2019-04-24

CERBERUS

Kaan Kaan Emir
Worcester Polytechnic Institute

WORCESTER POLYTECHNIC INSTITUTE

CERBERUS: Computer Enabled Robotic Base Enhancing Remote Unmanned Security

Hannah Olshansky (RBE/ECE)

Kaan Emir (RBE)

MQP-KZS-1803

Advised by:

Professor Kenneth Stafford (ME/RBE)

Professor Alexander Wyglinski (ECE)

April 23, 2019

Abstract

The Computer Enabled Robotic Base Enhancing Remote Unmanned Security (CERBERUS) platform was a partially implemented robotic sentry with working tele-operation. This project sought to add autonomous functionality to the CERBERUS platform. This objective was partially realized in that an autonomous infrastructure was created. The framework still requires further testing due to prohibitive mechanical failures which impeded testing.

Acknowledgments

We would like to thank our advisors, Professor Stafford and Professor Wyglinski for their help and guidance throughout this project. We would have been lost without it. We would also like to thank members of last year's MQP team, namely Marissa Bennett and Jeff Tolbert, for answering our myriad of questions. Special thanks as well to Kyle Postans for volunteering your time to get involved with our project and also to Nico Machado for his help with the HERO board.

Table of Contents

List of Figures

List of Tables

Executive Summary

The United States Air Force (USAF) has many unmanned bases that require monitoring in remote areas throughout the United States. The combination of the sheer number of facilities that necessitate security and the remote nature of some facilities makes it impractical to use humans to secure these sites. This is where robotic security comes in. Having autonomous security robots deployed at these sites would allow the USAF to monitor the sites in real time without having to deploy human resources to the locations.

The CERBERUS robot was created to remedy this problem. The robot is responsible for a facility with a 1200 ft perimeter and must be able to autonomously navigate to the site of a perimeter breach within two minutes and transmit video back to United States Air Force employees. Once autonomous terminates, CERBERUS should transition from autonomous operation to teleoperation allowing USAF members to drive the platform and manipulate CERBERUS' cameras in order to perform a thorough assessment of the situation. The full sequence of robot operation is summarized below in Figure 1.

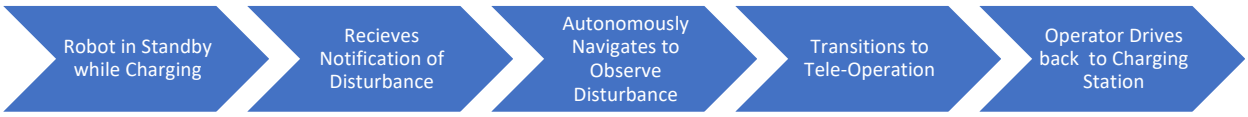

Figure 1: CERBERUS Operation Flow
This graphic illustrates the flow of events for a fully functional CERBERUS platform.

The previous team began the process of creating a robot for this application; however, only certain parts of the project have been completed. Upon inheriting this project, the robot was able to function in teleoperation mode and the cameras were able to transmit video as desired;

however, the robot lacked any autonomous capabilities. The purpose of this project was to build upon the existing platform and integrate autonomous functionality with the pre-existing tele-operation.

To that end, we created an autonomous concept of operation which defined eight observation points reached through the execution of eight distinct trajectories. The execution of one such trajectory is shown below in Figure 2. This strategy simplified autonomous functionality so that it could be implemented through a simple set of motor actions where the specific sequence is selected based upon which observation point is chosen as the destination.

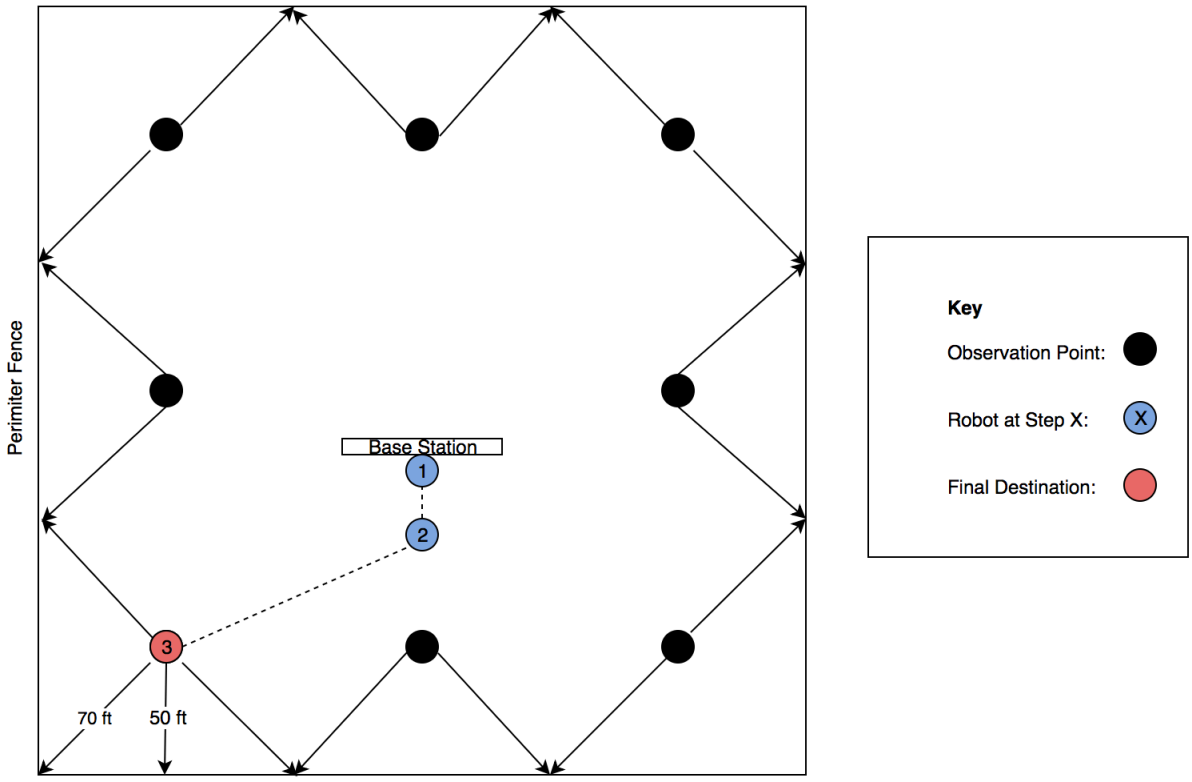

Figure 2: Autonomous Observation Points with Single Turn Trajectory
This image shows the configuration of the eight observation points along with the execution of a single-turn trajectory.

In this project we also worked with the GPS in order to gain viable location data that can be used to validate correct trajectory execution in the future. Our configuration produced a raw data stream that was accurate to the ninth decimal place, an example of which can be seen below in Table 1.

Latitude	Longitude
42.16571761	71.48010757
42.16571982	71.47999851
42.16571721	71.47999117
42.1657171	71.48001415
42.16573001	71.48001792
42.16576736	71.47993297
42.16576757	71.47992894

Table 1: Sample of GPS Data
These values are a small unfiltered subsection from the GPS output

Due to hardware failures, autonomous functionality still requires more testing at the culmination of this project. Nonetheless, this project succeeded in creating an architecture for an autonomous protocol that will be easily integrated with varied perimeter detection systems and easily improved through further sensor integration.

1. Introduction

Each year the United States Air Force issues design challenges in pursuit of innovative solutions to a variety of problems. The CERBERUS platform began as a response to one such challenge issued in 2017. This particular challenge called for the creation of a Robotic Sentry using an All-Terrain Wheelchair Platform.

1.1 The Robotic Sentry Challenge

The CERBERUS robot was created to monitor remote, unmanned Air Force facilities. According to the challenge document found in Appendix A, the robot is responsible for a facility with a 1200 ft perimeter. The CERBERUS robot must be able to autonomously navigate to the site of a perimeter breach within two minutes and transmit video back to United States Air Force employees. At this time, CERBERUS should transition from autonomous operation to teleoperation allowing USAF members to drive the platform and manipulate CERBERUS' cameras. This allows USAF personnel to perform a thorough assessment of the situation.

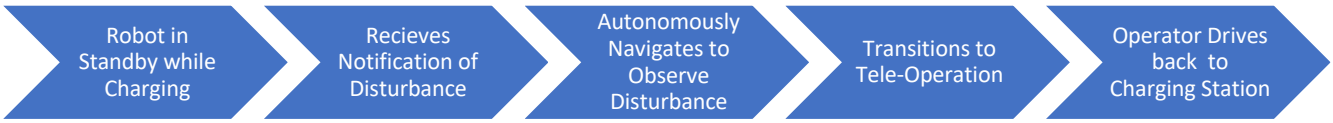

Figure 3: CERBERUS Operation Flow
This graphic illustrates the flow of events for a fully functional CERBERUS platform.

Because CERBERUS will be functioning outside in unknown conditions, the platform must be able to traverse gravel, grass, 1ft puddles and 1ft snow drifts. For storage, the robot must return

to a docking station when it is not in use in order to charge while in a standby state, ready to respond should a perimeter breach arise.

1.2 Current State of the Art

In order to execute the tasks detailed in Figure 1, the CERBERUS platform must have three key functionalities: charging, autonomy, and tele-operation. The previous team began the process of creating a robot for this application; however, they only implemented the tele-operation component. As illustrated below in Figure 4, there was still a lot of functionality to be added in order to create a fully functional platform.

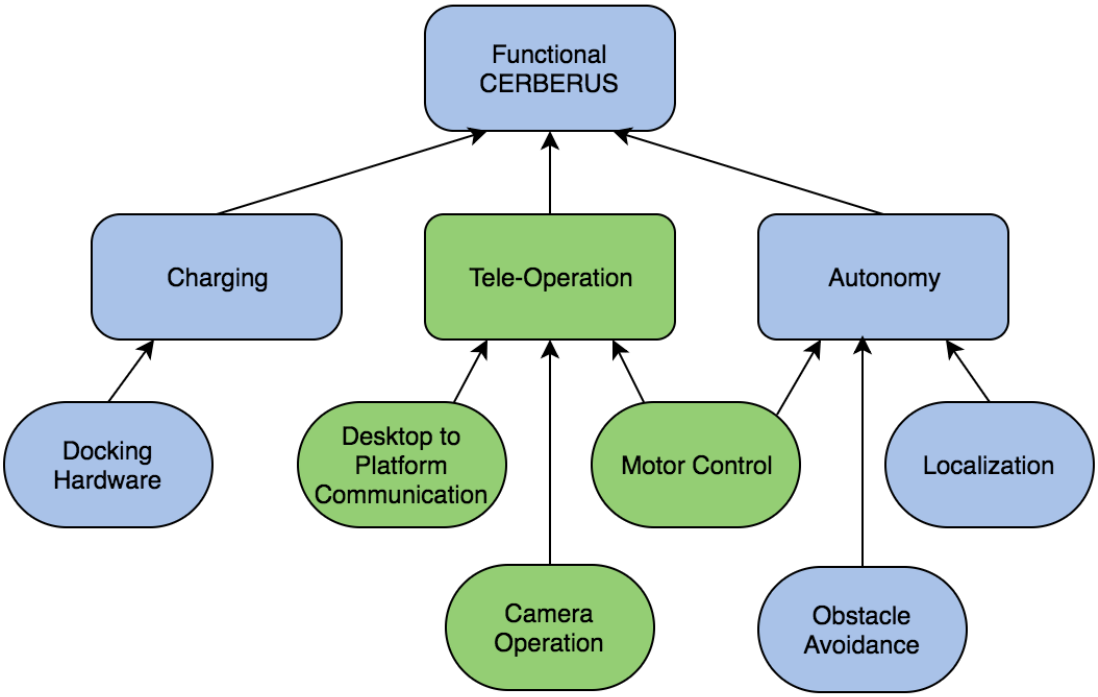

Figure 4: CERBERUS Functionalities
This diagram shows the components necessary for a fully functional platform.
The implemented functionalities are shown in green.

Upon inheriting this project, the robot was able to function in teleoperation mode and the cameras were able to transmit video as desired, but the robot had no charging or autonomous

capabilities. As a result, our project sought to implement the next large missing component: autonomy.

2. CERBERUS Crash Course

One of the first challenges of this project was to understand the existing system. In this chapter we examine strategies used in the development of platforms similar to CERBERUS. We then assess the initial CERBERUS implementation in order to understand both the strengths and weaknesses of the platform. This system appraisal helped us to understand the optimal approach to adding autonomy to the robot.

2.1 Existing Methods of Autonomous Navigation

In achieving autonomous operation for CERBERUS, we faced three main challenges. First, we had to implement some form of path planning from the platform's original location to the destination position. Then, in order to ensure accurate navigation to the desired endpoint, we needed to repeatedly localize the platform in order to verify the robot's position in relation to the destination. Lastly, we had to implement obstacle avoidance during autonomous navigation.

