4-2010

Slip detection and compensation in autonomous tracked micro-vehicles.

Aaron T. Stewart
University of Louisville

SLIP DETECTION AND COMPENSATION IN AUTONOMOUS TRACKED MICRO-VEHICLES

By

Aaron T. Stewart

B.S. University of Louisville

A Thesis

Submitted to the Faculty of the

University of Louisville

J. B. Speed School of Engineering

as Partial Fulfillment of the Requirements

for the Professional Degree

MASTER OF ENGINEERING

Department of Mechanical Engineering

April 2010

SLIP DETECTION AND COMPENSATION IN TRACKED AUTONOMOUS MICRO-VEHICLES

Submitted by: _____

Aaron T. Stewart

A Thesis Approved on

_____

by the Following Thesis Committee:

_____

Thesis Director, Dr. Christopher Richards

_____

Dr. William Hnat

_____

Dr. Tamer Inanc

Acknowledgements

I would like to thank Dr. Chris Richards for his continued commitment to me and the completion of this work, along with the faculty and staff of the University of Louisville who helped me solve problems along the way. Lastly I would like to thank my girlfriend and my family for their love and support and for keeping me focused on the task at hand.

ABSTRACT

When tracked vehicles traverse terrain such as sand, soil, or even concrete, they may encounter a variance in density or viscosity of the medium that the vehicle is traveling along. When this happens, one track begins to move faster or slower relative to the ground than its counterpart, causing a change in its orientation and position. Autonomous tracked vehicles must be able to detect how much change occurred in the orientation and position of the vehicle and it must then determine a new path to reach the target location.

This paper focuses on development of the ability for a small tracked vehicle to detect when a slip has occurred, how much the vehicle has slipped and how the autonomous vehicle should correct for the slip that has occurred. Three different algorithms are tested, the Straight Line Slip Method, the Arc Extension Method and the Arc Compensation Method. The Arc Compensation Method returned the best and most predictable results. The Arc Compensation Method averaged 60 mm to the target location from where the vehicle stopped the smallest of the three methods. This method also maintained a smaller standard deviation and range for the distance to the target location than the other two methods.

Table of Contents

Nomenclature

CEP	Circular Error Probable
2DRMS	Distance Root Mean Squared
COR	Center of Rotation
D_{pt}	Distance past threshold, distance traveled while slipping only after crossing the slip threshold, in *x* direction
D_s	Distance to slip in *x* direction
D_{st}	Slip travel in *x* direction
D_t	Total distance traveled in *x* direction
D_{thresh}	Distance traveled prior to crossing the slip threshold
L	Length of the arc traveled during the slip
L_p	Partial arc length, length of the arc after crossing the slip threshold
r	Radius
β	Angle of change for the slipping arc
Δx	Change in *x* direction
Δy	Change in *y* direction
ε_x	Distance traveled in the *x* direction while slipping
ε_y	Distance traveled in the y direction while slipping
θ	Change in orientation of the vehicle during the slip
θ_t	Slip threshold

List of Tables

List of Figures

I. INTRODUCTION

Research into autonomous navigation will lead to fewer traffic accidents, faster traveling times, and more efficient highway systems (Thurn, Junior: The Stanford Entry in the Urban Grand Challenge, 2007). On a smaller scale, autonomous robots could perform hazardous missions for the military such as delivering troop supplies to hostile areas, evacuating injured troops and civilians, or even patroling enemy territory. Civilian applications include autonomous mapping of hazardous areas such as mines or caves, factory material management, and search and rescue missions during natural or terrorist induced disasters.

A large amount of research has already been performed in the area of autonomous robots. One of the most well known ventures into this subject matter is the DARPA Grand Challenge. This program was launched by the Defense Advanced Research Project Agency to motivate research into unmanned ground vehicle navigation. The Challenge goal was to build an autonomous robot that is capable of covering a vast distance of rough terrain that had not been previously navigated by the robot. The group that would cross the finish line first would receive a prize worth $1 Million. The first challenge took place on March 13, 2004 and required vehicles with no drivers to travel 142 miles

across the Mojave Dessert in less than 10 hours. During the first challenge, 15 teams were selected to compete, yet none of the teams we able to navigate through more than the first 5% of the course. In 2005, the race was set to take place again. In this year, 23 teams raced across a new course and five of the teams were able to finish (Thurn, Stanley: The Robot that won the DARPA Grand Challenge, 2006).

Another popular robot design competition is the RoboCup World Cup, a competition in which designers develop robots that are capable of playing soccer (The Robocup Federation, 2010). In 2001, the organization developed RoboCup Rescue as a means to increase the awareness of the challenges involved in search and rescue operations. RoboCup Rescue provides an objective evaluation of a robot's design and strategy to negotiate an environment and gives researchers a chance to collaborate. At these events, the robots demonstrate their ability to plan and map their paths, overcome obstacles, and search for victims in simulated disaster environments. Yearly competitions provide direct comparisons of different approaches and stimulate steady advancement in robotic technology that will ultimately save lives (The Robocup Federation, 2010).

Mobile robots must be able to navigate autonomously in any environment to be useful in modern society. One of the key technologies for achieving this is map building and localization. Although GPS is a useful technology that is capable of providing an absolute position and can be used for mobile robot navigation, it is sometimes difficult to estimate precisely an exact location during

outdoor navigation. Due to the availability of signals received from global positioning satellites, the accuracy of measuring the robots location by GPS may vary. To improve the accuracy, simultaneous localization and mapping (SLAM) algorithms are designed for mobile robot navigation. These robots require sensors that are able to detect objects and determine orientation. These systems develop errors when placed in rough terrain due to slipping on obstacles such as loose dirt or pebble. When this happens, a robot will lose its orientation due to accumulated error and ultimately not end up at their target location.

The literature that has been reviewed focused on a variety of topics such as robotic competitions, robotic space endeavors, and the use of autonomous heavy machinery. However, the main focus of the research is on current techniques in navigation, sensors and strategies used to navigate using various sensors. From the information obtained, ideas were generated as to how to enable the robot to perceive its surroundings and determine its absolute position. For modern autonomous robots, there are three major methods that are being implemented as a means to determine the absolute location of the robot. These methods are inertial navigation, vision based navigation such as simultaneous localization and mapping and lastly, odometry based methods

Inertial Navigation Systems (INS) rely heavily in the use of accelerometers and gyroscopes as a means to determine position. These systems use dynamic equations of motion to calculate where a robot should be relative to its starting position. This is accomplished by measuring the acceleration on the X, Y and Z axis and integrating this information into velocity and position data. Yang and

Jianmin discuss the merits and limitations of using Global Positioning Satellites and Inertial Navigation Systems (Yang & Jianmin, 2007). They present a data analysis technique that combines the use of fuzzy logic and a Kalman filter as a way to combine the data from multiple sensors, detect errors and present the information into a coherent matrix that a computer can use to decipher the data to determine the position of the robot or any other vehicle type. Inertial navigation systems experience large amounts of error due to large amounts of noise in the sensors and from the low resolution of GPS. The majority of the research performed in the area of inertial navigation systems focuses on methods to filter the noise with the creation of adaptive Kalman filters and better fuzzy logic control systems (Bian, Jin, & Tian, 2005).

A notable disadvantage of Kalman filters and fuzzy logic control systems is that the data is processed and then discarded. The GraphSLAM method offers a way to remember where a robot has been and build a map from this information (Thurn & Montemerlo, GraphSLAM Algorithm with Applications to Large Scale Mapping of Urban Structures, 2005). SLAM stands for simultaneous localization and mapping and SLAM algorithms use sensors to perceive the environment around the mobile robot. The types of sensors used vary but generally include cameras, laser interfaces, radar, and GPS (Thurn & Montemerlo, GraphSLAM Algorithm with Applications to Large Scale Mapping of Urban Structures, 2005). GraphSLAM applications include the DARPA Grand Challenge where autonomous vehicles used are to create a digital picture of the robots immediate surroundings so that the robot can use that information to determine how to avoid

obstacles, how fast to go, and how to get to its destination. Figure 1 is visualization of the GraphSLAM algorithm.

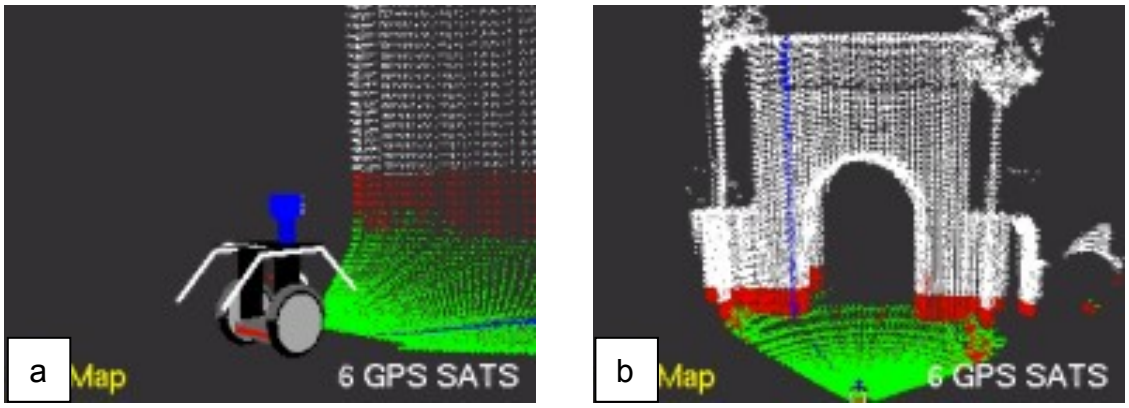

FIGURE 1. (a) An example of GraphSLAM as viewed from the side. (b) An Example of GraphSLAM as viewed by a robot (Thurn & Montemerlo, GraphSLAM Algorithm with Applications to Large Scale Mapping of Urban Structures, 2005).

Odometry based systems, also known as “dead reckoning” systems, are very reliable and accurate on smooth, flat terrain and over short distances. However, when traversing rough terrain, especially soft surfaces like sand or gravel in which the terrain itself can move, dead reckoning based systems are typically not considered useful because wheels slip and the measured rotation does not accurately reflect the distance truly traveled (Ojeda, Cruz, Reina, & Borenstein, 2006). Much research is being conducted in this field to improve the accuracy and simplify the systems that are being implemented on various platforms.

Mobile robots are increasingly being used to explore rough terrain situations such as planetary exploration and military applications. Current control and localization algorithms are not well suited for rough terrain and do not consider the physical characteristics of the vehicle and its environment. Little research has

been performed on the effects of wheel slip which will affect odometry accuracy, traction and performance and could lead to the failure of a localization routine and ultimately the failure of the robot to meet its objective (Reina, Ojeda, Milella, & Borenstein, 2006). Current methods to detect wheel slippage make use of encoders, gyroscopes and a current indicator to monitor the speed of rotation in the wheels and the resistance incurred (Ojeda, Cruz, Reina, & Borenstein, 2006).

To many, the obvious solution to keep track of the exact or absolute position of an autonomous vehicle is through the use of GPS. However, GPS has many limitations; the most relevant for this application is the precision of the GPS. GPS manufacturers us a statistic known as CEP or Circular Error Probable and are usually tested under ideal conditions (Earth Measurement Consulting). The CEP is the radius of the circle that will contain approximately 50 percent of the position measurements reported by the GPS receiver. This also means that 50% of the positions reported by a GPS will be outside of this circle. Another way to measure the accuracy of the GPS is the Distance Root Mean Squared method (2DRMS). 2DRMS is the 95-98% probability that the position will be within the stated 2 dimensional accuracy. The probability varies between 95-98% because the standard deviation of latitude and longitude may not always match. Figure 2 illustrates the Circular Error Probable for GPS.

