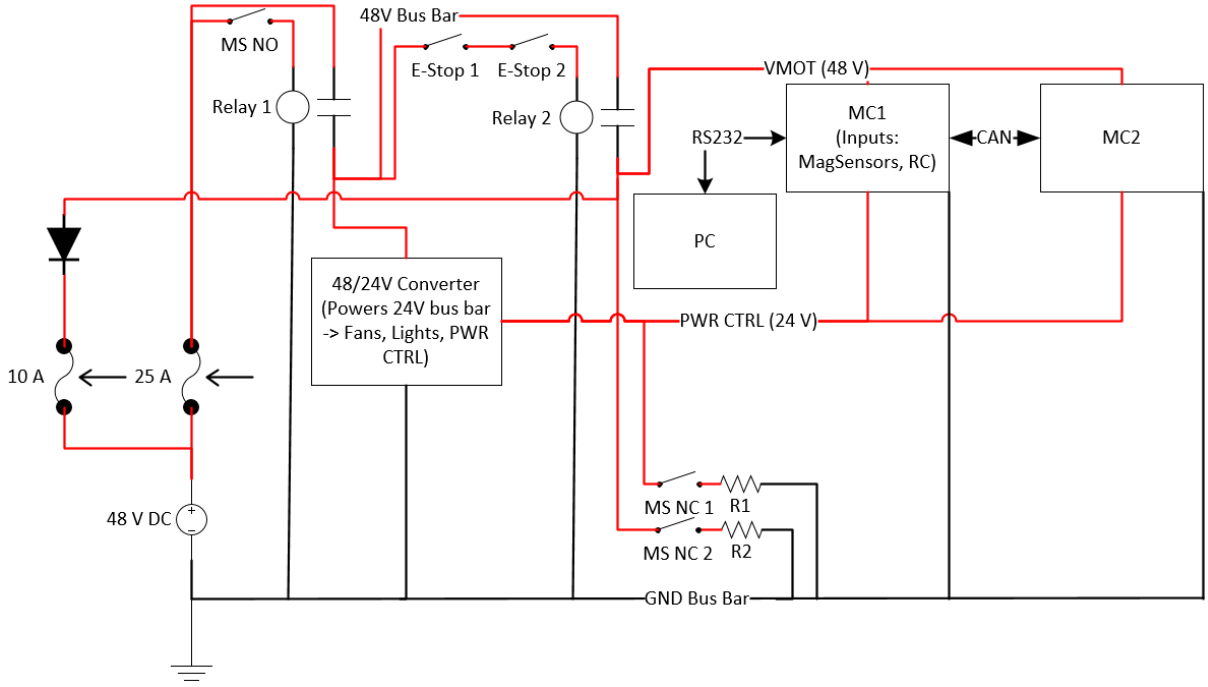

Figure 28. Shorthand wiring diagram.