There are several existing autonomous path planning methodologies that could be applied to the CERBERUS platform in order to implement the path planning functionality. Many robots employ a two-fold approach creating a global path composed of waypoints and then using local cost maps to navigate between waypoints. These local cost maps can be obtained using Dijkstra's algorithm to find optimal paths (Berczi, Ostafew, Stenning, Barfoot, Jones, Tornabene, Osinski, & Daly; 2014). These types of path planning functionalities can be achieved through pre-existing ROS stacks, some of which even include obstacle avoidance (Dimitrov, Wills, & Padir; 2015). This obstacle detection has been implemented through a variety of systems ranging from LIDAR arrays to computer vision systems. In our case, this may

be an appropriate application for the WALBOT radar device. Integrating the obstacle detection with the local cost map allows for efficient navigation while avoiding obstacles along the way (Dimitrov, Wills, & Padir; 2015).

In order to accurately execute this path planning, the robot needed to be able to localize. There are several methods of localization that can be achieved without using devices such as GPS. Firstly, computer vision can be used to localize a robot when used in conjunction with distinct geometric finials. With geometric finials, there are pre-existing image processing libraries in C that can be used to generate a location and orientation of the platform relative to the finials based on the image of the distinct shapes (Gu, Ohi, Lassak, Strader, Kogan, Hypes, Harper, Hu, Gramlich, Kavi, Watson, Cheng, & Gross; 2017). However, this may pose issues when implemented in an environment where the weather is uncertain. Unfavorable visibility could result in distorted images and inaccurate position calculations. One method of improving the accuracy of position estimation based on vision is combining it with data from other onboard sensors such as encoders and IMUs (Veth & Raquet, 2007). In addition, several autonomous systems have experienced success with using ranging radios in order to estimate position (Gu et al., 2017). These radios can be placed at the robot's docking station in order to provide a consistent reference for platform localization as the robot traverses the predicted waypoints.

2.2 Existing Docking and Charging Systems

Just as there are several established methods of autonomous navigation, there are also many implementations of robot charging stations that could be used as a basis for the CERBERUS charging station design. Firstly, there are several charging systems that are completely wireless which eliminates the need for precise docking (Balakrishnan, Growthaman, Jaya Kumaran, &

Sabhapathy, 2015) (Li, 2017). However, we are required to use the two 12 V batteries supplied with the Action Trackchair and these wireless charging systems may not integrate well with our existing power system.

In the case that we need to create a physical connection, many docking stations make use of physical guidance aides in order to help the robot properly align with the docking station in order to create a high-fidelity connection (Luo, Liao, Su & Lin; 2005). In addition to physical systems to assist with alignment, some robots made use of fixed waypoints in close proximity to the docking station in order to help correctly align their platform (Berczi et al., 2015). Another option for docking alignment is the use of three-dimensional fiducials which, through vision processing, can be used to ascertain not only the distance from the docking station but also angle relative to the station as well (Dimitrov, Wills, & Padir; 2015). However, as this method is reliant upon vision processing it is also subject to disruption should visibility be limited. Another option is to make use of ranging radios for this application as well. These would be able to help the robot locate its home base and align properly so as to successfully dock with the charging station (Gu et al., 2018).

2.3 Mechanical Design Analysis

The original Action Trackchair infrastructure was modified in an attempt to optimize the chassis for the desired CERBERUS functionality. The modified mechanical system implemented on the Action Track Chair framework sought to emulate the original chassis' low center of gravity in order to retain the platform's stability and resulting effectiveness on rough terrain. In addition, the mechanical design emphasized containing all electronic components in order to protect them

from puddles or snow-drifts that the robot must navigate. With these goals in mind, the original Action Trackchair chassis was modified as seen below in Figure 5.

Figure 5: Initial CERBERUS Implementation
This image shows the CERBERUS robot as it was when we began work on the project.

The previous team removed some of the original chassis in order to reshape the robot and added steel sheeting to protect from the elements. Additionally, they moved the motors and consolidated the robot electronics into two boxes at the front and rear of the robot.

Although the original mechanical systems achieved certain criteria such as successfully protecting the robot electronics, the design fell short in several areas as well. Table 2 describes the different strengths and flaws of the original mechanical design.

Mechanical Design Strengths	Mechanical Design Flaws
• The system is stable and has a low center of gravity. It is equipped to traverse uneven terrain.	• There is no vibration damping. This causes instability for the security camera and other components.
• All electronics are self-contained in designated electronics boxes.	• The batteries are press-fit into the chassis restricting access to system electronics.
	• Steel externals are not securely attached, unnecessarily heavy, and not waterproof.
	• Robotic motion is inconsistent.
	• Internal components, such as the coolant pump, are unsecured.
	• Coolant lines inhibit access to system electronics.

Table 2: Mechanical Design Assessment
This table details the strengths and weaknesses of the initial mechanical design

The main issue with the original mechanical system was that it severely inhibited robot maintenance. Due to incorrectly estimated tolerances, the batteries were press-fit into the chassis restricting access to all of the electronic systems which were located behind the batteries. In addition, there were several components that were not mechanically stable. For one, the motors were subject to significant oscillation during robot operation. This instability impacted the reliability of robot operation. Similarly, the security camera had no stabilization which meant that during operation, the picture was highly unstable. In addition, the screws used throughout the device could not handle the vibrations experienced during operation and often fell out. As such, further improvements could be implemented in order to remedy these existing issues with the initial mechanical design.

2.4 Control Systems & Electronics Analysis

The CERBERUS robot required several independent electronic and sensing systems in order to achieve full functionality. The originally implemented electronic system can be seen below in Figure 6.

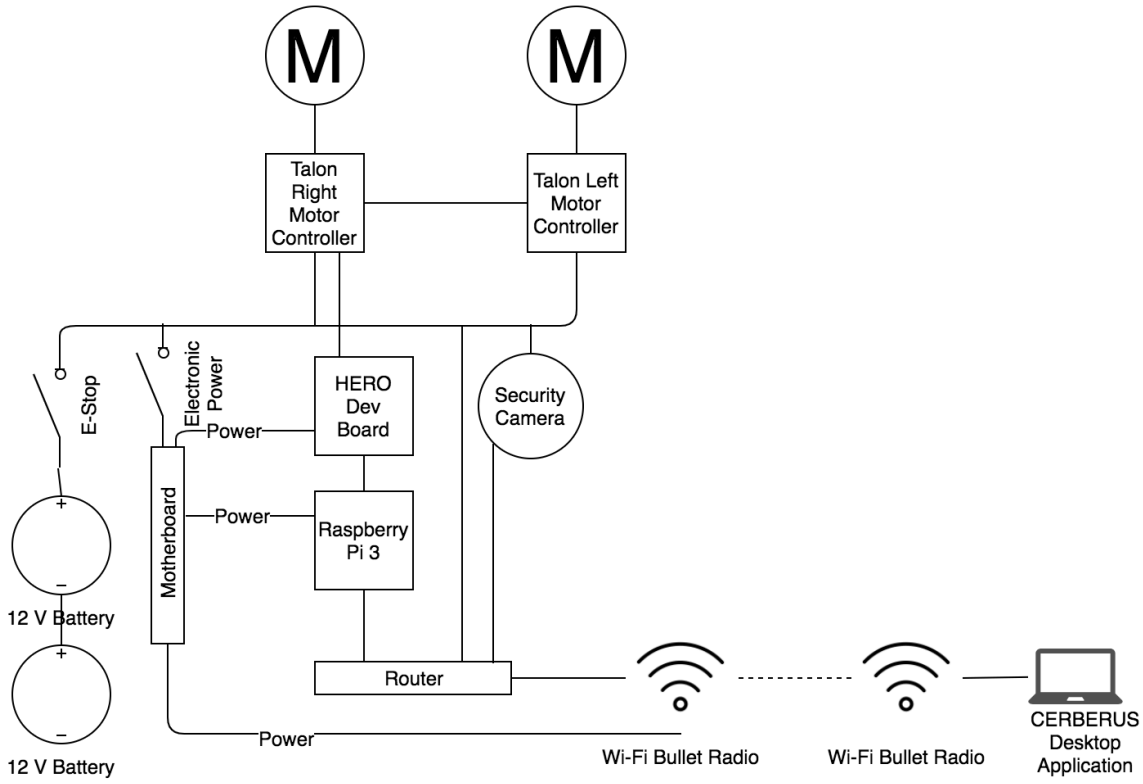

Figure 6: CERBERUS Electronics
This diagram shows the functional electronic system on the CERBERUS platform.

As with the mechanical systems, we began by taking stock of the system in order to understand its strengths and its flaws. The results of this assessment are detailed below in Table 3.

Control System Strengths	Control System Flaws
• HERO board is compatible with Talon Motor Controllers	• Daisy chained platforms lead to excessive data transmission
• E-Stop stops all systems and enables safe testing	• No sensing systems whatsoever meaning no localization or obstacle detection capabilities.
	• No reliable battery monitoring

Table 3: CERBERUS Control System Assessment
This table details the strengths and weaknesses of the original control system.

At the most basic level, CERBERUS must be able to turn on and off, run the motors, connect to the Desktop application, charge, perform health and battery monitoring, and run its sensors and cameras. The system shown above in Figure 6 was capable of turning the platform on and off, running the motors, running the security camera, and communicating with the desktop. The most significant system weakness was the complete lack of sensing capability. The platform possessed a robust sensor array; however, none of the sensing components had been integrated into the system. Table 4 below details the sensors that remained unconnected and the functionality associated with each sensor.

Sensor	Functionality
RTK GPS	Combined GPS and 9 degree of freedom IMU unit which could be used for localization.
Walabot RADAR	RADAR array for obstacle detection.
ZED Camera	Camera equipped for vision processing which could be used for obstacle detection.

Table 4: Unimplemented Sensors
This table describes the sensor that were present on the CERBERUS platform but not integrated into the existing control system.

Integrating these sensors would help remedy some of the issues we identified namely localization and obstacle detection. As a result, analyzing the potential of these sensors provided a viable avenue to begin our autonomous implementation.

Additionally, there was no system in place for charging the robot other than the provided Action Track Chair Charger. Consequently, the robot would require a charging system for its two 12

sealed lead acid (SLA) batteries though that task was outside the scope of our project. Additionally, current battery monitoring methods were unreliable and needed to be re-implemented as well.

2.5 Software Application Analysis

Several of the software components of this system had been implemented when we began work. But, there was still room for improvement. The previous group established communication between the robot platform and a simulated USAF computer. To facilitate this communication, they created the user interface application seen below in Figure 7.

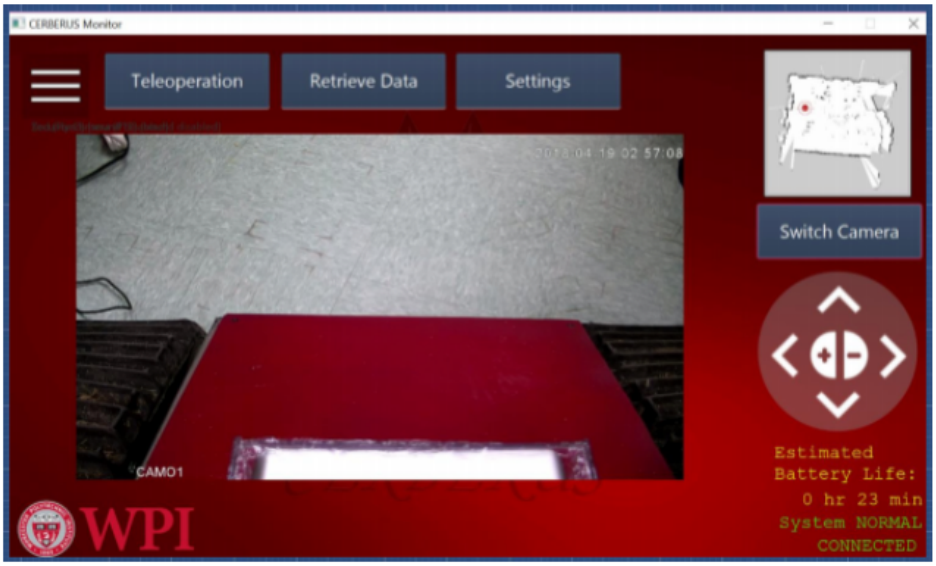

Figure 7: CERBERUS Desktop Application
This is a picture of the CERBERUS Application view for security camera manipulation.

This application allowed camera manipulation and remote robot control. As before, we assessed the system and the strengths and weaknesses of the existing software application are detailed in the following Table.

Software Application Strengths	Software Application Flaws
• Clean and easy to use	• Threads do not terminate on their own.
• Runnable on any OS	• Camera and Tele-operation cannot run concurrently due to an issue related to a shared library.
	• No functionality related to autonomy

Table 5: Desktop Software Application Assessment
This table details the strengths and flaws of the original software application.

Although the camera implementation itself was functional, the vision thread could not function concurrently with robot teleoperation. Additionally, software for autonomous operation did not exist.

2.6 Chapter Summary

The research found in this chapter helped us to understand the many different avenues we could pursue as we tried to implement autonomous operation. The subsequent analysis of the existing system allowed us to understand the resources we were working with and analyze which autonomy strategies would be most effective on this specific platform. Using this information, we created our proposed approach.