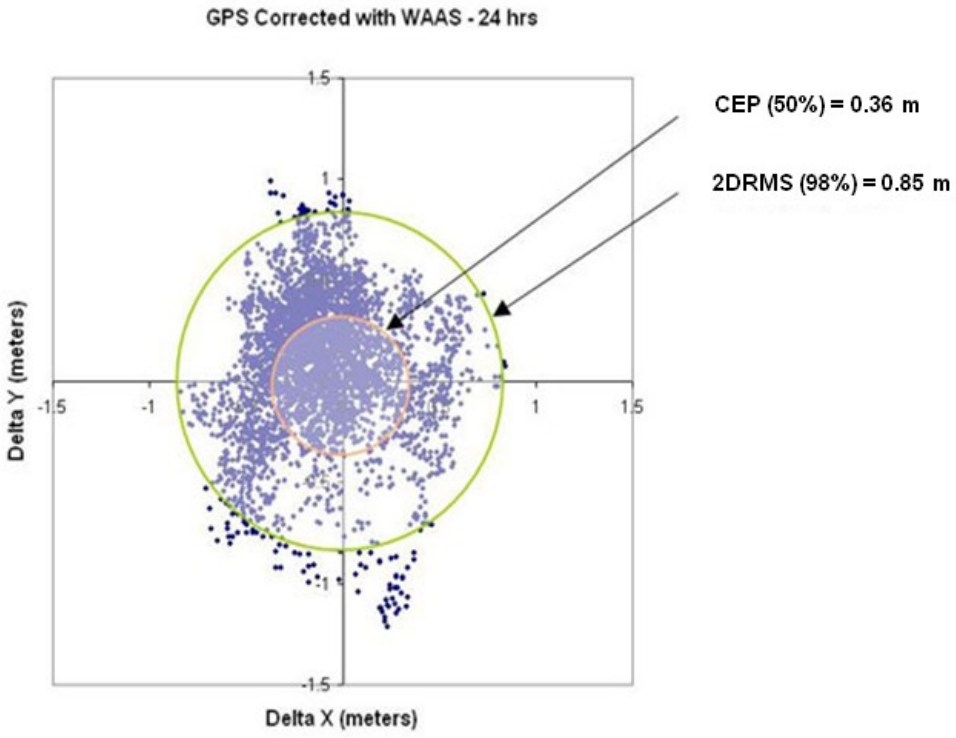

FIGURE 2. Reported GPS location points relative to the absolute position; corrected using WAAS which is GPS supplemented using ground stations (Earth Measurement Consulting).

The accuracy and precision of GPS is not reliable enough for small scale robotic applications. Micro-vehicles will maneuver in areas not much larger than a few square meters and will need to know their location to within a few millimeters depending on the application. Other limitations include area obstructions above 5° of elevation and can include trees, buildings, fences and cables. These obstructions have the effect of reducing the number of satellites that the GPS can "see," reducing the strength of the signal, creating satellite signal multipath, resulting in the delay of the satellite signal and corrupting the GPS measurements (Earth Measurement Consulting). Figure 3 illustrates how a GPS device can receive incorrect signals through signal multipath.

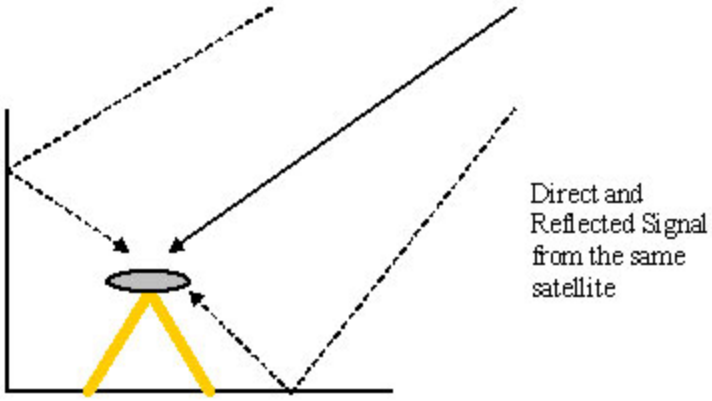

FIGURE 3. An example of GPS signal multipath with a GPS receiver mounted to a stable base (Earth Measurement Consulting).

Multipath can reduce accuracy up to a few meters (Byun, Hajj, & Young, 2002). To improve the accuracy of GPS, Real Time Kinematic, or RTK GPS, has been introduced. RTK is a process where GPS signal corrections are transmitted in real time from a reference receiver at a known location to a remote receiver. RTK capable GPS can compensate for atmospheric delay, orbital errors and other variables in GPS geometry, increasing positioning accuracy up to within a centimeter. However, these systems often cost tens of thousands of dollars to purchase and setup (Ashtech Technology, 2010).

Due to the limitations of standard GPS devices and the projected uses of these autonomous micro-vehicles, standard GPS should not be used as a means to precisely locate an autonomous micro-vehicle; thus, a new system must be used as a way to determine absolute position. These strategies include the development of a wide array of sensors, localization techniques, and course planning strategies. When these strategies are employed on tracked vehicles as

they move across landscapes, the vehicles may encounter obstructions that may not be detected as an obstacle that requires avoidance by the array of sensors. These obstacles such as gravel, sand piles, or loose dirt can cause changes in orientation as they shift when being traversed by small vehicles. The change in orientation must be detected and accounted for so that autonomous vehicles will continuously know their location relative to a starting location within a small operating area. This research can be applied to various robotic applications such as extra-planetary exploration and underground mapping, where GPS localization is not available, or other applications in which precision beyond the capability of GPS required such as autonomous air duct cleaning or lawn care.

II. PROBLEM STATEMENT

Consider an autonomous two track micro-vehicle designed to travel a straight trajectory towards a target location. If the vehicle encounters a change in surface friction, a loss of traction, or slip, occurs. For this research, a slip is defined when a reduction in surface friction results in relative motion between one or both tracks and the surface traveled. The objective of this project is to measure the change in position and orientation of an autonomous two track vehicle which experiences a slip and corrects for the error associated with the slip that has occurred. The primary motivation for this objective is to direct a tracked vehicle to move autonomously to a desired location from a known starting location. Figure 4 is the autonomous tracked micro-vehicle used as the test vehicle for this research. This vehicle is discussed in detail in the section titled "Vehicle Platform." In Figure 4, the major components used for navigation, the compass, the microcontroller along with the encoder wheel and receiver, are shown.

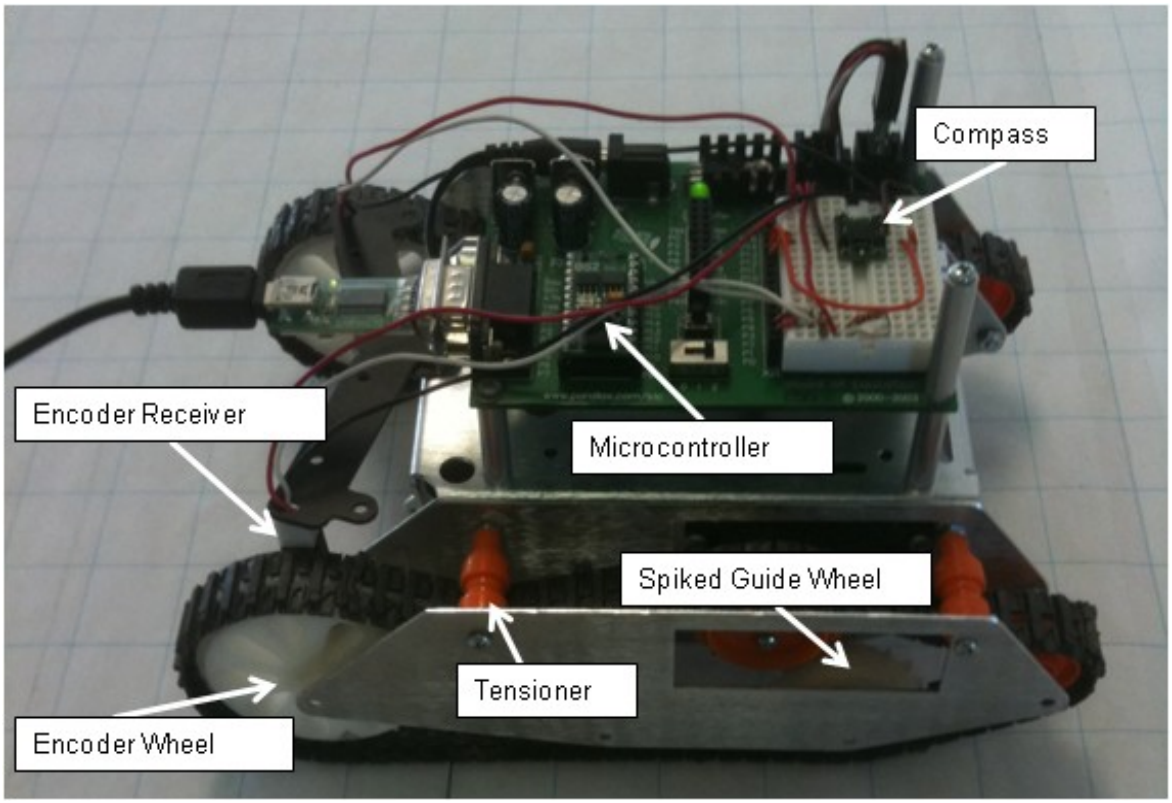

FIGURE 4. Autonomous tracked micro-vehicle with major components.

In the event that the autonomous two track vehicle does not encounter a slip, the vehicle will continue along its original trajectory to the target location (Figure 5(a)). Simultaneous slippage of both tracks, which results in relative motion between both tracks and the surface traveled, will not be considered since the position of the vehicle relative to its starting location is lost. The absolute position of the vehicle is lost because both tracks continue to move and the encoders continue to measure the rotation and translate that information to linear travel when in reality, only the tracks are moving, not the entire vehicle (Figure 5(b)). This case will not be considered because it is outside the scope of the project and must be handled using other localization methods such as those used by

inertial navigation systems. During a single track slip, the vehicle experiences a loss of absolute position in one track due to a loss of traction on that track, while still retaining absolute position in the other track. This changes both the position and orientation of the vehicle from the continued forward motion (Figure 5(c)). Of the cases presented in Figure 5, only Figure 5(c) will be considered for this project. Preliminary thought suggests that GPS localization should be the primary method for locating the autonomous micro-vehicle. However, as discussed in section I, when GPS is available it is not accurate enough because of signal multipath (Figure 3) nor is the level of precision of the GPS measurements great enough (Figure 2) for small scale applications.

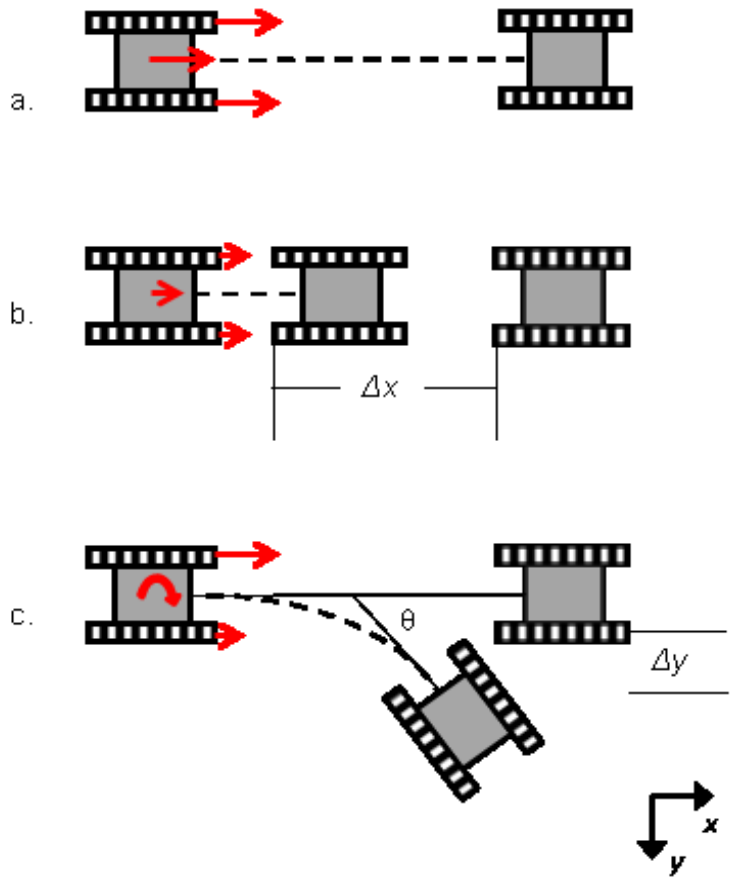

FIGURE 5. The ghosted figure represents both the position and orientation of the vehicle if a slip did not occur and the autonomous vehicle's presumed position and orientation when a slip condition is encountered. Δx and Δy are the error in the x and y direction and θ is the change in orientation of the vehicle. A tracked micro-vehicle with (a) traction at each track progresses in a straight line, (b) reduced traction on each track cannot maintain absolute position and (c) traction at left track and reduced traction at right track will rotate and travel in an arc in the clockwise direction.