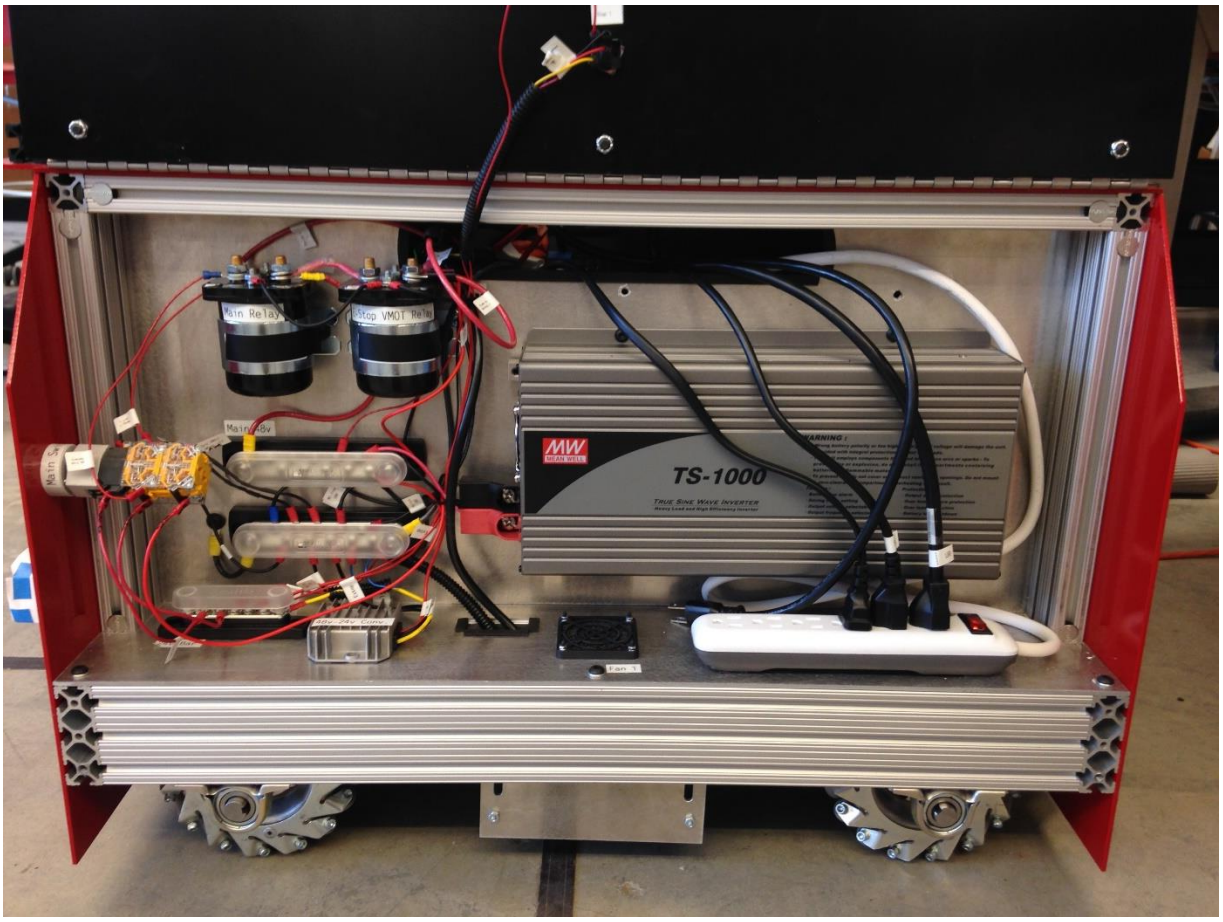

Figure 29. "Power Side" of the rover.

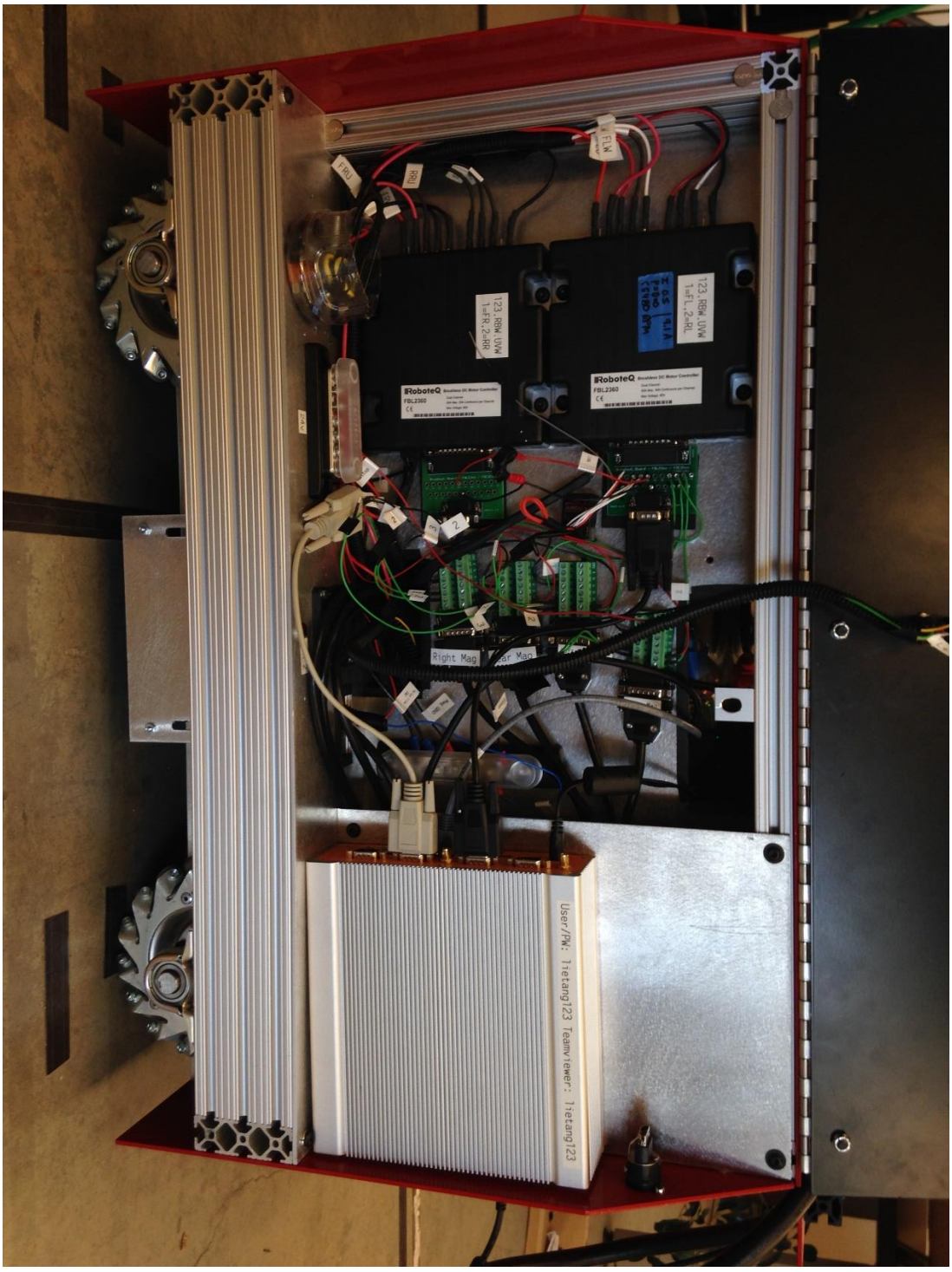

Figure 30. "Brain Side" of the rover.

7. Troubleshoot. Make sure connections are as desired, and all are safe.
8. Program, etc. Using Roborun+: download software to MC1, download profiles (collections of settings) to MC1 and MC2. Using MagSensor PC Utility, zero (calibrate) the TS's. Using Microsoft Visual Studio, program with the Roboteq C++ API, and/or the lab's software. RS232 communication with MC's is preferred (USB is unstable).

9. The final product *might* look like the robot below.

Figure 31. Finished rover, without end-effector sensors (optional).