3. Proposed Approach

In tackling this project, we began by setting project goals. As the project progressed, these goals evolved to emphasize our focus on autonomy. Using these goals, we were able to understand the project scope in order to organize project execution.

3.1 Project Goals

The following details found in Table 6 constitute our initial goals for the continuing development of the CERBERUS platform. The *Must Haves* encompass the work that needs to be done in order to achieve basic robot functionality and satisfy the initial use-case. The *Nice to Haves* encompass work that we would like to do in order to not only produce a functional robot, but a robust one as well. Lastly, the *Reach Goals* represent ways the robot can be improved and augmented to go beyond simply fulfilling the original use-case. The goals in bold are associated specifically with autonomy.

Must Haves	Nice to Haves	Reach Goals
Tele-operation which allows for successful robot docking in 90% of docking attempts by a proficient robot operator	Improved mechanical design to promote ease of access for robot maintenance	Autonomous navigation home
Receive a goal position and orientation	Improved motor stability	Autonomous Docking
Autonomously navigate to within 10 ft of goal position	Sturdier screws	Redesigned robot externals
Achieve a final orientation within 45° of target orientation	Improved weatherproofing	Improved GUI
Avoid obstacles during autonomous navigation	Stabilized cameras	
Reach final position and orientation within 2 minutes	Accurate battery monitoring	
Initiate video transmission upon reaching goal position and orientation		
Transition to teleoperation upon reaching goal position and orientation		
Docking and Charging system operating while the robot is in standby mode		

Table 6: Project Goals
This table represents the initial project goals.
Goals associated with autonomy are shown in bold.

During the course of the project, these goals were amended to focus more exclusively on autonomy. Additionally, the autonomy goals shifted as we created a concrete autonomous concept of operation. The revised goals are shown below in Table 7.

Must Haves	Nice to Haves	Reach Goals
Receive a goal position	Improved mechanical design to promote ease of access for robot maintenance	Improved GUI
Autonomously navigate to within 10 ft of goal position	Improved motor stability	
Avoid obstacles during autonomous navigation	Stabilized cameras	
Reach final position and orientation within 2 minutes		
Initiate video transmission upon reaching goal position and orientation		
Transition to teleoperation upon reaching goal position and orientation		

Table 7: Revised Autonomy-Focused Project Goals
This table shows our revised list of goals which were selected in order to focus our efforts exclusively on autonomous operation.

We chose to limit our project scope to autonomous operation and to that end, we eliminated any goals that did not directly affect our autonomous implementation. Additionally, our concept of autonomous operation evolved to no longer require orientation related goals because we devised an autonomous concept that worked independently of specific disturbance locations on the perimeter. Since the robot no longer had a specific point to observe, manipulating robot orientation was no longer applicable to successful autonomous execution.

We retained several *Nice to Have and Reach* goals that are not strictly associated with autonomy; however, we thought they were appropriate to include as they were goals that affect autonomous functionality even if they are not strictly part of the autonomy protocol. For instance, although ease of access for maintenance is not outwardly associated with autonomy, working on this

aspect of the robot proved integral to our success. We had to work on this aspect of the robot in order to make the platform suitable for the development of autonomy. Similarly, the GUI may seem disjoint from autonomy, but in fact it could be modified to add functionality to the desktop application in order to send goal locations to the autonomous protocol.

3.2 Autonomous Implementation Strategy

Once we took the time to gain a comprehensive understanding of the system, we needed to identify the best approach to autonomy. As previously discussed, the CERBERUS platform provided several untapped resources in the form of different sensing systems that could be used to implement autonomy. Additionally, our background research revealed several different approaches to autonomy, many of which proved feasible options for CERBERUS.

For our autonomy we could either attempt to use the ZED Camera's vision processing capabilities, the GPS, or the IMU as the main sensing strategy for our autonomy. Obviously, the ideal solution would combine several of these options in order to produce a more robust system; however, we wanted to limit our focus to one strategy initially in order to focus our efforts in one direction. To that end, we did some preliminary work with each sensing strategy and analyzed the benefits and drawbacks of each as seen in Table 8.

Vision Processing		GPS		IMU	
Pros	**Cons**	**Pros**	**Cons**	**Pros**	**Cons**
+ Use one system for localization and obstacle avoidance	- Extremely processing heavy, potentially slow	+ Low processing load	- Requires line of site to satellite.	+ Low processing load	- Dead reckoning, meaning it cannot correct course
+ Able to self-correct	- Requires installation of onboard CPU, graphics, etc.	+ Able to self-correct		+ Easy to integrate with existing electronics	
	- Requires implementation of a whole new ROS based system	+ Easy to integrate with existing electronics			
	- Susceptible to error if there are changes in the environment, e.g. snow				

Table 8: Autonomous Sensor Comparison
This table details the benefits and drawbacks of using three different strategies for our autonomous implementation.

This analysis led us to conclude that the GPS was the most viable candidate as the primary sensing method for our autonomy. Ideally we wanted to combine GPS localization with the IMU to achieve more robust autonomy; however, we decided to focus on using the GPS initially because unlike the IMU, the GPS could compensate for error as the robot was in the progress of executing its autonomous trajectory.

3.3 Project Schedule

In order to execute this project, we created the initial project schedule seen below in the Gantt chart (Figure 8).

Task Mc	Task Name	Duration	Start	Finish	Team	Mechanical	Electrical	Software	Predece
	Project proposal	19 days	Mon 9/17/18	Thu 10/11/18	No	No	No	No	
	Background Research	5 days	Tue 9/18/18	Sun 9/23/18	Yes	No	No	No	
	Draft Background	4 days	Fri 9/21/18	Wed 9/26/18	No	No	No	No	
	Revise Background	3 days	Thu 9/27/18	Mon 10/1/18	No	No	No	No	
	Analyzing Robot Design	7 days	Tue 9/18/18	Wed 9/26/18	No	Yes	Yes	Yes	
	Draft Statement of Work	4 days	Wed 9/26/18	Sun 9/30/18	No	No	No	No	
	Revise Statement of Work	7 days	Sun 9/30/18	Sun 10/7/18	No	No	No	No	
	Compile and Revise Final Proposal	3 days	Sun 10/7/18	Tue 10/9/18	No	No	No	No	
	Find Workspace	4 days	Tue 9/18/18	Fri 9/21/18	Yes	No	No	No	
	PDR	39 days	Wed 10/24/18	Sat 12/15/18	Yes	No	No	No	
	Design Implementation	37 days	Thu 1/10/19	Fri 3/1/19	No	No	No	No	
	Testing and Validation	37 days	Tue 3/12/19	Wed 5/1/19	No	No	No	No	
	Project Report	35 days	Tue 3/12/19	Mon 4/29/19	Yes	No	No	No	

Figure 8: Project Schedule
This Gantt Chart shows the large-scale schedule created at project initialization.

3.4 Chapter Summary

By creating project goals, we were able to identify autonomy as the singular focus of our project. This provided a well-defined project scope so that we could begin analyzing different methods of executing this goal. Using our research and understanding of the platform, we identified different autonomy strategies and selected a GPS driven autonomy as the starting point of our project execution.

4. Implementation & Methodology

After researching different approaches to autonomy, we re-assessed the parameters of our problem. Within our specific system, autonomy could be simplified in order to eliminate a lot of the complexity found in traditional autonomous robotic systems. To that end, we created a new autonomy strategy.

4.1 Autonomous Operation Strategy

Knowing that the robot is operating within a 300 ft square facility was key to coming up with this new strategy to simplify our autonomous operation. The robot needed to provide a clear view of all points on the site's perimeter. The Security Camera used for perimeter observation has a nighttime range of 90 ft which means that the robot must be within 90 ft of the point of interest upon the termination of autonomous operation. Since nighttime conditions could vary, we decided to include a safety margin of 20 ft to ensure that the robot is transmitting a clear image of the disturbance. Thus, when the robot reaches the final destination of its autonomy, it needed to be at a position that is at most 70 ft from the location of the reported disturbance.

The other challenge in determining how the robot's autonomous should function was that the method of disturbance reporting was undefined. We did not know how the robot would receive the point of interest and as such, we sought to create a system that would be simple and easily integrated with any perimeter reporting system. In the interest of simplification, we identified eight distinct destinations or "observation points," seen below in Figure 9, which allow the robot to be within 70ft of all points on the perimeter fence.

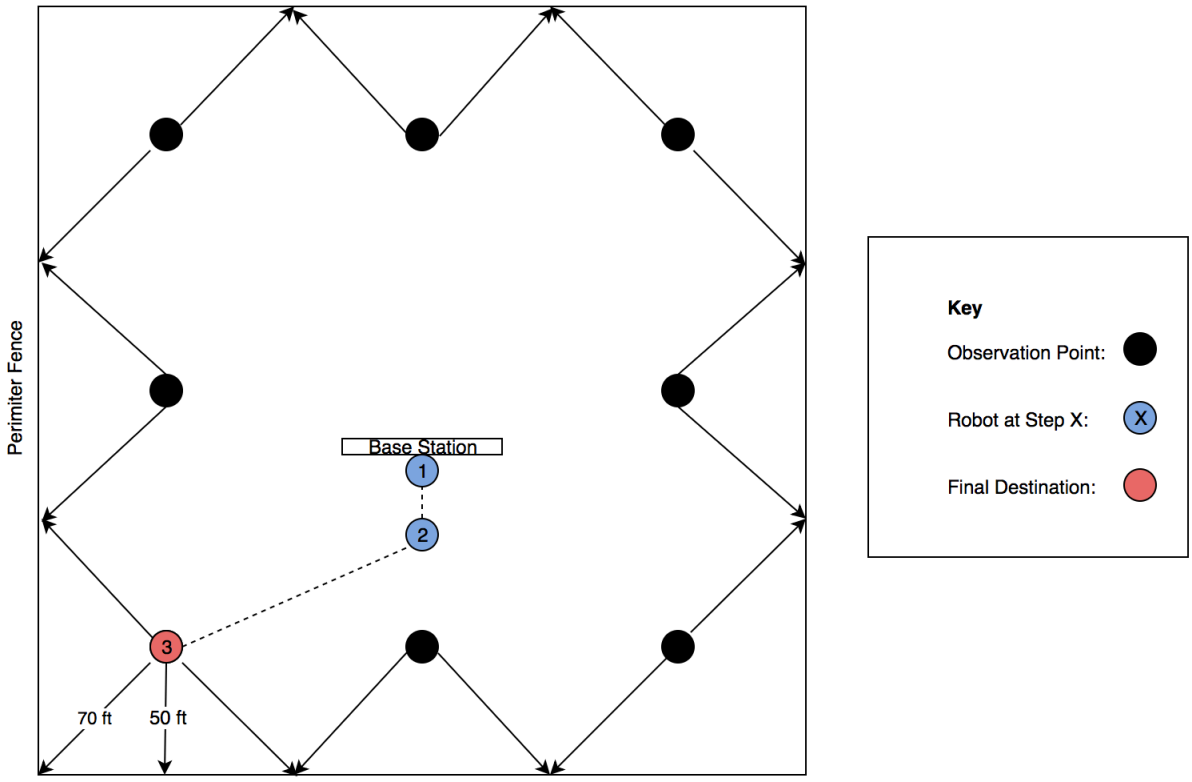

Figure 9: Autonomous Observation Points with Single Turn Trajectory
This image shows the configuration of the eight observation points along with the execution of a single-turn trajectory.

This strategy eliminates uncertainty of an unknown reporting system by assuming that the any data received will be translated into a command to go to one of the eight observation points. By identifying eight autonomous destinations, the functionality is simplified to traversing eight predefined trajectories. These trajectories can be further simplified into two general types of trajectories: single turn trajectories and multi-turn trajectories. Figure 3 shown above illustrates the execution of a single turn trajectory. In all cases the robot begins by driving straight in order to safely disengage from its charging station. In the case of a single turn trajectory, the robot then turns to the correct heading and proceeds straight until it reaches the observation point. If the robot is proceeding to one of the observation points behind the charging station, then it must

execute a slightly more complex multi-turn trajectory; however, even the multi-turn trajectories are quite simple as shown below in Figure 10.