Estimation of the vehicle's new position and orientation is generated using information from a pair of optical encoders and an electronic compass on the vehicle. A process that has been designed to control the autonomous vehicle will determine a new trajectory for the vehicle to travel to the target location. The

process used in this research is limited to only being able to detect one slip per trial, or test of the vehicle.

III. VEHICLE PLATFORM

The vehicle platform consists of a modified version of the Parallax® Boe-Bot (Parallax, 2010) as seen in Figure 4. Modifications include the use of a tank tread kit from Parallax® that has been fitted to custom wheels that reduce track/wheel separation. These modifications include the addition of spiked guide wheels, pinions used as track tensioners, and a custom encoder wheel. Three components reside onboard the vehicle for navigation: a microcontroller, a compass, and a pair of optical incremental encoders and corresponding light emitter/detector sensors (Parallax, 2010). The microcontroller receives data from the encoders' sensors and compass, which indicate the vehicle's absolute position and orientation with respect to the coordinate system illustrated in Figure 5. Algorithms encoded on the microcontroller then resolve the error associated with an encountered slip condition. The microcontroller is a single processor unit with limited processing speed of 20MHz that allows only one process to execute at a time. Consequently, the microcontroller cannot simultaneously control the direction and speed of the motors driving the tracks while monitoring the compass and encoder sensors. Therefore a ServoPAL from Parallax® (Parallax, 2010) is utilized for controlling the two Parallax® continuous rotation servo-

motors. The ServoPAL is sent instructions by the microcontroller each time the vehicle needs to start or stop motion. The ServoPAL then executes the instructions independently of the microprocessor until new instructions are received.

The compass, a Hitachi HM55B compass module, has a typical sensitivity of 1.0 μT (microtesla) and a maximum sensitivity of 1.6 μT (Parallax, Inc, 2005). Due to the sensitivity of the compass, small external magnetic fields can alter the compass reading. A range of up to 9 degrees has been experienced with the compass with the vehicle at rest. Also, due to the sensitivity of the compass, testing could not be completed in the presence of other electrical machinery or near metallic objects, both of which would alter the reading of the compass. These limitations are particular only to the device used on this autonomous micro-vehicle which was chosen for compatibility with the microcontroller. Other digital compass devices have increased sensitivity and precision for more accurate and consistent results.

Encoders and sensors are utilized on the vehicle, one on each track, as the rear track guide wheel and are in line with the gears that drive each track. The encoder wheel has 9 holes cut into its surface allowing for 18 instances when the signal of the encoder sensor will change polarity during one revolution of the encoder wheel. The distance the vehicle travels during two polarity changes of the signal to the receiver, is known as an encoder period (ep) (Pilgrim, 2004), which is the equivalent of 1.089 cm of travel (Figure 6). The length of the ep was determined by instructing the vehicle to move a known number of eps and

measuring the resulting distance. Once the distance had been measured, the number of eps was divided by the distance, resulting in one ep being the equivalent of 1.089 cm. This procedure was performed throughout the testing process to ensure its reliability.

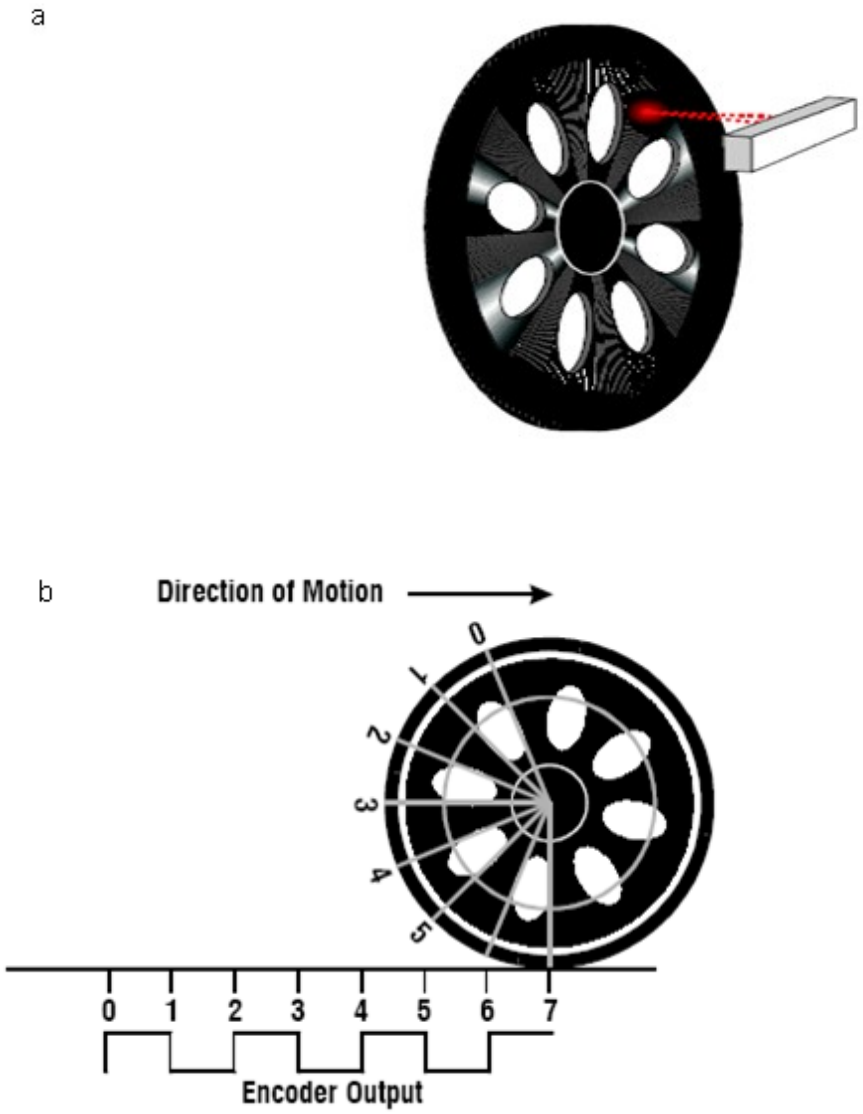

FIGURE 6. Example of (a) an incremental encoder wheel and optical receiver and (b) incremental encoder and its square wave output. (Pilgrim, 2004)

IV. THEORY

To develop an algorithm to correct for the change in orientation due to a slip, the following factors must be known: 1) the distance from the starting location to the target location, 2) the direction (clockwise or counter clockwise) and magnitude of the change in orientation and 3) how far the track that did not slip traveled during the slip. At the beginning of each trial, the distance to the target destination is provided. The process controlling the vehicle will also determine its beginning orientation using the compass module before initiating movement to the target location. Establishing this baseline, or original orientation, is the first step in detecting the amount of slip that occurs. Once the autonomous tracked vehicle begins traveling to the target location, the controlling process continuously accesses the compass to check the vehicles current orientation and continuously compares it to the original orientation. If the current orientation exceeds the slip threshold, which is the amount of orientation variation allowed within the process, the process will determine that the tracked vehicle is experiencing a slip.

Preceding the detection of a slip, the process continuously monitors the distance traveled by the micro-vehicle. Once the slip threshold has been

exceeded, the process records the distance already traveled and the encoders are reset so that the distance traveled only during the slip can be recorded. While the vehicle is slipping, it will be assumed that vehicle will travel in an arc of constant radius (Figure 7). During the slip, current and previous compass readings are compared. To indicate that the tracked vehicle is no longer in a rotational state, matching orientation readings must be received consecutively from the compass module, signifying that the vehicle is no longer slipping. Once the slip has been completed, the compass is accessed to obtain the current orientation which is then be compared to the original orientation. This allows the process to determine the change in orientation during the slip. The change in orientation combined with the distance that the vehicle traveled will allow the algorithm to determine the vehicles new absolute position based upon the arc length and constant radius assumption. By assuming that the radius of the arc is constant, the parallel, ε_x, and perpendicular, ε_y, position coordinates can be determined allowing for the absolute position to be calculated. This is illustrated in Figure 7 where COR is the center or rotation, *r* is the radius of the arc, *L* is the length of the arc that was traveled during the slip and θ is the change in orientation due to the slip.

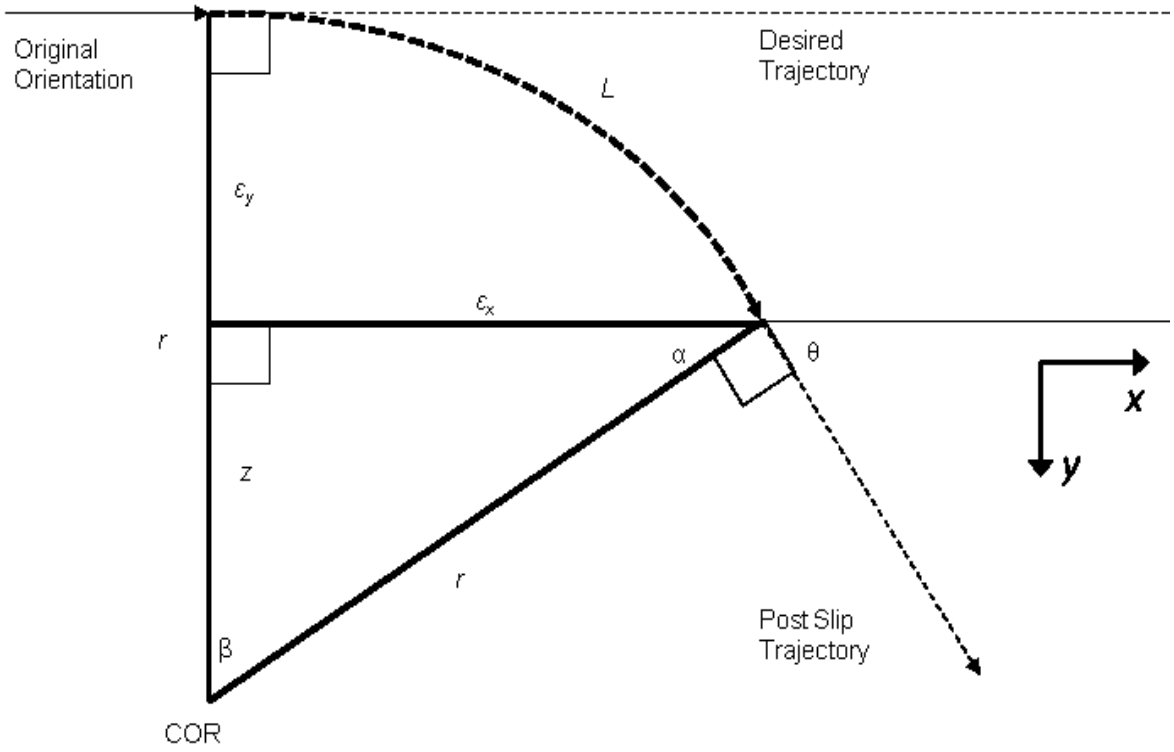

FIGURE 7. Slip geometry with constant radius assumption.

The primary variables from this figure are the lengths ε_x and ε_y, which are the position change that the tracked vehicle experiences due to the slip. Appendix I shows the calculations used to find ε_x and ε_y.