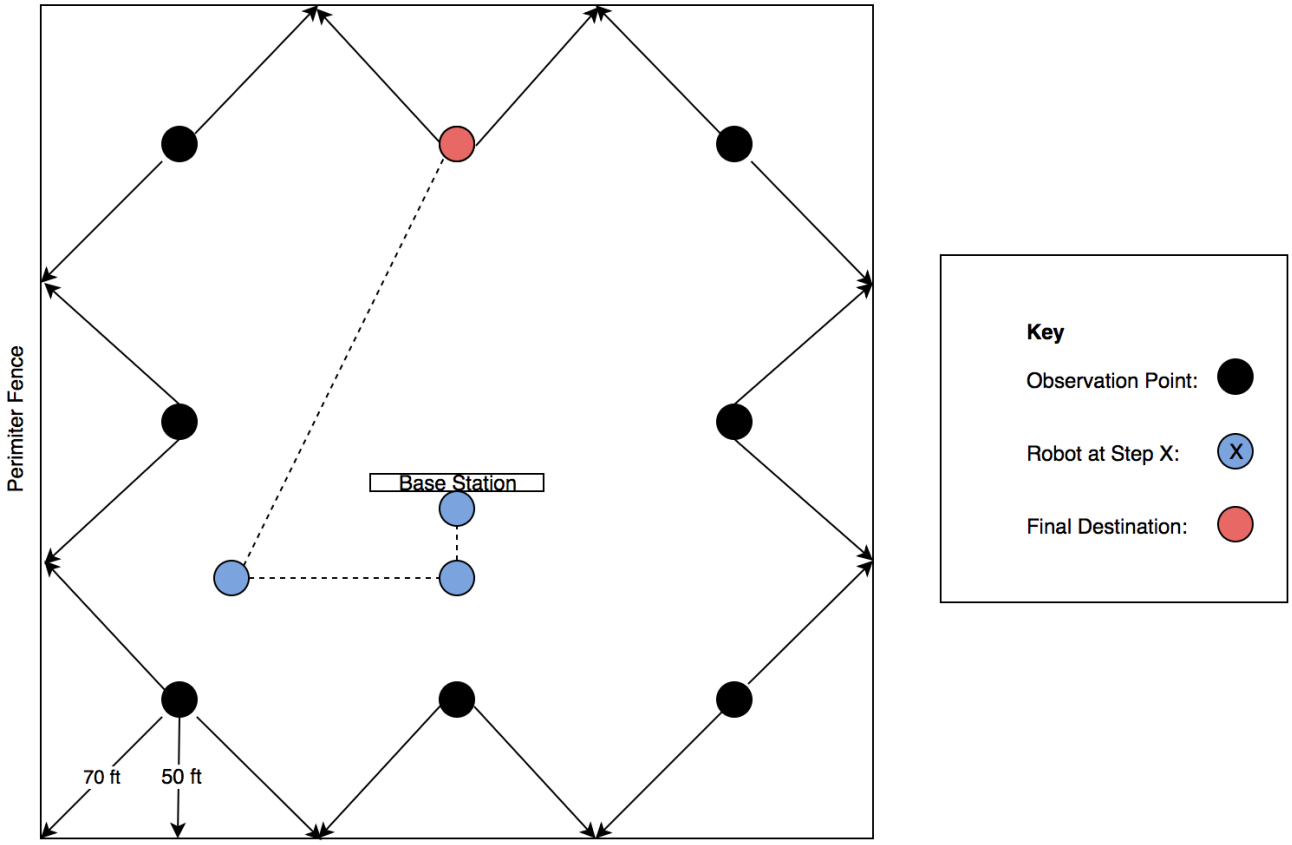

Figure 10: Multi-Turn Trajectory
This image shows the robot navigating to an observation point that requires a multi-turn trajectory.

In this case the robot drives forward as before, but then turns 90° in the appropriate direction and drives forward until it has cleared the charging station. From there the robot is able to emulate the single-turn trajectory behavior and simply turn to the correct heading and drive directly to its destination. Using these two trajectory models, the robot is able to reach all eight destinations that allow it to monitor all points on the facility's perimeter.

4.2 Autonomous Implementation

When beginning to implement the autonomous protocol, the first thing to consider was what hardware was going to be responsible for handling autonomous operation. We began by assessing the control flow of the system and identified the communication pathway observed below in Figure 11.

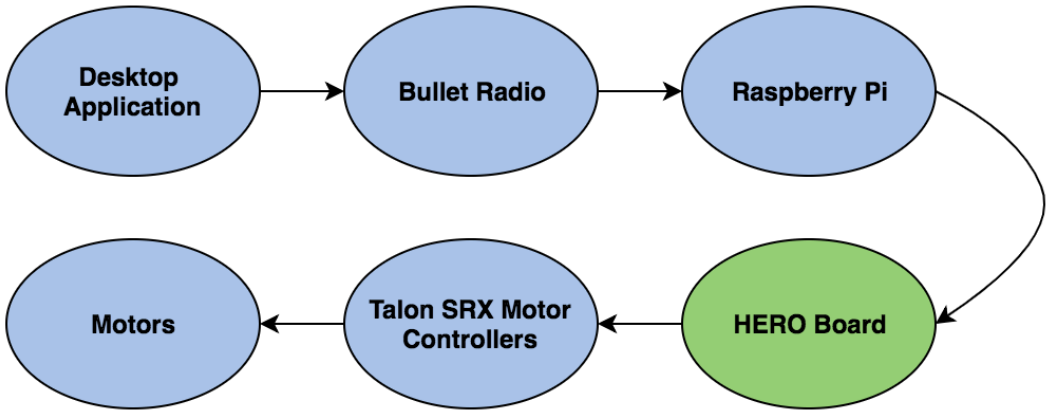

Figure 11: Motor Control Flow
This diagram shows the sequence of devices responsible for motor operation. The location selected for the autonomous code is shown in green.

The autonomy logic needed to be able to receive an activation signal from an external system in order to model response to whatever perimeter detection method the USAF decided to use. To emulate that behavior, we decided to use a signal from the desktop application to trigger autonomous operation. Outside of the activation signal though, it was important that the rest of the autonomy processing be self-contained on the platform. Since the main action of the autonomous is simply to run the motors in various ways, we wanted to optimize motor control and minimize signal transmission in order to eliminate opportunities for error. As a result, we

found that the HERO development board should be responsible for housing the autonomy logic. Since the HERO was directly connected to the motor controllers this eliminated any excess data transmission and provided both the most secure and efficient platform for autonomy.

Additionally, the HERO board was a good candidate because it was optimized for motor control using the Talon SRX motor controllers. The Talon SRX motor controllers have a C# API made for HERO development boards which made it simple to write voltages to the motors through the Talon motor controllers.

Once the correct hardware for autonomy was identified, the next question was how to organize the actual autonomy logic. Since our autonomous was made up of eight distinct actions depending upon a singular input, the implementation was essentially a large case statement as shown below in Figure 12.

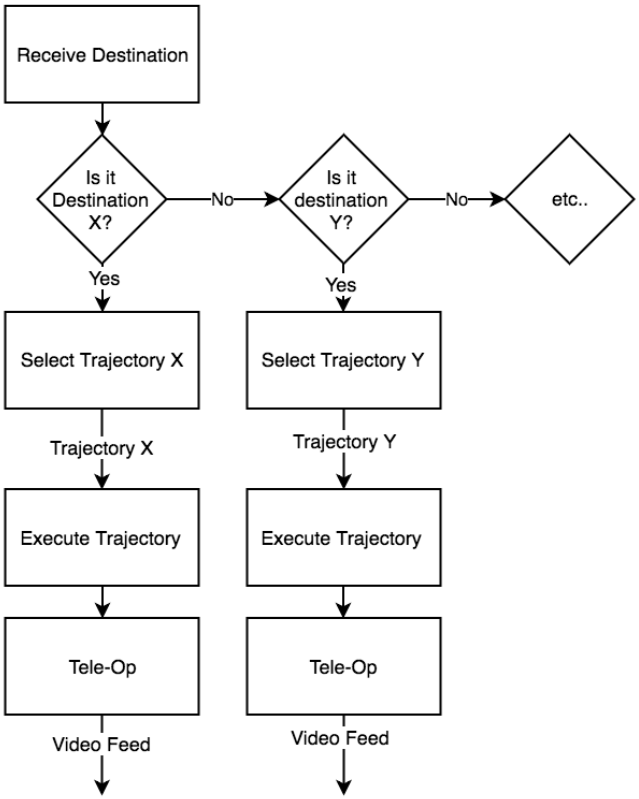

Figure 12: Autonomous Code Flow
This diagram shows the logic flow of the autonomy code.

The code has each trajectory hard-coded and simply chooses the correct sequence of motor events to execute depending upon the value of the external signal.

4.3 GPS Functionality

In creating our autonomous implementation, we wanted to use the RTK GPS module for localization and although the functionalities are not currently linked, we worked with the GPS in order to gain viable location data that can be used to validate correct trajectory execution in the future. In order to acquire raw data from the RTK GPS module we used server and client scripts written in python to establish a socket connection and receive longitude and latitude data. This

produced a raw data stream with nine decimal places worth of data, an example of which can be seen below in Table 9.

Latitude	Longitude
42.16571761	71.48010757
42.16571982	71.47999851
42.16571721	71.47999117
42.1657171	71.48001415
42.16573001	71.48001792
42.16576736	71.47993297
42.16576757	71.47992894

Table 9: Sample of GPS Data
These values are a small unfiltered subsection from the GPS output

GPS data was recorded at points approximately 50ft apart and illustrated a change on the order of .02 degrees giving the GPS a functional resolution of around .0004 degrees per ft. This resolution is sufficient to use for robot localization and can be improved through a simple data-smoothing filter. The full GPS output can be found in Appendix D.

Within the system we developed the python script to be integrated with the code on the Raspberry Pi as both systems employ python scripts. This creates the system architecture seen below in Figure 13.

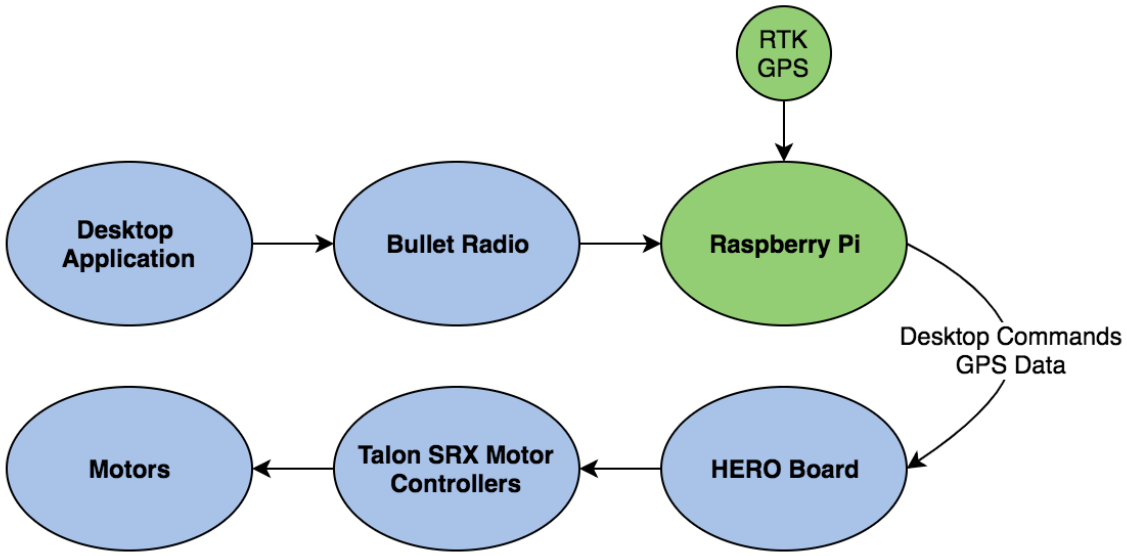

Figure 13: System Architecture with GPS
This diagram shows the robot control flow with the GPS added and the location of the addition highlighted in green

This requires that the GPS data be included in the packet sent from the Raspberry Pi to the HERO board so that the autonomy implementation on the HERO can actually use the GPS data to improve execution accuracy.

4.4 Chapter Summary

We began our implementation by defining our autonomous concept of operation which simplified our autonomous protocol to navigating to eight observation points. We then identified the correct location for the autonomous control logic and implemented our autonomous protocol on the HERO development board. Additionally, we worked with the GPS in order to acquire data to the raspberry pi. The autonomous protocol still requires testing as a broken motor controller inhibited our ability to test the implementation. However, our implementation established a framework for autonomy and data acquisition from the GPS.

5. Mechanical and Electrical Improvements

While autonomy was the main goal of this project, along the way we implemented other alterations to the robot infrastructure both as design improvements and in response to failure within the system.

5.1 Batteries

Upon initial examination of the platform it became evident that the original batteries in the system could no longer hold charge. The original system ran on two 12 V, 100Ah sealed lead acid batteries connected in series to provide a 24 V power source. These batteries were problematic not only because they no longer held a charge, but also because they were press-fit into the chassis restricting access to the electronics box, the silver box in the first image, which is situated in the empty space behind the batteries shown below in Figure 14's second image.

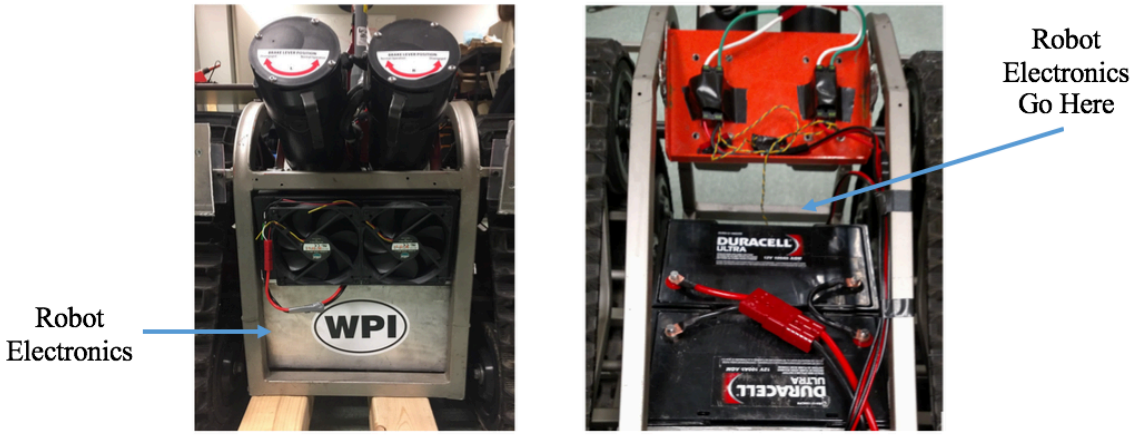

Figure 14: Batteries and Electronics Box
This image shows the location of the robot electronics and how they are inaccessible due to the location of the batteries.