There are three different methods that have been developed to resolve the error associated with an encountered slip condition. Each method, the Arc Compensation Method, the Arc Extension Method and the Line Slip Method, will be tested individually and compared to each other to see which produced the most desirable results. The experiments have been designed to test the algorithms ability to adjust to each variation without further input from the user. The Arc Compensation Method resolves errors associated with slip conditions by

detecting the distance that the non slipping track moved once the algorithm detected a slip after crossing the slip threshold. This differs from the Arc Extension Method through the use of an equation that, once the vehicle has finished slipping, is used to extrapolate how far the vehicle traveled while experiencing slip conditions but prior to crossing the slip threshold. The third method, the Line Slip Method, assumes that when the vehicle encounters a slip condition that no change in position along the *y* axis occurs. This is discussed in detail in the section titled "The Algorithms." The key difference between the Arc Compensation Method and the Arc Extension Method is how the arc length is calculated. For the Arc Compensation Method, the arc length that is detected by the encoder after crossing the threshold is the length that is used. For the Arc Extension Method, the length of the arc prior to crossing the compass threshold is calculated using

$$L = L_p + \theta_t \left(\frac{L_p}{\theta - \theta_t} \right) \tag{1}$$

where L_p is the partial arc length as detected by the encoders and θ_t is the slip threshold and is equal to 15° for this project. Figure 8 illustrates the different arc lengths.

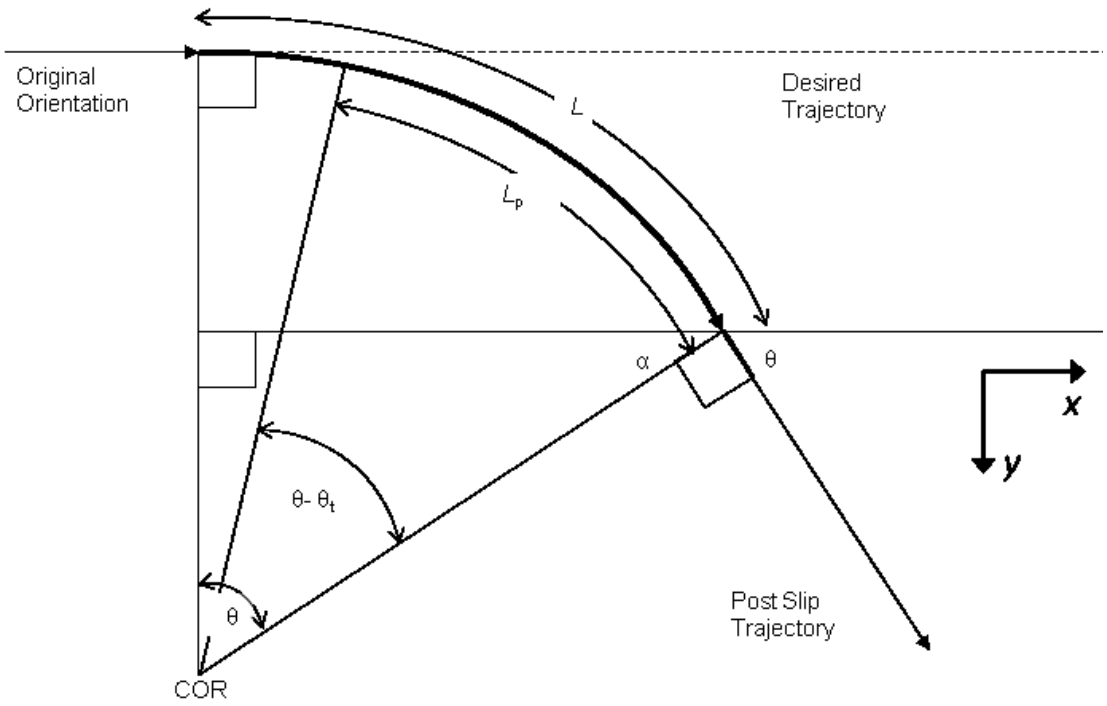

FIGURE 8. Illustration of partial and total arc length.

The process is designed to power the vehicle on a straight trajectory to its target location and monitor if a slip occurs. If a slip occurs the controlling process will then calculate the vehicles new position and orientation. Once the current position has been determined, the distance from the vehicles current location to the target location is calculated. The vehicle is then oriented to the target destination and then directed to move in a straight trajectory to the target location. The use of triangles to find the distance and the angle to the destination is illustrated in Figure 9.

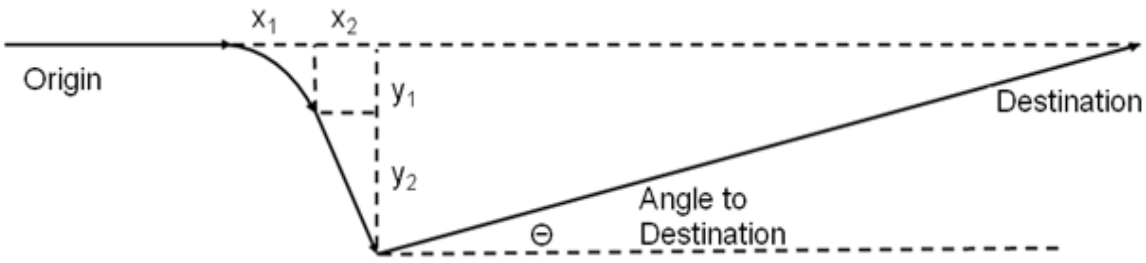

FIGURE 9. An illustration for returning to the target location.

V. THE ALGORITHMS

The algorithms are encoded on the microcontroller and are used to resolve the error associated with an encountered slip condition. The microcontroller has 32 bytes of RAM memory and 2 Kbytes of memory dedicated to the EEPROM which is used for the storage of the algorithm. Figure 11 is a flow chart categorizing the algorithm into different stages.

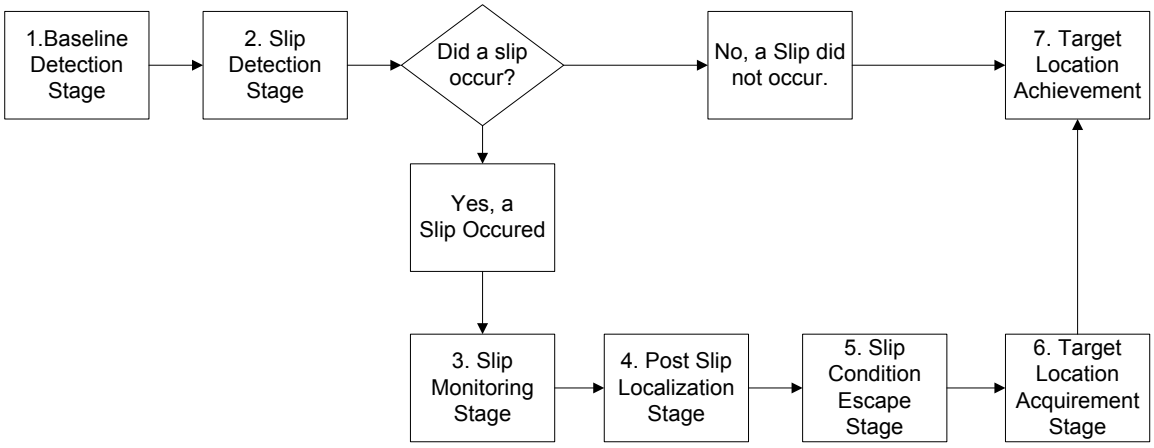

FIGURE 10. A flow chart describing the secession of events within the algorithms as various stages.

The algorithm begins in the Baseline Detection Stage by running an auto-alignment routine. The routine checks the encoders to see if a high or low signal

is being received. If the encoder returns a high signal, the algorithm directs the vehicle to rotate the track until the signal from the encoder becomes low. The transition from high to low is seen in Figure 6. This routine is applied to both the left and right tracks to ensure that the encoders are always in the same starting position for each side of the vehicle. Each encoder should begin on an edge so that the first encoder period is 10.89 mm like each encoder period preceding it. Potentially, this could cause a displacement error of up to 10.89 mm if the encoder begins just past the edge of the encoder period. For example, if the encoders begin at two different locations for a 10 ep move, one track may move 108.9 mm and the other track may move 112 mm because of the extra distance to cover before a signal change occurred and would still read the move as 10 eps. This auto-alignment routine is run prior to placing the robot at the origin. After the vehicle has been aligned and placed at the origin, a loop that reads the compass output is executed that makes 10 orientation readings and then determines an average of these 10 values, which then becomes the nominal orientation. An average is needed to deal with the flux the compass experiences, most fluxes are no more than 2 degrees. The distance from the origin to the target location is then provided. A flow chart detailing the steps of the Baseline Detection Stage is presented in Figure 12.

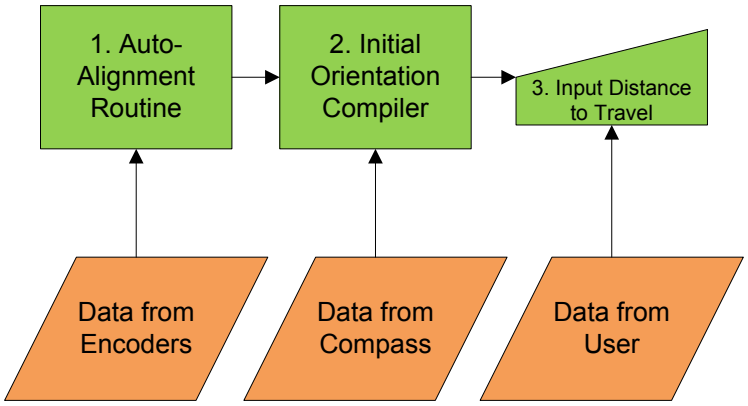

FIGURE 11. A flow chart of the three events that are the Baseline Detection Stage. Green indicates a process and orange describes where the data for that process is acquired.

The autonomous tracked vehicle will begin to move forward from the origin and the algorithm will enter the Slip Monitoring Stage. This stage continuously monitors the compass and the encoders so that the distance traveled and the orientation of the vehicle is always known. The algorithm will stay in this loop unless one of two conditions is met. The first condition is if the amount of encoder pulses equals the desired distance to travel. If this condition is met then the autonomous tracked vehicle has arrived at its destination without experiencing any slip. When this occurs, the algorithm will move to the Target Achievement Stage, as seen in Figure 11, where the vehicle is told to stop forward motion. The second condition for the algorithm to exit the loop is if the compass returns an orientation that is outside of the limits of the given orientation threshold. The slip threshold, θ_t, for this system is 15 degrees from the nominal orientation. If the compass returns a reading outside of these limits, it is assumed that the vehicle is slipping. As the vehicle begins to slip, the algorithm saves the

distance that has already been traveled and resets the counter for the encoders so that the distance traveled during the slip is recorded. The algorithm will then move into the Slip Monitoring Stage which continuously monitors the compass and the encoders. The Slip Monitoring Stage is illustrated in Figure 13.

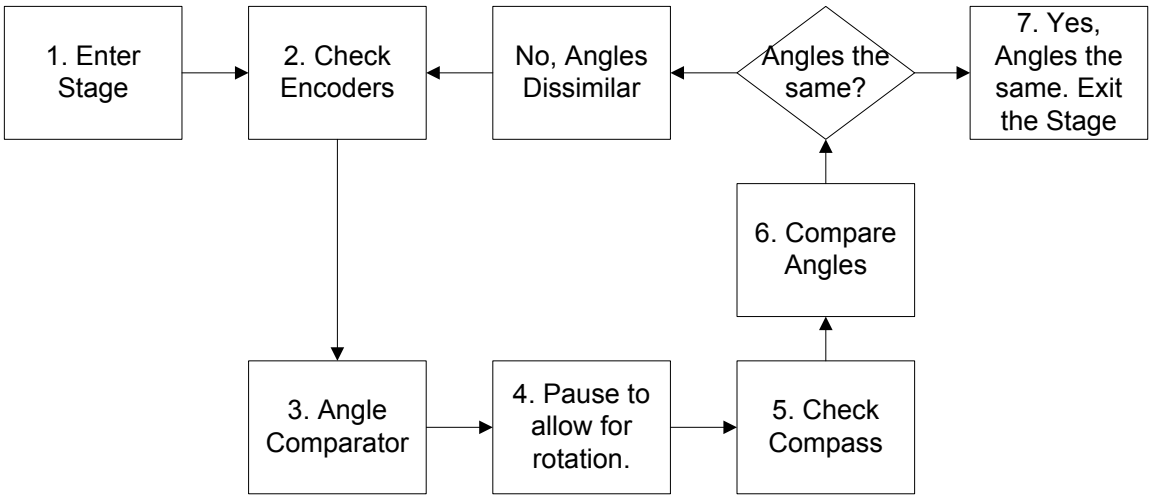

FIGURE 12. Flow diagram for the Slip Monitoring Stage.

The order in which this loop acquires information is significant due to the way the information is compiled and compared. If the order is changed, the value assigned to the variable for the previous angle reading will always be equal to the value assigned to the variable for the current angle reading and cause only a single cycle through the loop. First the loop checks the encoders to keep track of distance and then the loop compares the previous angle reading from the compass to the current angle reading. If the two angles are different, the loop saves the new angle as the previous angle for comparison in the next time through the loop. The loop then pauses for 0.1 seconds to allow for any change

in orientation that is occurring and then accesses the compass to get a new angle reading and begins the loop again. This loop will continue until the compass provides two consecutive readings at the same orientation.

When two consecutive readings occur, the algorithm will know that the slipping has ceased and will exit the Slip Monitoring Stage and move into the Post Slip Localization Stage which signals a stop in forward motion for the tracked vehicle. The algorithm then takes 10 orientation readings and averages the values to get a final orientation after the slip. The algorithm then enters a routine to find out if the vehicle rotated clockwise or counter clockwise. From this information, the appropriate encoder information can be selected and the angle change can be measured for use later in the algorithm. If the vehicle rotated CW then the data from the left encoder is used, otherwise the information from the right encoder is used. The Post Slip Localization Stage is illustrated in Figure 14.

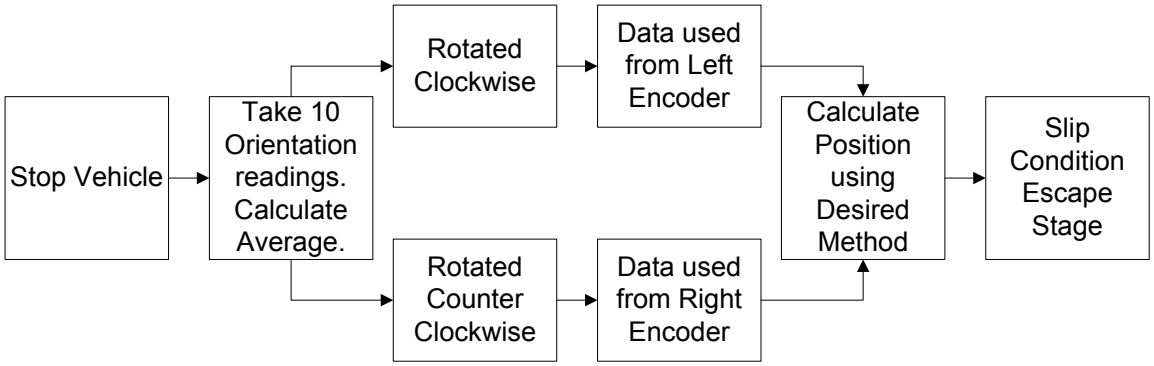

FIGURE 13. Flow Diagram for the Post Slip Localization Stage.

Up to this point in the algorithm, all versions are the same. For the Arc Compensation Method, the distance traveled during the slip, but prior to the threshold is treated as if the vehicle only travels in the *x* direction and there is no displacement in the *y* direction. The distance that the non slipping track traveled is saved as ε_x. The displacement in the *y* direction, ε_y, is only accounted for after the slip exceeds the slip angle. Equation 2 is not applied in this version of the algorithm. Figure 15 illustrates that only L_d is accounted for in this algorithm and is found simply from encoder information on the non slipping track.

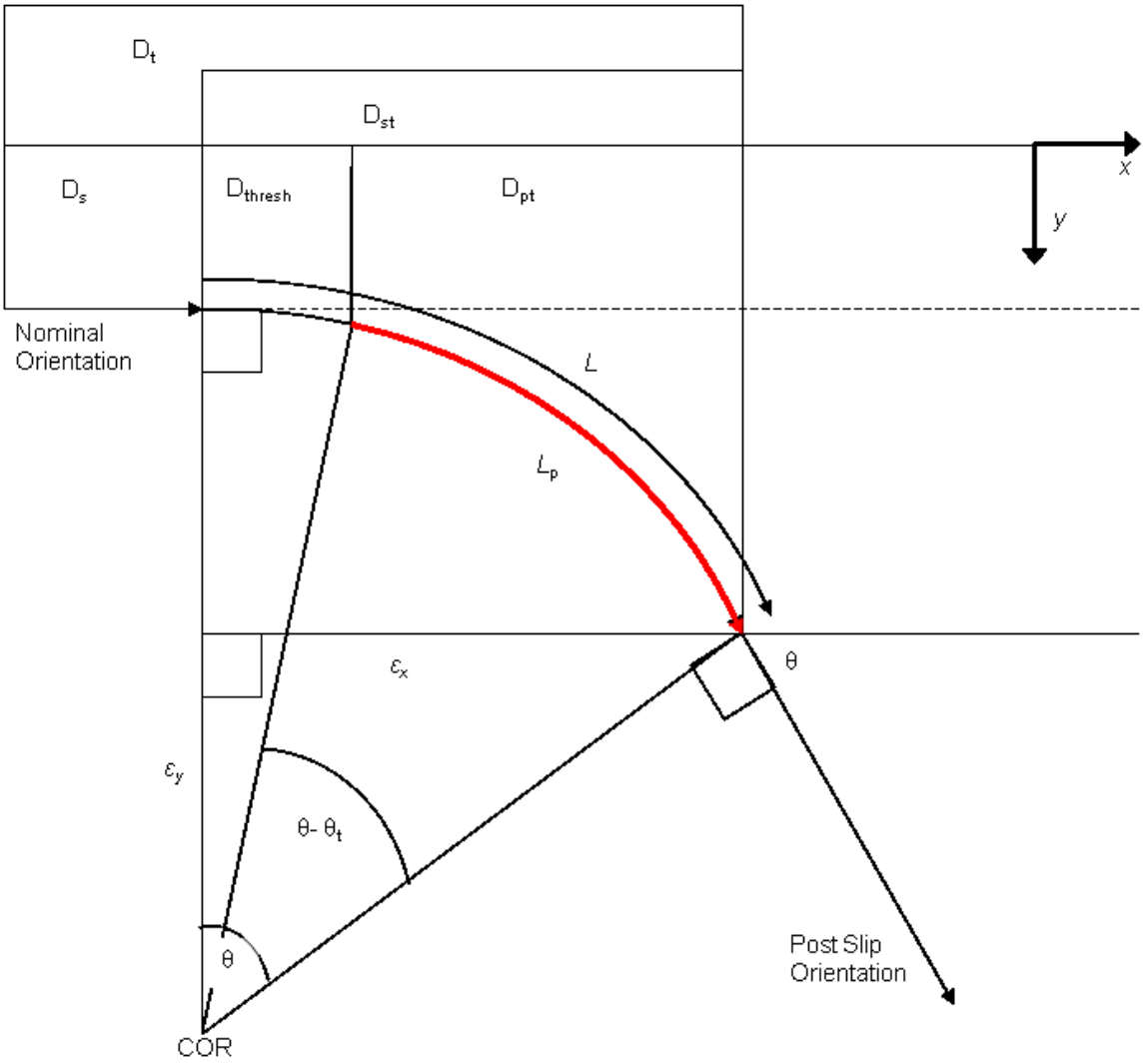

FIGURE 14. Illustration of the arc, L_p, as detected by the non slipping track and traveling distances for the Arc Extension Method.

For the Arc Extension Method, the distance that the vehicle travels during the slip is estimated using Equation 1 to compensate for the unmeasured arc length. Finding the *x* position requires more manipulation in this algorithm because the distance traveled during the slip but prior to reaching the threshold has already

been accounted for and just adding the *x* distance for the entire slip would be the equivalent of

$$D_t = \mathrm{D_s} + \mathrm{D_{thresh}} + \mathrm{D_{st}} \quad (2)$$

where $\mathrm{D_t}$ is the total distance traveled prior to slipping and after the slip, $\mathrm{D_s}$ is the distance to the slip, $\mathrm{D_{thresh}}$ is the distance traveled prior to reaching the threshold and $\mathrm{D_{st}}$ is the slip travel, or the distance traveled during the entire slip as illustrated in Figure 14. The total distance equation should not account for $\mathrm{D_{thresh}}$ and look like

$$D_t = \mathrm{D_s} + \mathrm{D_{st}} \quad (3)$$

To find the unknown distance of how far the tracked vehicle traveled during the slip but prior to reaching the threshold, the distance traveled during the slip after the threshold can be subtracted from the estimated distance traveled during the entire slip. Equation 7 then becomes

$$D_t = \mathrm{D_s} + \mathrm{D_{st}} + \mathrm{D_{thresh}} - (\mathrm{D_{st}} - \mathrm{D_{pt}}) \quad (4)$$

where $\mathrm{D_{pt}}$ is the distance traveled while slipping only after crossing the threshold and

$$D_{thresh} = (\mathrm{D_{st}} - \mathrm{D_{pt}}) \quad (5)$$

From Figure 15, ε_x is equal to $\mathrm{D_{st}}$ for the Arc Extension Method. After cancellation, Equation 4 becomes Equation 3 and allows $\mathrm{D_t}$ to be found.

The third algorithm, the Straight Line Slip Method, treats the slip as no displacement along the *y* axis for the entire slip. The distance that is recorded by the encoders during the slip is treated as travel only in the *x* direction. Figure 16 is an illustration of the Straight Line Slip Method.

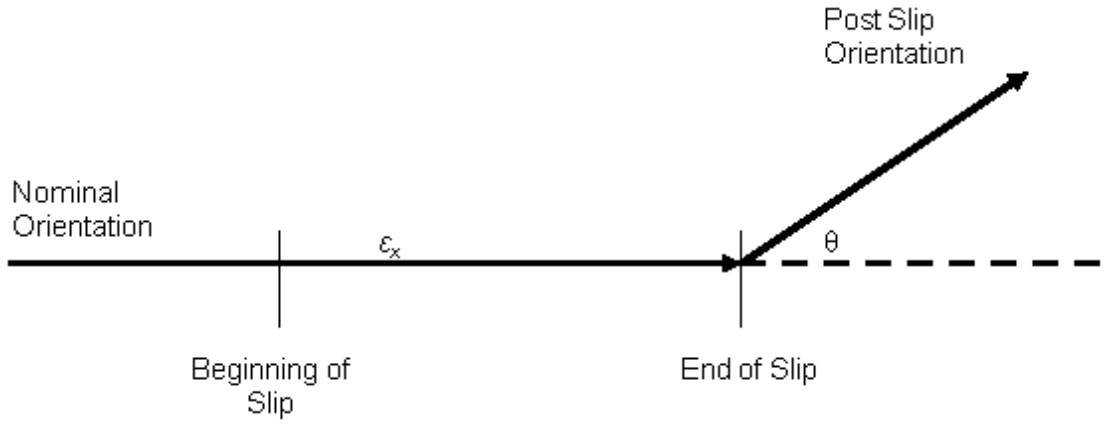

FIGURE 15. Illustration of how the Straight Line Slip Method treats a slip.

From this point forward, all three algorithms use the same equations and methods to arrive at the target location. Once the algorithm determines the position of the tracked vehicle after the slip, the algorithm enters the Slip Condition Escape Stage where vehicle precedes 30.5 cm on the post slip

orientation in order to clear the slip condition. The algorithm then enters the Target Location Acquirement Stage and finds the ε_x and ε_y distances traveled, and the orientation required to proceed to its target location is determined. The tracked vehicle then rotates to that orientation and moves forward to the destination. Once the tracked vehicle arrives at the target location and enters the Target Location Achievement Stage and the vehicle rotates back to the initial orientation. Figure 17 is an illustration of the decision making process for the complete algorithm.