This was highly problematic because all the components necessary for system development including the raspberry pi, HERO board, and RTK GPS, were completely inaccessible. In order to remedy this issue, we sought to replace the original batteries with smaller versions that would still be capable of powering the system.

In order to determine the appropriate replacement, we assessed the power requirements of the system and found that 100 Ah was much more than the system mandated. The original Action Track Chair platform was built to operate for a full day while carrying the entire weight of a human being. Because both the period of operation and physical load were significantly reduced in our case, it was clear that we could reduce the ampere-hours of the batteries. When determining the maximum power requirements, we began by defining the maximum required period of operation which we found to be an hour. The autonomous execution time is limited to 2 minutes, and we estimate that operators will not use the platform in tele-op mode for extended periods of time during an actual threat assessment; however, we allotted an hour of continued use in order to facilitate operator training sessions. Maximum current draw is encountered upon system initialization and as such Equation 1 was used in conjunction with component data sheets to find a conservative estimate of the necessary ampere-hours to be 40 Ah.

$$minimum\ Ah \approx \sum componenet\ current\ draw\ at\ initialization$$

Using this as an upper-bound and including a safety factor, we selected two 12 V, 55 Ah sealed lead acid batteries to replace the original 100 Ah set. These batteries had the added advantage of being smaller than the originals. The new batteries were 9 inches by 5.5 inches by 8.4 inches as

opposed to the originals which were significantly larger measuring 12 inches by 6.7 inches by 8.5 inches. This created much more space in the chassis and made the batteries both easily replaceable should they die again and removable to allow access to the electronics housed behind them. As before, the new batteries were wired in series to provide a 24 V source.

5.2 Wire Management for Testability

One of the difficulties of the original platform was that it was not at all configured to facilitate further development. All of the pertinent electronics were contained in a box that is not only located at the back of the robot but also restricted by coolant system that runs through the container's lid making it difficult to fully remove. In order to mitigate this inaccessibility, we added extension wires to all development components so that code can be modified and loaded to the robot without necessitating any disassembly. This also allows for concurrent code modification and testing which simplifies the development process immensely.

5.3 Coolant System Fidelity

One issue that we encountered was leaking antifreeze from the coolant pump. The pump had been secured in one orientation and when turned onto the other side it did not retain integrity. In order to combat this the source of the leak was identified and sealed. The pump was then re-secured in the orientation known to be reliable.

5.4 Chapter Summary

Through the improvements described in this chapter, namely replacing the batteries, we were able to achieve our goal of improving accessibility to the robot electronics. Although this may not have been our primary goal, this work was integral as we could not develop on the robot

without it. Additionally, these improvements will pay dividends in the future as the robot is now far better equipped for further development and maintenance.

6. Recommendations & Further Development

After our work on CERBERUS this year, the platform still requires more work in order to achieve complete functionality. In this section we detail all of the areas identified for further improvement as well as any suggestions we have come up with from our experience working with the platform this year.

6.1 Further Work on Autonomy

Our project has established a framework for autonomy; however, there is still extensive room for improvement. Some of the next major steps are outlined here:

1. Add GPS data to the data packets sent from the raspberry pi to HERO in order to incorporate it into the autonomous as a method of localization and error correction.
2. Add an IMU to the system for accurate turning.
3. Implement the Walabot RADAR or ZED camera for obstacle detection.
4. Add a menu to the desktop application that allows the user to select desired observation point for the autonomous destination.

We recommend that any future contributors purchase a new IMU unit for the robot because the existing IMU is not meant for developers and as such, it is very hard to manipulate. Purchasing an IMU unit with a C# API that could easily interface with the HERO board would be a relatively simple method of making the autonomy significantly more reliable and robust.

Taking it a step further than that, the platform could even benefit from consolidating the control logic. Currently several boards are daisy-chained together and data and commands have a long path to travel. If either the raspberry pi or the HERO could be eliminated, the platform could be significantly simplified. This would streamline data transmission and also make future development much easier.

Lastly, when implementing obstacle detection, we recommend using the WALABOT unit because that would be much straightforward to implement than the ZED camera. The WALABOT can be run from a development board whereas the ZED camera would require a massive overhaul of the entire system. The ZED camera would require the entire electronics system to be refactored and routed through an onboard computer to provide the graphics processing required by the ZED camera. In contrast, the WALABOT radar could be connected in much the same way as the GPS or IMU and would not disrupt the current system architecture.

6.2 Miscellaneous Improvements

Although our project dealt with autonomy, in working with the robot we recognized several unrelated areas that could benefit from improvement. Some of our recommendations for these improvements include:

1. Eliminate sheet-metal plating and replace it with a watertight housing. A shell made out of fiber-glass may work well.
2. Redesign the electronics box in order to make it watertight. Additionally, this box could be modified so that the coolant system attachments do not impede access to the box's internal components.

3. Add vibration damping and improve stability for components such as the security camera and the motors.
4. Secure all internal components to the chassis with dedicated mounts.
5. Secure wireless communication that is currently occurring via Wi-Fi bullet.
6. Fully implement battery estimation
7. Improve internal wire management
8. Fix the desktop application so that tele-operation and camera observation can run simultaneously.
9. Correctly terminate the threads in the desktop application.

6.3 General Recommendations

In working on this project, we encountered many unforeseen challenges, many of which resulted from a lack of understanding of the existing system. We repeatedly encountered roadblocks that were difficult to surmount because there was little to no documentation of the CERBERUS system. In order to facilitate successful development on this platform in the future, we have created the document found in Appendix B, the CERBERUS User's Guide, which should assist future developers when working with the CERBERUS platform. Hopefully this will help them to avoid the challenges that we faced.

6.4 Chapter Summary

Our work this year laid the groundwork for the autonomous component of the CERBERUS platform, but there is still extensive work to be done both on the robot's autonomy and on other systems as well. There are multiple ways to proceed from here. Future contributors could

continue with the concrete steps outlined in section 6.1 above. However, a better approach may be to refactor the system entirely. After working with CERBERUS for a year, it is clear to us that if this robot is to be a viable candidate for continued development and eventual deployment, it could benefit from an overhaul of the control system architecture with an emphasis on effective system engineering.

References

Action Trackchair. (2017). Action trackchair ST. Retrieved from http://actiontrackchair.com/trackchair-st/

Balakrishnan, M., Gowthaman, S., Jaya Kumaran, S. P., & Sabhapathy, G. R. (Mar 2015). (Mar 2015). A smart spy robot charged and controlled by wireless systems. Paper presented at the 1-4. doi:10.1109/ICIIECS.2015.7193096 Retrieved from https://ieeexplore.ieee.org/document/7193096

BCG. (2018). Global spending on military robotics from 2000 to 2025 (in billion U.S. dollars). In *Statista - The Statistics Portal*. Retrieved October 12, 2018, from https://www-statista-com.ezproxy.wpi.edu/statistics/441960/forecast-for-military-robotic-market-spending-worldwide/.

Bennett, M., Quartuccio, K., & Tolbert, J. (2018). *CERBERUS: Computer enabled robotic base enhancing remote unmanned security*

Berczi, L. P., Ostafew, C., Stenning, B., Barfoot, T. D., Jones, E., Tornabene, L., . . . Daly, M. (2014). *Place revisiting and teleoperation for a sample-return mission control architecture.* (). Retrieved from http://asrl.utias.utoronto.ca/~tdb/bib/berczi_isairas14.pdf

Dimitrov, V., Wills, M., & Padir, T. (2015). Realization of vision-based navigation and object recognition algorithms for the sample return challenge. Paper presented at the *Aerospace Conference, 2015 IEEE,* 1-8.

Gu, Y., Ohi, N., Lassak, K., Strader, J., Kogan, L., Hypes, A., Gross, J. (2018). Cataglyphis: An autonomous sample return rover. *Journal of Field Robotics, 35*(2), 248-274. doi:10.1002/rob.21737

Li, Z. (Jul 2017). (Jul 2017). Wireless charging system based on substation inspection robot. Paper presented at the 919-923. doi:10.1109/ICMIC.2017.8321587 Retrieved from https://ieeexplore.ieee.org/document/8321587

Luo, R. C., Liao, C. T., Su, K. L., & Lin, K. C. (2005). (2005). Automatic docking and recharging system for autonomous security robot. Paper presented at the 2953-2958. doi:10.1109/IROS.2005.1545197 Retrieved from https://ieeexplore.ieee.org/document/1545197

Poindexter, J. W. *Robotic sentry using all-terrain wheelchair platform*

Ravankar, A., Ravankar, A. A., Kobayashi, Y., Lv Jixin, Emaru, T., & Hoshino, Y. (Oct 2015). (Oct 2015). An intelligent docking station manager for multiple mobile service robots. Paper presented at the 72-78. doi:10.1109/ICCAS.2015.7364881 Retrieved from https://ieeexplore.ieee.org/document/7364881

Veth, M. M., & Raquet, J. (2007). Fusing low-cost image and inertial sensors for passive navigation. *Navigation, 54*(1), 11-20. doi:10.1002/j.2161-4296.2007.tb00391.x

Appendix A: AFRL Challenge Document

Robotic Sentry Using All-Terrain Wheelchair Platform

Background: The need for active security around facilities such as prisons, high value storage, missile silos, hazardous waste storage and political borders has created a demand for robots that can respond to breaches, alarms, or general inspection to reduce the workload on humans. A new generation of wheelchairs from Action Trackchair is designed for use in all terrains and offers a starting point to address a robust, robotic monitoring and response capability for these situations.

A New Design Concept: The Robotic Sentinel

Using the Action Trackchair (provided by the manufacturer) as a platform, develop a robotic sentinel that meets the security requirements of secure facilities, as indicated by the design scenario below. Design sensors, communications, control systems, programming, and systems management (power, maintenance, control) approaches that can be tested on the Trackchair as part of the project.

Scenario:
You've been asked to provide 24/7 security coverage of remote fenced in areas around an underground storage facility. The facility has two perimeter fences with a 10ft gap between them. Most of the surface between the fences is a 3ft sidewalk but there are areas that have gravel, grass and in some cases puddles up to 1ft deep. During winter months, snow drifts up to 1ft deep may be encountered. These facilities are square, 2 acre compounds which means you have a 1200ft perimeter to monitor.

There are dozens of these facilities so it is impossible to provide 24/7 human coverage of each site. You have stationary sensors and closed circuit TV so you know when there is a disturbance at each site. However, when there is an alarm you must quickly investigate the reason and if possible send a sentry to the breach within 2 minutes of an alarm to collect and provide more detailed information. The design goal is to design and build a robot that can do the job.

The robot shall be able to be stored in a standby condition in a small shelter that provides it with power, communication and shelter from the elements. Upon deployment, the sentry shall disconnect from the electrical connections, leave the shelter, proceed to a point designated before leaving the shelter via the shortest route, and position itself facing the area of interest to standby for further instruction. The approach to calculation and response to these requirements (i.e., real time onboard calculation vs. preprogrammed routes, obstacle avoidance, and location determination) will be part of the design process.

Skills Needed for This Work: (1) Basic understanding of robotics. (2) Control system theory for closed loop navigation. (3) Mechanical and Electrical design skills for mounting and constructing components. (4) Understanding of wireless audio/video data transmission and communications. (4) Test planning and testing skills to ensure capability and durability of a finished design.
Sponsor: James W. Poindexter, AFRL /RXMS, james.poindexter@us.af.mil, (937) 904- 4596

Appendix B: CERBERUS User Guide

Running the Robot

1. Connect Wi-Fi Bullet to personal computer using a POE (Power Over Ethernet) cable
2. Set your laptop's IP address to be static
3. Turn on robot (pull up the big red button on top)
4. Turn on electronics (small black button in red square case on top of the robot)

   4.1. You should see the security camera rotate as the robot initializes
5. Ping IP address 192.168.1.66 (on board wifi bullet)

   5.1. You should see more LEDs light up on the bullet when it establishes the connection
6. Login to raspberry Pi from linux terminal ($ ssh pi@192.168.1.66)

   6.1. Password: *CERB_666!*
7. Open CERBERUS UI Java Application in IntelliJ(preferably)
8. Connect controller to computer
9. Search for "getVendorId" and "getProductId" methods in CERBERUS UI code. Change the IDs there to your controller's hardware IDs
10. In "Main" class, make sure the client object is using "2000" as TCP port, so that it can match the port of "server.py" which would be running on Raspberry Pi as soon as it boots
11. Run tele-op thread

Moving the Security Camera

1. Launch the CERBERUS application

2. Click “Connect Cameras”, if it appears

3. click on the arrow keys on the right side application screen to move the camera.