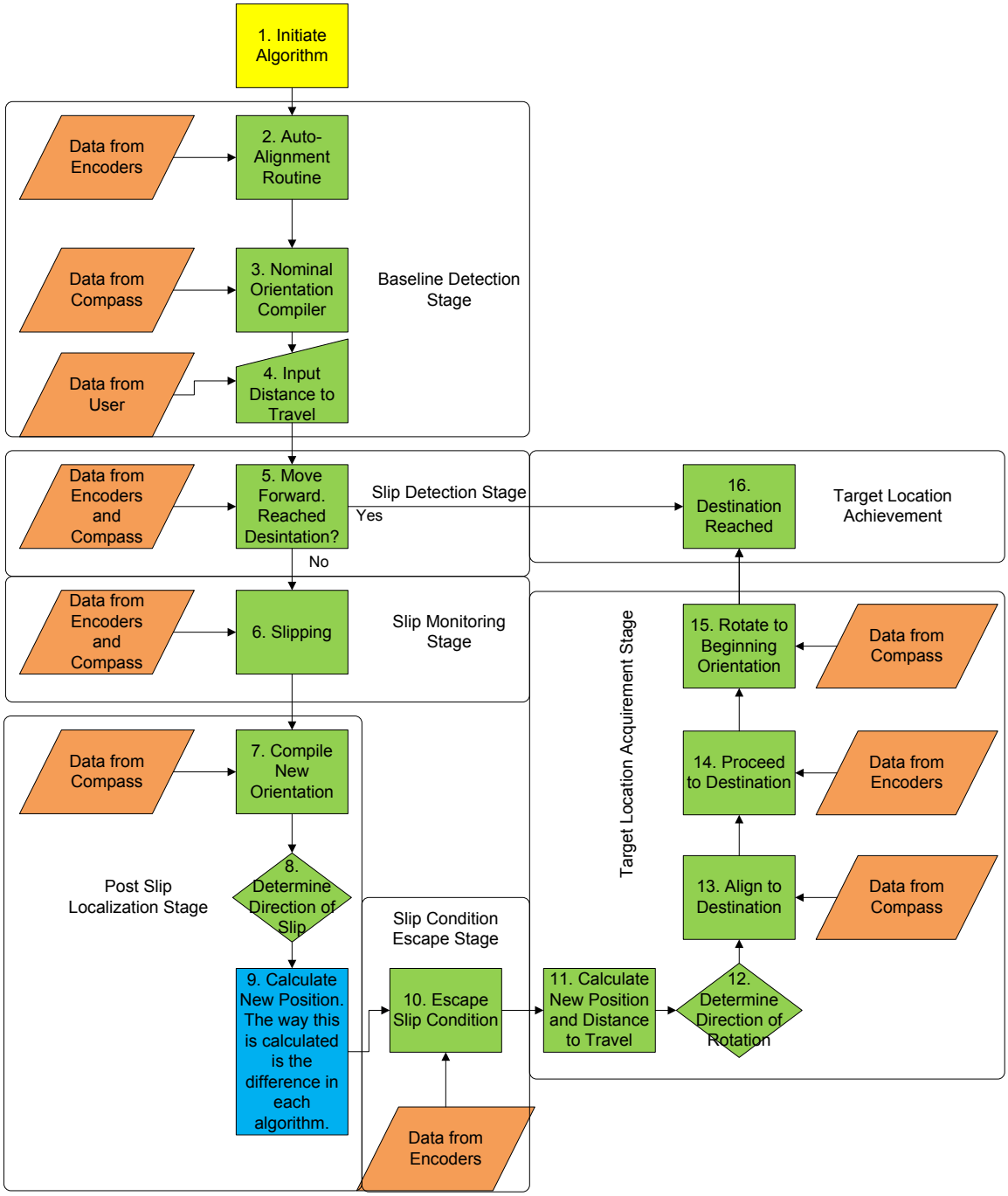

FIGURE 16. Flow Chart for the structure of the algorithms. Green spaces are processes and decisions, orange spaces are information used to control the process or decision and the blue box is where each algorithm differs. Each stage of the algorithm has been boxed in.

VI. EXPERIMENT

Many external conditions can cause an autonomous tracked vehicle to deviate from a straight trajectory. In order to simulate a slip condition with a high level of repeatability and control, a deck of new plastic coated playing cards was used. Thirty playing cards are stacked and given a horizontal shear at an angle of 30° from the ground as shown in Figure 18.

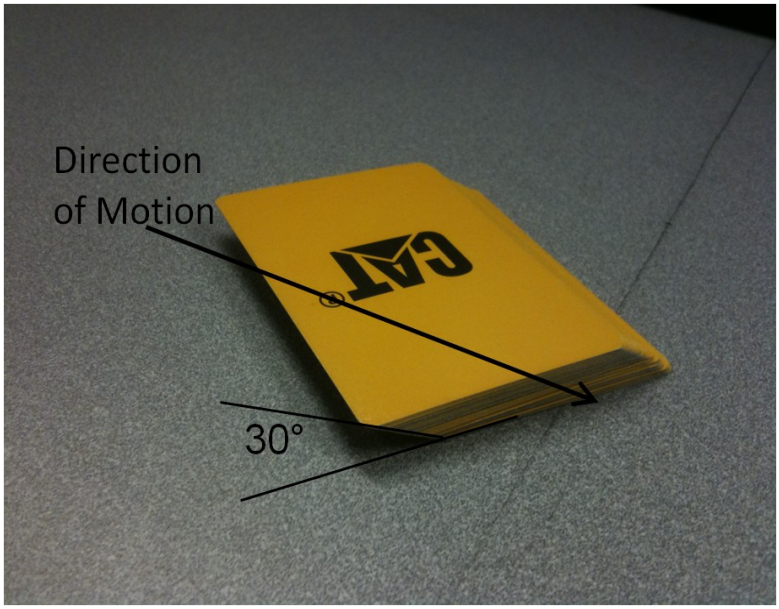

FIGURE 17. Stack of playing cards with an induced shear.

Each trial was setup so that the autonomous tracked micro-vehicle always rotated clockwise and followed a path similar to the one illustrated in Figure 19.

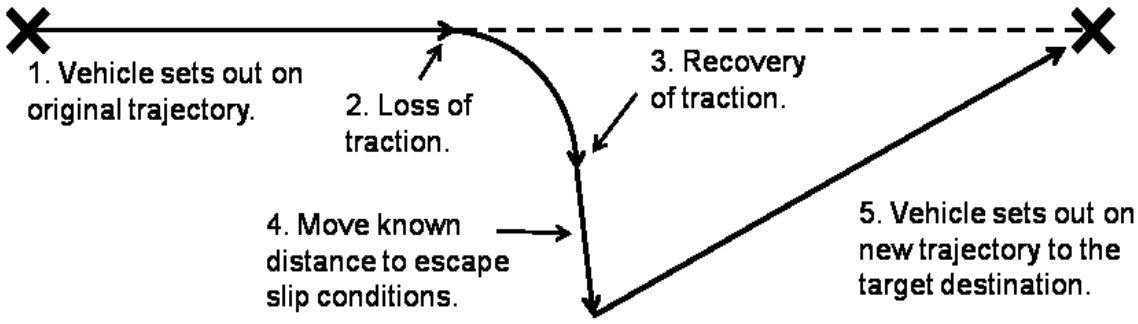

FIGURE 18. An illustration of the general path taken by the micro-vehicle for each trial.

The experiment was completed in a series of trials that were designed to test the autonomous tracked vehicles capabilities in various situations. Three different

variables were tested: distance to the destination from the origin, distance to the slip and the version of the algorithm that is controlling the tracked vehicle. Table I shows the variables tested during each trial for all three algorithms.

TABLE I

VARIABLE MATRIX FOR ALL THREE ALGORITHMS

Distance to Slip	Distance to Destination: .9m	1.2m	1.8m
5cm	X	X	
15.2cm	X	X	X
25.4cm			X

Each different combination of the variables as marked above with an “X” was tested 10 times on a large wooden table measuring 3 meters by 1.2 meters, resulting in 180 tests for the 18 different combinations. The 1.8 m test substituted a 5 cm slip distance for a 25.4 cm slip distance; the tests for 91.5 cm and 122 cm distances will not have a 25.4 cm distance to slip. This is due to the arctangent (ATN) command in the algorithm which has is limited to a range of -127 to + 127 (Martin, Williams, Gracey, Alvarez, & Lindsay, 2005).This command can be seen on line 222 of the source code for Arc Compensation Method in Appendix IIpI. If the distance after the slip exceeds 127 eps the command will overflow. For example, if the distance remaining after the slip is 132 eps, the algorithm will

calculate this as -122 eps. The algorithm then believes that it has exceeded its destination by 122 eps and will perform later calculations under this assumption which will lead to extremely poor test results. This cause for error was found during testing and leads to a new distance limitation for the tracked vehicle. The new limitation is the traveling distance after recovering from the slip cannot exceed 127 eps or 138 cm.

Testing begins by pressing the reset button on the tracked vehicle, this runs the automatic encoder alignment routine to ensure that the encoders are in the same position. The tracked vehicle is placed on a sheet of one inch grid paper and aligned so that the tracks of the vehicle is parallel with the lines on the grid paper and that the measuring point on the vehicle is consistently in the same position. The distance to the slip is measured from the lower front axle of the vehicle to the center of the set of 30 stacked playing cards which are in line with the trajectory of the tracked vehicle as shown in Figure 20. All distance measurements were taken with a standard steel tape measure to the nearest 1/8 inch and converted to metric.

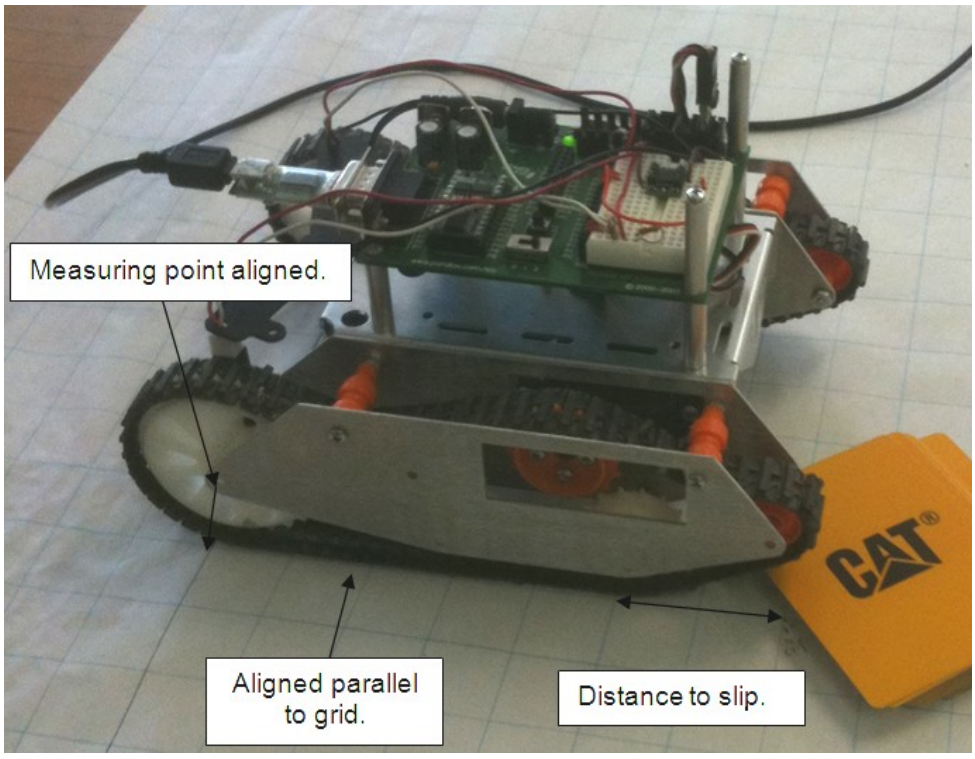

FIGURE 19. Autonomous tracked vehicle in initial position for testing.

The cards that will become the slip condition are placed in the vehicles paths and visually aligned so that they form a 45 degree angle to the grid paper as shown in Figure 21.

FIGURE 20. Alignment of playing cards that will become the slip condition.

Once the Autonomous tracked vehicle has been aligned and placed at the origin and the cards have been placed in the path of the tracked vehicle, the vehicle sets off on its trajectory. The robot continues until a slip is encountered and after the slip condition is cleared, the vehicle pauses 15 seconds to allow physical measurement of the actual slip angle with a protractor. The vehicle then reorients itself toward the target destination and again pauses for 15 seconds for another measurement with the protractor. Finally, the vehicle continues along its new path to the target destination. Upon completion of the algorithm, the *x* and *y* distances from the target location to the place where the robot stopped are recorded.

VII. RESULTS, DISCUSSION AND ANALYSIS

For each trial of the algorithm, the algorithm detected the angle of slip that it encountered during the trial and reported the angle in degrees to the nearest whole number to the computer interface, the computer only received data, it did not externally control the vehicle. Concurrently, each slip was measured to the nearest whole number using a protractor and recorded in a separate spreadsheet. Both readings were taken as a means to record any error related to the compass as reported by the algorithm with the measurements taken with the protractor being the definitive measure. All measurements taken from the compass are considered measured readings; all measurements taken from the protractor are considered actual readings. The average difference between the actual and measured slip angle is 2 degrees for all the algorithms. The maximum difference is 14 degrees and the minimum is no error between the measurements. The standard deviation between the actual and measured values is 2 degrees. Figure 21 breaks down the amount of difference, by percent, between the actual and measured angles of occurrences.

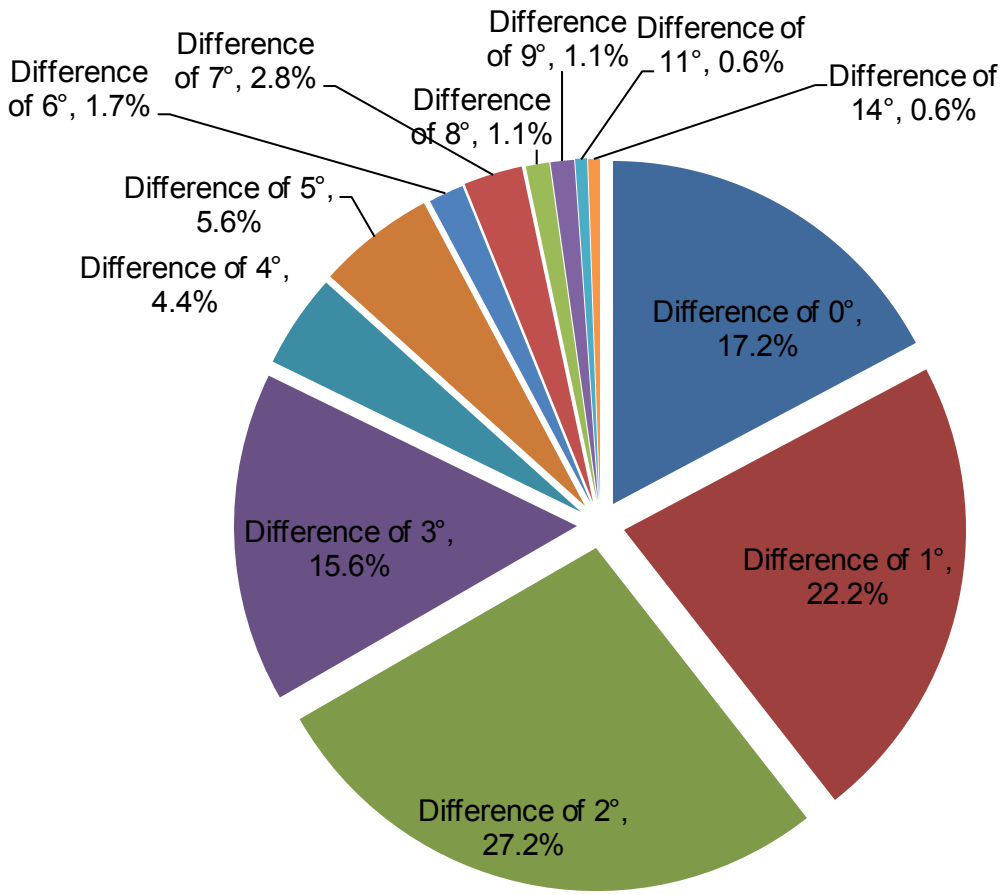

FIGURE 21. The amount of error in the slip angle measurement.

From Figure 21, 82.2% of the readings reported by the algorithm experience a difference of 3 degrees or less when compared to the actual value. Figure 21 validate that even though the compass is prone to error; the readings are consistently similar to the actual value and are reliable for use in the algorithm as a means to detect changes in orientation. However, in an effort to reduce the amount of error that propagates in the results, only trials that have a difference of three degrees between the actual reading and the measured reading where considered. All of the following figures represent this change. Figure 22 breaks

down the amount of difference, by percent, between the actual and measured angles of occurrences with a difference of three degrees or less.

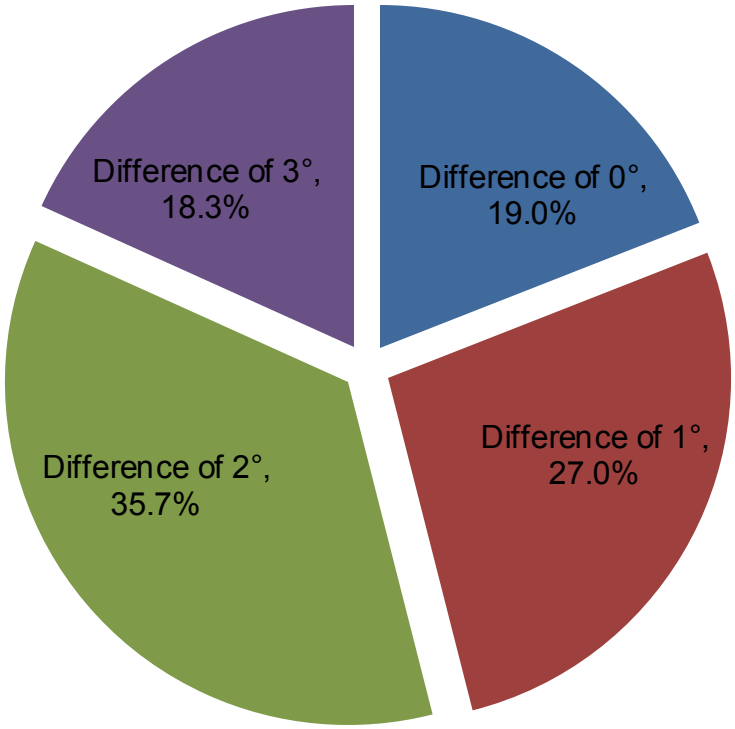

FIGURE 22. The amount of error in the slip angle measurement after eliminating trials with a measurement difference of more than three degrees.

This reduces the compasses level of uncertainty to +/- 3°. The protractor has a level of uncertainty of +/- 0.5°.

Table II highlights the metrics for the amount of slip for each algorithm as measured with the protractor (Actual) and as measured by the algorithm (Measured).

TABLE II

THE AVERAGE, MINIMUM, MAMXIMUM AND STANDARD DEVIATION FOR MEASURED SLIP (M) AND ACTUAL SLIP (A) AS ENCOUNTERED BY EACH ALGORITHM. ALL UNITS IN MM.

	Avg.		Min		Max		StdDev	
	M	A	M	A	M	A	M	A
Arc Extension Method	16	15	10	8	29	28	4	5
Arc Compensation Method	16	16	7	8	23	24	4	5
Straight Line Slip Method	18	19	9	12	33	31	4	5

Table II shows that the characteristics of slipping are similar for each algorithm as measured by the algorithm. Because the amount of slip is not a controlled variable, the results of each slip are monitored and recorded to ensure that there are no biases between each algorithm.

Another point of interest is if the amount of slip the tracked vehicle encountered would have any impact on the vehicles ability to reach its target location. Figures 23 – 25 are the final destination results versus the amount of slip measured by the compass the vehicle experienced for each trial and each method.

.

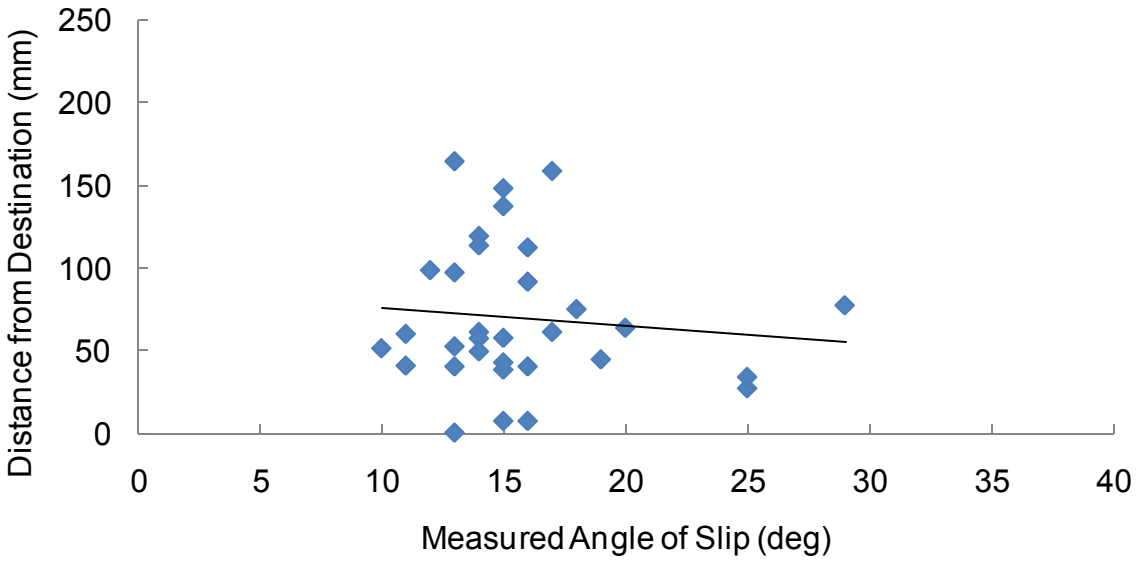

FIGURE 23. Measured angle of slip versus absolute error between actual and target location for the Arc Extension Method.

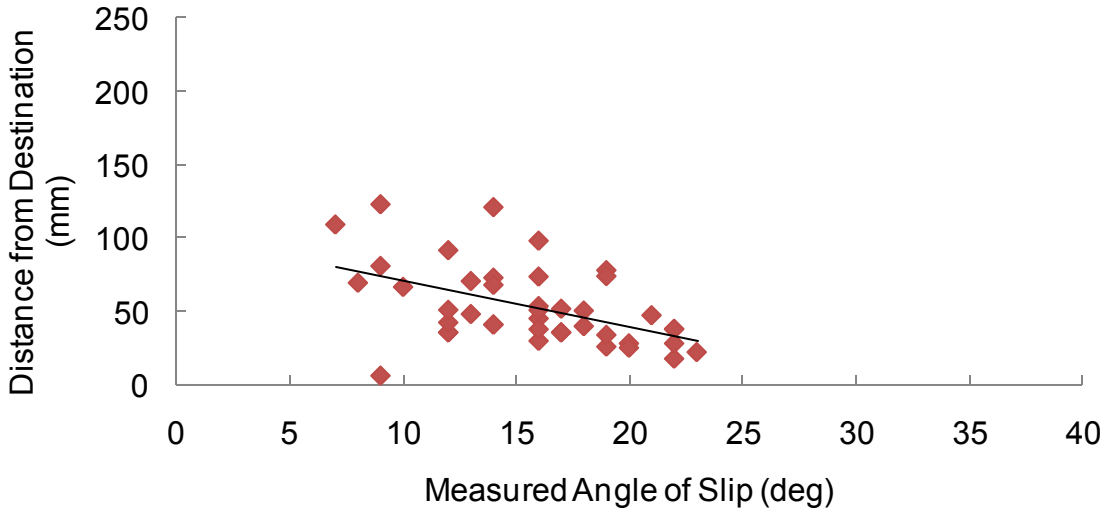

FIGURE 24. Measured angle of slip versus absolute error between actual and target location destination for the Arc Compensation Method.