Finding the Wi-Fi Bullet’s Address in Range

1. In Ubuntu Terminal use nmap command, for example :

   $ nmap 192.168.1.0-99

   or

   $ nmap 192.168.1.*

Raspberry Pi

2. From Ubuntu Terminal use the command:
   ssh pi@192.168.1.66
3. use “ls” command to see which files are present in Raspberry Pi. “server.py” is the one that runs when pi is booted.
4. server.py is run from .bashrc file. The command that runs it is located at the end of the .bashrc file. It can be viewed by “ls -a” command.

HERO Board Development

1. Install Visual Studio
2. Install .net framework extension
3. Open HERO solution file in Visual Studio
4. Plug HERO into computer via micro-usb port on the HERO
5. Modify Script
6. Run Script

6.1. This auto-uploads to the HERO board

6.2. Make sure that Visual Studio can see the HERO board by going to Project --> [project name] Properties --> .NET Micro Framework--> Device and check from the comboBox if the HERO board shows up.

Tip: Should you need to update the HERO firmware; you can do this via the normal USB port on the board. (You will need your own USB A to A cable)

Suggested Resources for HERO

1. HERO Board code examples: https://github.com/CrossTheRoadElec/Phoenix-Examples-Languages
2. HERO User's Guide: https://www.ctr-electronics.com/downloads/pdf/HERO%20User's%20Guide.pdf
3. Talon SRX User Guide: https://www.ctr-electronics.com/Talon%20SRX%20User's%20Guide.pdf
4. Talon SRX Software Reference: https://www.ctr-electronics.com/downloads/pdf/Legacy%20Talon%20SRX%20Software%20Reference%20Manual.pdf

Appendix C: Action TrackChair Information

The mechanical systems for this robot were designed using the Action Track Chair, an all-terrain wheelchair, as the base chassis (Figure 1). The Action Track Chair uses a low center of gravity to avoid tipping when traversing rough terrain. In addition, the chair has a built-in electromagnetic parking brake to ensure stability when stopping on uneven topography.

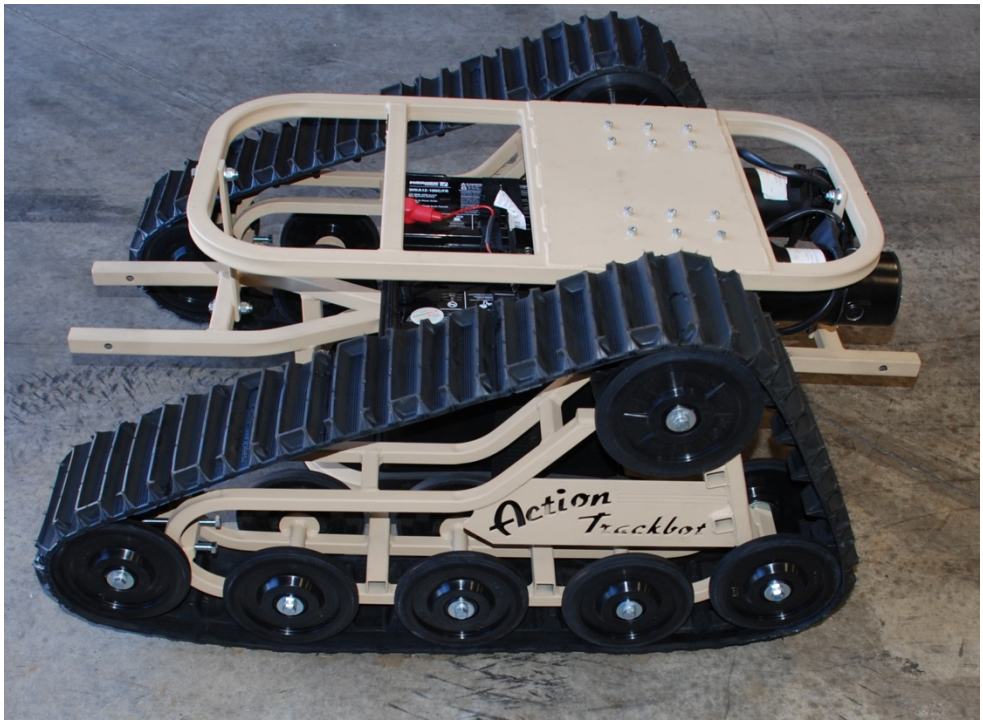

Figure 15: Un-Modified Action Trackchair Chassis

The Action Trackchair is a tracked vehicle powered by two 24V DC motors. These motors are powered by two 12V batteries connected in series creating a 24V supply. Before modifications, the Action Trackchair chassis had an estimated range of 10 miles and a maximum speed of approximately 3 mph. These specifications made the chair a suitable base for the CERBERUS platform as they allow the platform to navigate to any point in the compound in under two minutes as required by the USAF specifications.

Appendix D: GPS Log

Trial 1

No Fix
No Fix
lat = 4216.5723558 lat_dir = N lon = 07148.0138056 lon_dir = W
No Fix
No Fix
No Fix
No Fix
lat = 4216.5773254 lat_dir = N lon = 07148.0120262 lon_dir = W
No Fix
No Fix
No Fix
lat = 4216.5718360 lat_dir = N lon = 07148.0154508 lon_dir = W
No Fix
No Fix
lat = 4216.5692455 lat_dir = N lon = 07148.0169763 lon_dir = W
lat = 4216.5731713 lat_dir = N lon = 07148.0151373 lon_dir = W
lat = 4216.5732084 lat_dir = N lon = 07148.0150802 lon_dir = W
No Fix
No Fix
No Fix
No Fix
lat = 4216.5695150 lat_dir = N lon = 07148.0178784 lon_dir = W
No Fix
No Fix
lat = 4216.5733498 lat_dir = N lon = 07148.0152054 lon_dir = W
No Fix
No Fix
lat = 4216.5676211 lat_dir = N lon = 07148.0192253 lon_dir = W
lat = 4216.5733548 lat_dir = N lon = 07148.0150527 lon_dir = W
lat = 4216.5733063 lat_dir = N lon = 07148.0150381 lon_dir = W
No Fix
No Fix
lat = 4216.5664595 lat_dir = N lon = 07148.0202372 lon_dir = W
No Fix
lat = 4216.5781315 lat_dir = N lon = 07148.0134732 lon_dir = W
No Fix
lat = 4216.5781630 lat_dir = N lon = 07148.0134802 lon_dir = W
lat = 4216.5781499 lat_dir = N lon = 07148.0134930 lon_dir = W
lat = 4216.5781800 lat_dir = N lon = 07148.0134640 lon_dir = W
lat = 4216.5781531 lat_dir = N lon = 07148.0134025 lon_dir = W
lat = 4216.5782072 lat_dir = N lon = 07148.0133277 lon_dir = W

lat = 4216.5782000 lat_dir = N lon = 07148.0133342 lon_dir = W
lat = 4216.5781623 lat_dir = N lon = 07148.0134135 lon_dir = W
lat = 4216.5781623 lat_dir = N lon = 07148.0134135 lon_dir = W
lat = 4216.5781915 lat_dir = N lon = 07148.0134626 lon_dir = W
lat = 4216.5781870 lat_dir = N lon = 07148.0134251 lon_dir = W
lat = 4216.5781108 lat_dir = N lon = 07148.0134095 lon_dir = W
lat = 4216.5781324 lat_dir = N lon = 07148.0133652 lon_dir = W
lat = 4216.5782225 lat_dir = N lon = 07148.0133489 lon_dir = W
lat = 4216.5781785 lat_dir = N lon = 07148.0133878 lon_dir = W
lat = 4216.5781303 lat_dir = N lon = 07148.0134650 lon_dir = W
lat = 4216.5780850 lat_dir = N lon = 07148.0134494 lon_dir = W
lat = 4216.5781065 lat_dir = N lon = 07148.0134403 lon_dir = W
lat = 4216.5780814 lat_dir = N lon = 07148.0134317 lon_dir = W
lat = 4216.5779811 lat_dir = N lon = 07148.0135240 lon_dir = W
lat = 4216.5779338 lat_dir = N lon = 07148.0135227 lon_dir = W
lat = 4216.5779622 lat_dir = N lon = 07148.0134480 lon_dir = W
lat = 4216.5778296 lat_dir = N lon = 07148.0134862 lon_dir = W
lat = 4216.5778259 lat_dir = N lon = 07148.0134775 lon_dir = W
lat = 4216.5778064 lat_dir = N lon = 07148.0134383 lon_dir = W
lat = 4216.5778739 lat_dir = N lon = 07148.0134387 lon_dir = W
lat = 4216.5777611 lat_dir = N lon = 07148.0134923 lon_dir = W
lat = 4216.5763956 lat_dir = N lon = 07148.0150670 lon_dir = W
lat = 4216.5766056 lat_dir = N lon = 07148.0147903 lon_dir = W
lat = 4216.5766956 lat_dir = N lon = 07148.0146999 lon_dir = W
No Fix
lat = 4216.5763317 lat_dir = N lon = 07148.0150056 lon_dir = W
lat = 4216.5747538 lat_dir = N lon = 07148.0151735 lon_dir = W
lat = 4216.5750417 lat_dir = N lon = 07148.0150541 lon_dir = W
lat = 4216.5763814 lat_dir = N lon = 07148.0139432 lon_dir = W
lat = 4216.5763081 lat_dir = N lon = 07148.0143667 lon_dir = W
lat = 4216.5761979 lat_dir = N lon = 07148.0142011 lon_dir = W
lat = 4216.5778839 lat_dir = N lon = 07148.0127350 lon_dir = W
lat = 4216.5765914 lat_dir = N lon = 07148.0139219 lon_dir = W
lat = 4216.5766208 lat_dir = N lon = 07148.0140719 lon_dir = W
lat = 4216.5758517 lat_dir = N lon = 07148.0147907 lon_dir = W
lat = 4216.5777845 lat_dir = N lon = 07148.0139596 lon_dir = W
lat = 4216.5775780 lat_dir = N lon = 07148.0139805 lon_dir = W
lat = 4216.5774996 lat_dir = N lon = 07148.0138794 lon_dir = W
lat = 4216.5776243 lat_dir = N lon = 07148.0134857 lon_dir = W
lat = 4216.5779635 lat_dir = N lon = 07148.0124061 lon_dir = W
lat = 4216.5771506 lat_dir = N lon = 07148.0125684 lon_dir = W
lat = 4216.5778288 lat_dir = N lon = 07148.0123793 lon_dir = W
lat = 4216.5784743 lat_dir = N lon = 07148.0125273 lon_dir = W
lat = 4216.5782311 lat_dir = N lon = 07148.0127260 lon_dir = W
lat = 4216.5791966 lat_dir = N lon = 07148.0120588 lon_dir = W
lat = 4216.5793333 lat_dir = N lon = 07148.0119925 lon_dir = W

lat = 4216.5787676 lat_dir = N lon = 07148.0125797 lon_dir = W
lat = 4216.5793004 lat_dir = N lon = 07148.0120260 lon_dir = W
lat = 4216.5796698 lat_dir = N lon = 07148.0119877 lon_dir = W
lat = 4216.5795370 lat_dir = N lon = 07148.0120742 lon_dir = W
lat = 4216.5794273 lat_dir = N lon = 07148.0121997 lon_dir = W
lat = 4216.5790154 lat_dir = N lon = 07148.0123820 lon_dir = W
lat = 4216.5790154 lat_dir = N lon = 07148.0123820 lon_dir = W
lat = 4216.5781739 lat_dir = N lon = 07148.0128126 lon_dir = W
lat = 4216.5781459 lat_dir = N lon = 07148.0129886 lon_dir = W
lat = 4216.5782458 lat_dir = N lon = 07148.0132757 lon_dir = W
lat = 4216.5784505 lat_dir = N lon = 07148.0130583 lon_dir = W
lat = 4216.5786472 lat_dir = N lon = 07148.0131581 lon_dir = W
lat = 4216.5785678 lat_dir = N lon = 07148.0132702 lon_dir = W
lat = 4216.5789325 lat_dir = N lon = 07148.0129999 lon_dir = W
lat = 4216.5789309 lat_dir = N lon = 07148.0128599 lon_dir = W
lat = 4216.5791401 lat_dir = N lon = 07148.0128263 lon_dir = W
lat = 4216.5791550 lat_dir = N lon = 07148.0128368 lon_dir = W
lat = 4216.5790504 lat_dir = N lon = 07148.0129650 lon_dir = W
lat = 4216.5787147 lat_dir = N lon = 07148.0133201 lon_dir = W
lat = 4216.5783046 lat_dir = N lon = 07148.0136581 lon_dir = W
lat = 4216.5788511 lat_dir = N lon = 07148.0130055 lon_dir = W
lat = 4216.5794678 lat_dir = N lon = 07148.0121221 lon_dir = W
lat = 4216.5788588 lat_dir = N lon = 07148.0122693 lon_dir = W
lat = 4216.5789112 lat_dir = N lon = 07148.0122128 lon_dir = W
lat = 4216.5786550 lat_dir = N lon = 07148.0128391 lon_dir = W
lat = 4216.5788053 lat_dir = N lon = 07148.0129137 lon_dir = W
lat = 4216.5786424 lat_dir = N lon = 07148.0131828 lon_dir = W
lat = 4216.5787436 lat_dir = N lon = 07148.0133415 lon_dir = W
lat = 4216.5789610 lat_dir = N lon = 07148.0132020 lon_dir = W
lat = 4216.5788325 lat_dir = N lon = 07148.0134517 lon_dir = W
lat = 4216.5787428 lat_dir = N lon = 07148.0134965 lon_dir = W
lat = 4216.5787262 lat_dir = N lon = 07148.0134017 lon_dir = W
lat = 4216.5787262 lat_dir = N lon = 07148.0134017 lon_dir = W
lat = 4216.5783415 lat_dir = N lon = 07148.0137388 lon_dir = W
lat = 4216.5785645 lat_dir = N lon = 07148.0134051 lon_dir = W
lat = 4216.5787211 lat_dir = N lon = 07148.0132361 lon_dir = W
lat = 4216.5780617 lat_dir = N lon = 07148.0135386 lon_dir = W
lat = 4216.5784671 lat_dir = N lon = 07148.0132957 lon_dir = W
lat = 4216.5787566 lat_dir = N lon = 07148.0130590 lon_dir = W
lat = 4216.5785401 lat_dir = N lon = 07148.0134353 lon_dir = W
lat = 4216.5784899 lat_dir = N lon = 07148.0135328 lon_dir = W
lat = 4216.5778728 lat_dir = N lon = 07148.0141191 lon_dir = W
lat = 4216.5777633 lat_dir = N lon = 07148.0139128 lon_dir = W
lat = 4216.5779748 lat_dir = N lon = 07148.0136070 lon_dir = W
lat = 4216.5785698 lat_dir = N lon = 07148.0133425 lon_dir = W
lat = 4216.5782108 lat_dir = N lon = 07148.0135714 lon_dir = W

lat = 4216.5785533 lat_dir = N lon = 07148.0133681 lon_dir = W
lat = 4216.5785987 lat_dir = N lon = 07148.0131475 lon_dir = W
lat = 4216.5789803 lat_dir = N lon = 07148.0128647 lon_dir = W
lat = 4216.5792491 lat_dir = N lon = 07148.0126935 lon_dir = W
lat = 4216.5789285 lat_dir = N lon = 07148.0129550 lon_dir = W
lat = 4216.5795684 lat_dir = N lon = 07148.0122346 lon_dir = W
lat = 4216.5795812 lat_dir = N lon = 07148.0121819 lon_dir = W
lat = 4216.5794355 lat_dir = N lon = 07148.0126139 lon_dir = W
lat = 4216.5795665 lat_dir = N lon = 07148.0126973 lon_dir = W
lat = 4216.5798157 lat_dir = N lon = 07148.0125933 lon_dir = W
lat = 4216.5798173 lat_dir = N lon = 07148.0127919 lon_dir = W
lat = 4216.5800264 lat_dir = N lon = 07148.0125810 lon_dir = W
lat = 4216.5802001 lat_dir = N lon = 07148.0121757 lon_dir = W
lat = 4216.5803006 lat_dir = N lon = 07148.0122288 lon_dir = W
lat = 4216.5800362 lat_dir = N lon = 07148.0124451 lon_dir = W
lat = 4216.5799989 lat_dir = N lon = 07148.0124095 lon_dir = W
lat = 4216.5805387 lat_dir = N lon = 07148.0120859 lon_dir = W
lat = 4216.5804137 lat_dir = N lon = 07148.0122864 lon_dir = W

Trial 2

lat = 4216.5717612 lat_dir = N lon = 07148.0107573 lon_dir = W
lat = 4216.5719818 lat_dir = N lon = 07147.9998510 lon_dir = W
lat = 4216.5717212 lat_dir = N lon = 07147.9991170 lon_dir = W
lat = 4216.5717103 lat_dir = N lon = 07148.0014152 lon_dir = W
lat = 4216.5730014 lat_dir = N lon = 07148.0017919 lon_dir = W
lat = 4216.5767359 lat_dir = N lon = 07147.9932968 lon_dir = W
lat = 4216.5767566 lat_dir = N lon = 07147.9928942 lon_dir = W
lat = 4216.5766355 lat_dir = N lon = 07147.9933553 lon_dir = W
lat = 4216.5763349 lat_dir = N lon = 07147.9939599 lon_dir = W
lat = 4216.5769294 lat_dir = N lon = 07147.9922224 lon_dir = W
lat = 4216.5770171 lat_dir = N lon = 07147.9919162 lon_dir = W
lat = 4216.5771657 lat_dir = N lon = 07147.9915357 lon_dir = W
lat = 4216.5771209 lat_dir = N lon = 07147.9917117 lon_dir = W
lat = 4216.5767134 lat_dir = N lon = 07147.9927538 lon_dir = W
lat = 4216.5762603 lat_dir = N lon = 07147.9935935 lon_dir = W
lat = 4216.5765689 lat_dir = N lon = 07147.9928083 lon_dir = W
lat = 4216.5764518 lat_dir = N lon = 07147.9932466 lon_dir = W
lat = 4216.5763648 lat_dir = N lon = 07147.9928297 lon_dir = W
lat = 4216.5763304 lat_dir = N lon = 07147.9928642 lon_dir = W
lat = 4216.5762893 lat_dir = N lon = 07147.9928877 lon_dir = W
lat = 4216.5763547 lat_dir = N lon = 07147.9926076 lon_dir = W
lat = 4216.5763250 lat_dir = N lon = 07147.9927764 lon_dir = W
lat = 4216.5762900 lat_dir = N lon = 07147.9928860 lon_dir = W
lat = 4216.5762553 lat_dir = N lon = 07147.9929615 lon_dir = W
lat = 4216.5761992 lat_dir = N lon = 07147.9931893 lon_dir = W

lat = 4216.5763039 lat_dir = N lon = 07147.9930400 lon_dir = W
lat = 4216.5762560 lat_dir = N lon = 07147.9929276 lon_dir = W
lat = 4216.5762560 lat_dir = N lon = 07147.9929276 lon_dir = W
lat = 4216.5762575 lat_dir = N lon = 07147.9930395 lon_dir = W
lat = 4216.5760816 lat_dir = N lon = 07147.9929222 lon_dir = W
lat = 4216.5762778 lat_dir = N lon = 07147.9930768 lon_dir = W
lat = 4216.5759460 lat_dir = N lon = 07147.9931471 lon_dir = W
lat = 4216.5763720 lat_dir = N lon = 07147.9929778 lon_dir = W
lat = 4216.5764693 lat_dir = N lon = 07147.9928021 lon_dir = W
lat = 4216.5763227 lat_dir = N lon = 07147.9926685 lon_dir = W
lat = 4216.5765340 lat_dir = N lon = 07147.9929319 lon_dir = W
lat = 4216.5764581 lat_dir = N lon = 07147.9929708 lon_dir = W
lat = 4216.5762851 lat_dir = N lon = 07147.9932420 lon_dir = W
lat = 4216.5762693 lat_dir = N lon = 07147.9931584 lon_dir = W
lat = 4216.5762239 lat_dir = N lon = 07147.9930390 lon_dir = W
lat = 4216.5764189 lat_dir = N lon = 07147.9932455 lon_dir = W
lat = 4216.5761689 lat_dir = N lon = 07147.9933196 lon_dir = W
lat = 4216.5758055 lat_dir = N lon = 07147.9935941 lon_dir = W
lat = 4216.5756267 lat_dir = N lon = 07147.9936930 lon_dir = W
lat = 4216.5756450 lat_dir = N lon = 07147.9934538 lon_dir = W
lat = 4216.5752851 lat_dir = N lon = 07147.9937605 lon_dir = W
lat = 4216.5752752 lat_dir = N lon = 07147.9936773 lon_dir = W
lat = 4216.5754215 lat_dir = N lon = 07147.9936336 lon_dir = W
lat = 4216.5754173 lat_dir = N lon = 07147.9937703 lon_dir = W
lat = 4216.5755078 lat_dir = N lon = 07147.9936298 lon_dir = W
lat = 4216.5754614 lat_dir = N lon = 07147.9939662 lon_dir = W
lat = 4216.5754876 lat_dir = N lon = 07147.9939196 lon_dir = W
lat = 4216.5754333 lat_dir = N lon = 07147.9938942 lon_dir = W
lat = 4216.5754729 lat_dir = N lon = 07147.9938770 lon_dir = W
lat = 4216.5755463 lat_dir = N lon = 07147.9936729 lon_dir = W
lat = 4216.5756458 lat_dir = N lon = 07147.9933207 lon_dir = W
lat = 4216.5757049 lat_dir = N lon = 07147.9932708 lon_dir = W
lat = 4216.5757883 lat_dir = N lon = 07147.9934093 lon_dir = W
lat = 4216.5759376 lat_dir = N lon = 07147.9929610 lon_dir = W
lat = 4216.5758892 lat_dir = N lon = 07147.9930564 lon_dir = W
lat = 4216.5758971 lat_dir = N lon = 07147.9930128 lon_dir = W
lat = 4216.5759316 lat_dir = N lon = 07147.9929343 lon_dir = W
lat = 4216.5759218 lat_dir = N lon = 07147.9928005 lon_dir = W
lat = 4216.5759883 lat_dir = N lon = 07147.9926601 lon_dir = W
lat = 4216.5761003 lat_dir = N lon = 07147.9923984 lon_dir = W
lat = 4216.5761610 lat_dir = N lon = 07147.9922848 lon_dir = W
lat = 4216.5761143 lat_dir = N lon = 07147.9924001 lon_dir = W
lat = 4216.5762414 lat_dir = N lon = 07147.9922886 lon_dir = W
lat = 4216.5763455 lat_dir = N lon = 07147.9919938 lon_dir = W
lat = 4216.5763648 lat_dir = N lon = 07147.9918728 lon_dir = W
lat = 4216.5763109 lat_dir = N lon = 07147.9916248 lon_dir = W

lat = 4216.5763390 lat_dir = N lon = 07147.9915686 lon_dir = W
lat = 4216.5763308 lat_dir = N lon = 07147.9917103 lon_dir = W
lat = 4216.5762945 lat_dir = N lon = 07147.9919293 lon_dir = W
lat = 4216.5763780 lat_dir = N lon = 07147.9918347 lon_dir = W
lat = 4216.5764779 lat_dir = N lon = 07147.9917753 lon_dir = W
lat = 4216.5764147 lat_dir = N lon = 07147.9920205 lon_dir = W
lat = 4216.5764612 lat_dir = N lon = 07147.9917281 lon_dir = W
lat = 4216.5764525 lat_dir = N lon = 07147.9913655 lon_dir = W
lat = 4216.5764525 lat_dir = N lon = 07147.9913655 lon_dir = W
lat = 4216.5764659 lat_dir = N lon = 07147.9916458 lon_dir = W
lat = 4216.5765278 lat_dir = N lon = 07147.9915380 lon_dir = W
lat = 4216.5765793 lat_dir = N lon = 07147.9912352 lon_dir = W
lat = 4216.5765956 lat_dir = N lon = 07147.9910696 lon_dir = W
lat = 4216.5765975 lat_dir = N lon = 07147.9906785 lon_dir = W
lat = 4216.5765450 lat_dir = N lon = 07147.9907391 lon_dir = W
lat = 4216.5765028 lat_dir = N lon = 07147.9907274 lon_dir = W
lat = 4216.5765646 lat_dir = N lon = 07147.9904511 lon_dir = W
lat = 4216.5764492 lat_dir = N lon = 07147.9907317 lon_dir = W
lat = 4216.5759972 lat_dir = N lon = 07147.9911791 lon_dir = W
lat = 4216.5760715 lat_dir = N lon = 07147.9912381 lon_dir = W
lat = 4216.5760927 lat_dir = N lon = 07147.9912295 lon_dir = W
lat = 4216.5762381 lat_dir = N lon = 07147.9911489 lon_dir = W
lat = 4216.5761557 lat_dir = N lon = 07147.9913701 lon_dir = W
lat = 4216.5762266 lat_dir = N lon = 07147.9913692 lon_dir = W
lat = 4216.5761134 lat_dir = N lon = 07147.9917448 lon_dir = W
lat = 4216.5760899 lat_dir = N lon = 07147.9918391 lon_dir = W
lat = 4216.5761516 lat_dir = N lon = 07147.9915774 lon_dir = W
lat = 4216.5761706 lat_dir = N lon = 07147.9918806 lon_dir = W
lat = 4216.5763061 lat_dir = N lon = 07147.9917087 lon_dir = W
lat = 4216.5763106 lat_dir = N lon = 07147.9914263 lon_dir = W
lat = 4216.5764418 lat_dir = N lon = 07147.9912439 lon_dir = W
lat = 4216.5763903 lat_dir = N lon = 07147.9913272 lon_dir = W
lat = 4216.5763250 lat_dir = N lon = 07147.9913691 lon_dir = W
lat = 4216.5763250 lat_dir = N lon = 07147.9913691 lon_dir = W
lat = 4216.5759339 lat_dir = N lon = 07147.9914422 lon_dir = W
lat = 4216.5757987 lat_dir = N lon = 07147.9917341 lon_dir = W
lat = 4216.5756855 lat_dir = N lon = 07147.9918510 lon_dir = W
lat = 4216.5761163 lat_dir = N lon = 07147.9917244 lon_dir = W
lat = 4216.5760772 lat_dir = N lon = 07147.9915682 lon_dir = W
lat = 4216.5764068 lat_dir = N lon = 07147.9914285 lon_dir = W
lat = 4216.5763615 lat_dir = N lon = 07147.9911382 lon_dir = W
lat = 4216.5761234 lat_dir = N lon = 07147.9909071 lon_dir = W
lat = 4216.5761597 lat_dir = N lon = 07147.9910423 lon_dir = W
lat = 4216.5766315 lat_dir = N lon = 07147.9911055 lon_dir = W
lat = 4216.5766877 lat_dir = N lon = 07147.9907939 lon_dir = W
lat = 4216.5768022 lat_dir = N lon = 07147.9908187 lon_dir = W

lat = 4216.5763000 lat_dir = N lon = 07147.9910361 lon_dir = W
lat = 4216.5763324 lat_dir = N lon = 07147.9911827 lon_dir = W
lat = 4216.5764870 lat_dir = N lon = 07147.9918293 lon_dir = W
lat = 4216.5765820 lat_dir = N lon = 07147.9917832 lon_dir = W
lat = 4216.5770801 lat_dir = N lon = 07147.9918097 lon_dir = W
lat = 4216.5773653 lat_dir = N lon = 07147.9917430 lon_dir = W
lat = 4216.5768994 lat_dir = N lon = 07147.9919804 lon_dir = W
lat = 4216.5770487 lat_dir = N lon = 07147.9921129 lon_dir = W
lat = 4216.5775855 lat_dir = N lon = 07147.9923027 lon_dir = W
lat = 4216.5772762 lat_dir = N lon = 07147.9922716 lon_dir = W
lat = 4216.5765897 lat_dir = N lon = 07147.9918686 lon_dir = W
lat = 4216.5764288 lat_dir = N lon = 07147.9919714 lon_dir = W
lat = 4216.5760872 lat_dir = N lon = 07147.9918855 lon_dir = W
lat = 4216.5760872 lat_dir = N lon = 07147.9918855 lon_dir = W
lat = 4216.5761125 lat_dir = N lon = 07147.9917615 lon_dir = W
lat = 4216.5758153 lat_dir = N lon = 07147.9919401 lon_dir = W
lat = 4216.5758710 lat_dir = N lon = 07147.9923521 lon_dir = W
lat = 4216.5761445 lat_dir = N lon = 07147.9924277 lon_dir = W
lat = 4216.5761315 lat_dir = N lon = 07147.9925532 lon_dir = W
lat = 4216.5762356 lat_dir = N lon = 07147.9927303 lon_dir = W
lat = 4216.5765122 lat_dir = N lon = 07147.9924856 lon_dir = W
lat = 4216.5766995 lat_dir = N lon = 07147.9923336 lon_dir = W
lat = 4216.5766834 lat_dir = N lon = 07147.9923447 lon_dir = W
lat = 4216.5766452 lat_dir = N lon = 07147.9925378 lon_dir = W
lat = 4216.5765590 lat_dir = N lon = 07147.9929110 lon_dir = W
lat = 4216.5766608 lat_dir = N lon = 07147.9934666 lon_dir = W
lat = 4216.5765226 lat_dir = N lon = 07147.9938165 lon_dir = W
lat = 4216.5765685 lat_dir = N lon = 07147.9941778 lon_dir = W
lat = 4216.5765731 lat_dir = N lon = 07147.9943074 lon_dir = W
lat = 4216.5764744 lat_dir = N lon = 07147.9945365 lon_dir = W
lat = 4216.5765588 lat_dir = N lon = 07147.9948990 lon_dir = W
lat = 4216.5765455 lat_dir = N lon = 07147.9951922 lon_dir = W
lat = 4216.5766851 lat_dir = N lon = 07147.9952480 lon_dir = W
lat = 4216.5767156 lat_dir = N lon = 07147.9952657 lon_dir = W
lat = 4216.5767543 lat_dir = N lon = 07147.9952268 lon_dir = W
lat = 4216.5769574 lat_dir = N lon = 07147.9951451 lon_dir = W
lat = 4216.5768450 lat_dir = N lon = 07147.9948342 lon_dir = W
lat = 4216.5769981 lat_dir = N lon = 07147.9951617 lon_dir = W
lat = 4216.5770013 lat_dir = N lon = 07147.9953071 lon_dir = W
lat = 4216.5770013 lat_dir = N lon = 07147.9953071 lon_dir = W
lat = 4216.5772182 lat_dir = N lon = 07147.9948779 lon_dir = W
lat = 4216.5769770 lat_dir = N lon = 07147.9951118 lon_dir = W
lat = 4216.5768481 lat_dir = N lon = 07147.9950314 lon_dir = W
lat = 4216.5766088 lat_dir = N lon = 07147.9953099 lon_dir = W
lat = 4216.5766241 lat_dir = N lon = 07147.9954440 lon_dir = W
lat = 4216.5767512 lat_dir = N lon = 07147.9956462 lon_dir = W

lat = 4216.5767648 lat_dir = N lon = 07147.9957746 lon_dir = W
lat = 4216.5766633 lat_dir = N lon = 07147.9958444 lon_dir = W
lat = 4216.5767568 lat_dir = N lon = 07147.9958840 lon_dir = W
lat = 4216.5765989 lat_dir = N lon = 07147.9958395 lon_dir = W
lat = 4216.5768795 lat_dir = N lon = 07147.9958634 lon_dir = W
lat = 4216.5769287 lat_dir = N lon = 07147.9959945 lon_dir = W
lat = 4216.5771367 lat_dir = N lon = 07147.9958827 lon_dir = W
lat = 4216.5770835 lat_dir = N lon = 07147.9962340 lon_dir = W
lat = 4216.5770755 lat_dir = N lon = 07147.9964946 lon_dir = W
lat = 4216.5770901 lat_dir = N lon = 07147.9963891 lon_dir = W
lat = 4216.5772452 lat_dir = N lon = 07147.9967194 lon_dir = W
lat = 4216.5772464 lat_dir = N lon = 07147.9966326 lon_dir = W
lat = 4216.5775564 lat_dir = N lon = 07147.9967022 lon_dir = W
lat = 4216.5777259 lat_dir = N lon = 07147.9965079 lon_dir = W
lat = 4216.5779436 lat_dir = N lon = 07147.9965989 lon_dir = W
lat = 4216.5776832 lat_dir = N lon = 07147.9968470 lon_dir = W
lat = 4216.5775928 lat_dir = N lon = 07147.9968012 lon_dir = W
lat = 4216.5775928 lat_dir = N lon = 07147.9968012 lon_dir = W
lat = 4216.5770542 lat_dir = N lon = 07147.9968118 lon_dir = W
lat = 4216.5771359 lat_dir = N lon = 07147.9971191 lon_dir = W
lat = 4216.5767691 lat_dir = N lon = 07147.9974318 lon_dir = W
lat = 4216.5765898 lat_dir = N lon = 07147.9974948 lon_dir = W
lat = 4216.5757480 lat_dir = N lon = 07147.9977214 lon_dir = W
lat = 4216.5756693 lat_dir = N lon = 07147.9977314 lon_dir = W
lat = 4216.5761917 lat_dir = N lon = 07147.9976945 lon_dir = W
lat = 4216.5759213 lat_dir = N lon = 07147.9978763 lon_dir = W
lat = 4216.5760859 lat_dir = N lon = 07147.9978638 lon_dir = W
lat = 4216.5760352 lat_dir = N lon = 07147.9975245 lon_dir = W
lat = 4216.5759800 lat_dir = N lon = 07147.9975292 lon_dir = W
lat = 4216.5764215 lat_dir = N lon = 07147.9975879 lon_dir = W
lat = 4216.5763464 lat_dir = N lon = 07147.9978543 lon_dir = W
lat = 4216.5762926 lat_dir = N lon = 07147.9977954 lon_dir = W
lat = 4216.5766603 lat_dir = N lon = 07147.9977788 lon_dir = W
lat = 4216.5765522 lat_dir = N lon = 07147.9976846 lon_dir = W
lat = 4216.5765518 lat_dir = N lon = 07147.9976094 lon_dir = W
lat = 4216.5767287 lat_dir = N lon = 07147.9977731 lon_dir = W
lat = 4216.5767388 lat_dir = N lon = 07147.9981536 lon_dir = W
lat = 4216.5765318 lat_dir = N lon = 07147.9982637 lon_dir = W
lat = 4216.5765734 lat_dir = N lon = 07147.9983014 lon_dir = W
lat = 4216.5764871 lat_dir = N lon = 07147.9981519 lon_dir = W
lat = 4216.5765896 lat_dir = N lon = 07147.9983578 lon_dir = W
lat = 4216.5763138 lat_dir = N lon = 07147.9982814 lon_dir = W
lat = 4216.5763138 lat_dir = N lon = 07147.9982814 lon_dir = W
lat = 4216.5764157 lat_dir = N lon = 07147.9983354 lon_dir = W
lat = 4216.5764565 lat_dir = N lon = 07147.9987102 lon_dir = W
lat = 4216.5762172 lat_dir = N lon = 07147.9989199 lon_dir = W

lat = 4216.5763888 lat_dir = N lon = 07147.9989562 lon_dir = W
lat = 4216.5766894 lat_dir = N lon = 07147.9990603 lon_dir = W
lat = 4216.5764795 lat_dir = N lon = 07147.9991919 lon_dir = W
lat = 4216.5764020 lat_dir = N lon = 07147.9990838 lon_dir = W
lat = 4216.5764036 lat_dir = N lon = 07147.9992482 lon_dir = W
lat = 4216.5766131 lat_dir = N lon = 07147.9994711 lon_dir = W
lat = 4216.5764230 lat_dir = N lon = 07147.9996004 lon_dir = W
lat = 4216.5767822 lat_dir = N lon = 07147.9993498 lon_dir = W
lat = 4216.5768577 lat_dir = N lon = 07147.9995489 lon_dir = W
lat = 4216.5769741 lat_dir = N lon = 07147.9991748 lon_dir = W
lat = 4216.5770841 lat_dir = N lon = 07147.9993352 lon_dir = W
lat = 4216.5773696 lat_dir = N lon = 07147.9990953 lon_dir = W
lat = 4216.5775555 lat_dir = N lon = 07147.9993077 lon_dir = W
lat = 4216.5775611 lat_dir = N lon = 07147.9994075 lon_dir = W
lat = 4216.5776798 lat_dir = N lon = 07147.9992853 lon_dir = W
lat = 4216.5778219 lat_dir = N lon = 07147.9991817 lon_dir = W
lat = 4216.5777007 lat_dir = N lon = 07147.9991704 lon_dir = W
lat = 4216.5777741 lat_dir = N lon = 07147.9992102 lon_dir = W
lat = 4216.5778935 lat_dir = N lon = 07147.9994255 lon_dir = W
lat = 4216.5778019 lat_dir = N lon = 07147.9996942 lon_dir = W
lat = 4216.5777642 lat_dir = N lon = 07147.9999193 lon_dir = W
lat = 4216.5778281 lat_dir = N lon = 07147.9999543 lon_dir = W
lat = 4216.5778281 lat_dir = N lon = 07147.9999543 lon_dir = W
lat = 4216.5781142 lat_dir = N lon = 07148.0001182 lon_dir = W
lat = 4216.5781701 lat_dir = N lon = 07148.0002133 lon_dir = W
lat = 4216.5785542 lat_dir = N lon = 07148.0002435 lon_dir = W
lat = 4216.5786673 lat_dir = N lon = 07148.0004131 lon_dir = W
lat = 4216.5787414 lat_dir = N lon = 07148.0003432 lon_dir = W
lat = 4216.5789685 lat_dir = N lon = 07148.0002353 lon_dir = W
lat = 4216.5789515 lat_dir = N lon = 07148.0001067 lon_dir = W
lat = 4216.5789772 lat_dir = N lon = 07148.0001753 lon_dir = W
lat = 4216.5788072 lat_dir = N lon = 07148.0004669 lon_dir = W
lat = 4216.5727233 lat_dir = N lon = 07147.9864123 lon_dir = W
lat = 4216.5726440 lat_dir = N lon = 07147.9864356 lon_dir = W
lat = 4216.5726902 lat_dir = N lon = 07147.9865413 lon_dir = W
lat = 4216.5725558 lat_dir = N lon = 07147.9868216 lon_dir = W
lat = 4216.5724140 lat_dir = N lon = 07147.9869739 lon_dir = W
lat = 4216.5723882 lat_dir = N lon = 07147.9871499 lon_dir = W
lat = 4216.5724913 lat_dir = N lon = 07147.9874928 lon_dir = W