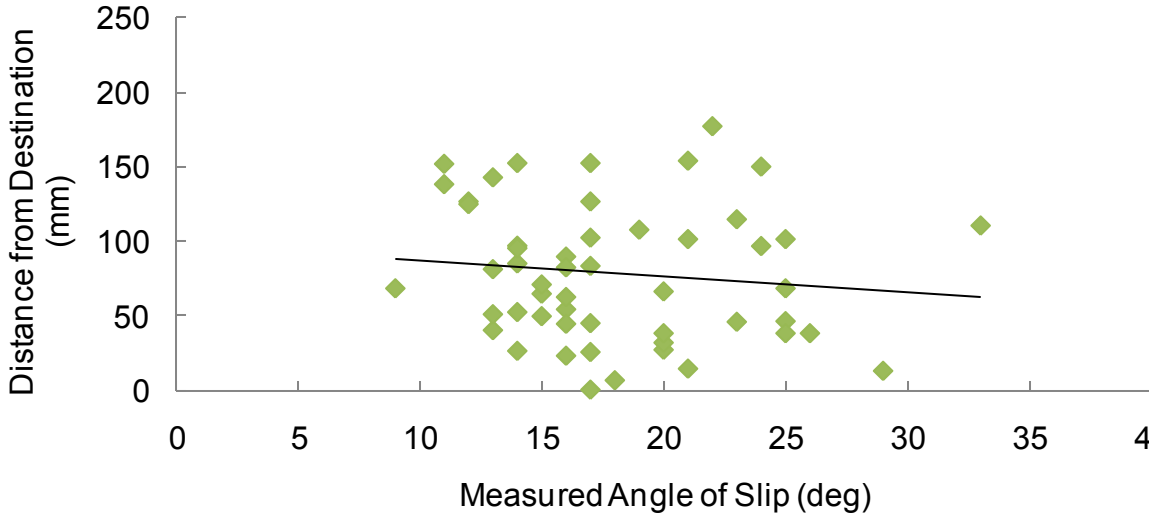

FIGURE 25. Measured angle of slip versus absolute error between actual and target location for the Straight Line Slip Method.

Figures 23-25 show a trend that is independent of each method. As the angle of slip increases, the distance to the target location from the point that the robot stopped decreases. This result was unexpected since greater orientation changes are associated with larger arc lengths and lower accuracy in achieving the target destination. This is not the case. The change in orientation is found to be independent of the arc length.

Along with being able to detect changes in orientation, the algorithms must be able to determine the vehicles new position that results from the slip. The algorithms combine orientation information and position information from the compass and the encoder, from which the algorithm can determine the new position of the vehicle. A key component to finding the position after a slip is determining the arc length that the tracked vehicle traveled during the slip. The

three algorithms employ three different methods to determining the vehicle's post-slip position.

The simplest method to determine the arc length is employed by the Straight Line Slip Method. This algorithm makes the assumption that the tracked vehicle did not travel in an arc during the slip, but rather slipped along a straight line and did not change position along the y axis. The most involved method to determine the arc length is the Arc Extension Method. This algorithm uses Equation 1 to account for the arc that the tracked vehicle traveled prior to crossing the slip threshold of 15 degrees. The Arc Compensation Method is similar to the Arc Extension Method but does not account for any slip prior to crossing the slip threshold.

Figure 26 shows the variations in the length of the arc for each slip.

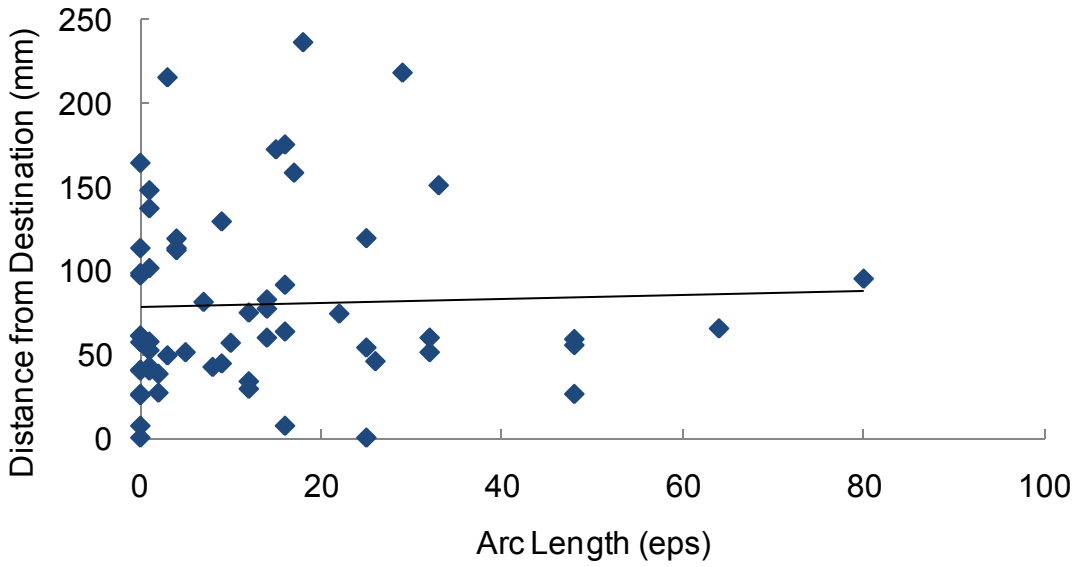

FIGURE 26. Arc length of each trial for the Arc Extension Method compared to where the tracked vehicle stopped relative to the target location before remove outliers. Each ep is equal to 10.89 mm.

From Figure 26, the arc length is sometimes equal to zero and ranges to 80 eps (37.1 cm). Based on observation, it is known that the case of zero arc length and the case of 80 eps arc length are untrue. From observation, a normal slip is between 1 and 16 cm. From the data recorded, only 51.7% of the values for the arc length when measured by the Arc Extension Method fall into this range. Also from the data, 21.7% of the values where measured to be zero. The reasons for these calculations is due to the compass; as soon as the compass reports that it has surpassed the 15 degree threshold it exits the loop and begins to wait for the compass to report back that the tracked vehicle is no longer slipping. The precision of the compass declines while the tracked vehicle is in motion, which is why the algorithm takes the orientation measurements used for calculations while

the tracked vehicle is stopped and why the 15 degree threshold is in place and not a lower amount. If the tracked vehicle was never slipping when the compass reported back the measurement or if the tracked vehicle was slipping but never changed orientation by more than 15 degrees then the arc was measured as zero. This is confirmed by the data which reports that for each occurrence of a zero arc length, the compass reported an angle change of 15 degrees or less 69% (9 of 13 instances) of the time.

The error resulting in large erroneous arc lengths originates in the assumption that the radius of the arc is constant and is manifested through Equation 2 which relies on a continuous change in orientation in order to be applicable. If the tracked vehicle changed orientation gradually up until the 15 degree threshold and then drastically changed orientation after crossing the threshold, Equation 1 assumes that the drastic change occurred throughout the entire slip and calculates the position information as if this were true.

In order to gain a better perspective on the Arc Extension Method's ability to measure the arc length, the erroneous data points were eliminated. All arc lengths that were greater than or equal to 25 eps have been eliminated, as have all data points equal to zero arc length. Figure 27 is the same as Figure 26 with the outliers removed.

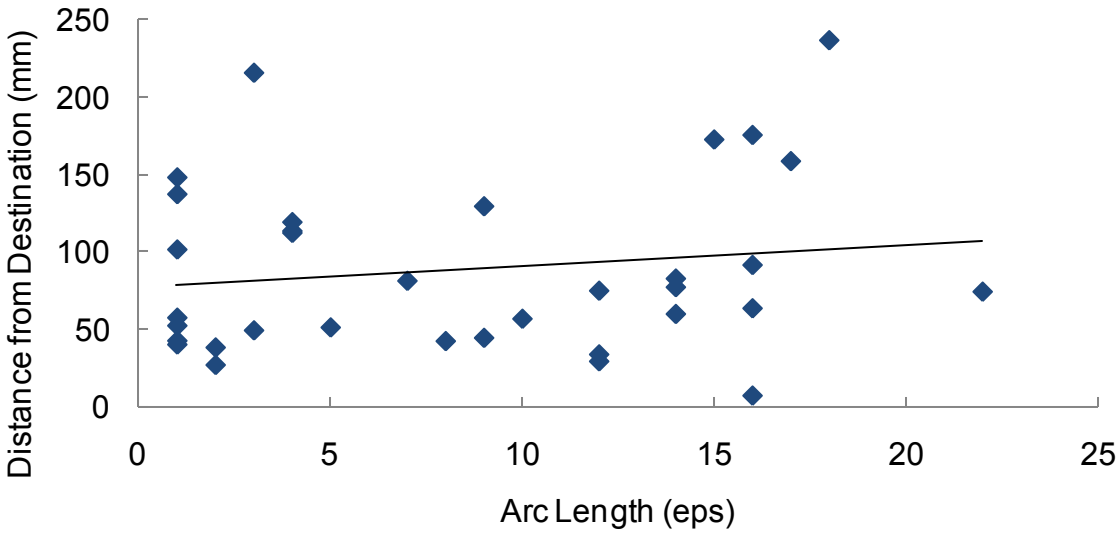

FIGURE 27. Arc length of each trial for the Arc Extension Method compared to where the tracked vehicle stopped relative to the target location with outliers removed. Each ep is equal to 10.89 mm.

From Figure 27, as the arc length increases during each slip, the distance to the target location from where the vehicle stopped also increases. The same trend can be seen for the Arc Compensation Method as illustrated in Figure 28.

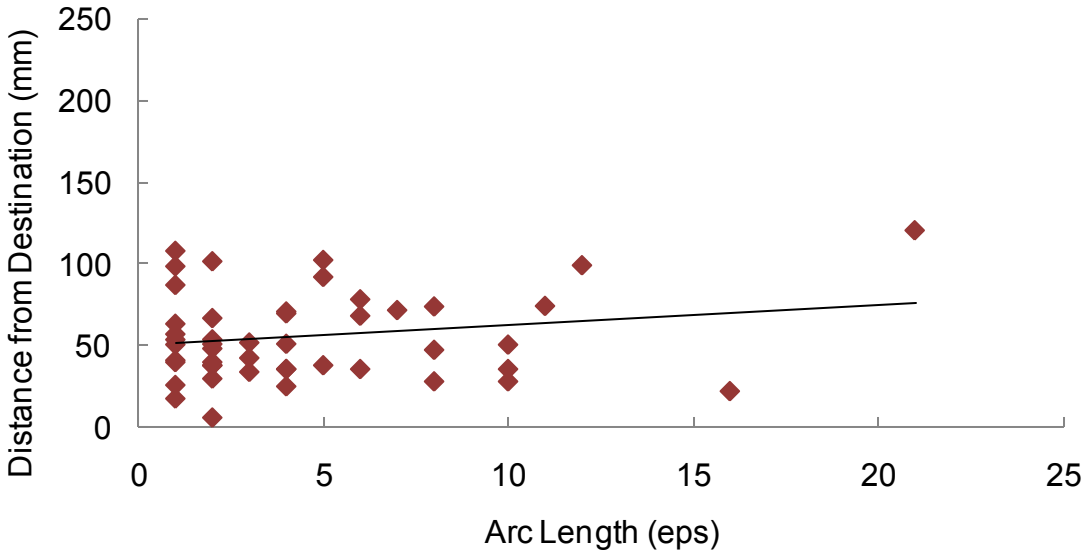

FIGURE 28. Arc length of each trial for the Arc Compensation Method compared to where the tracked vehicle stopped relative to the target location.

The results from Figures 27 and 28 are expected. The longer the arc length, the greater the displacement during a slip, which ~~this~~ requires more correction to the tracked vehicles path. Another result that can be seen in Figures 27 and 28 is the distribution of slip lengths. Figure 29 illustrates the percentage of occurrences at each arc length for the Arc Compensation Method.

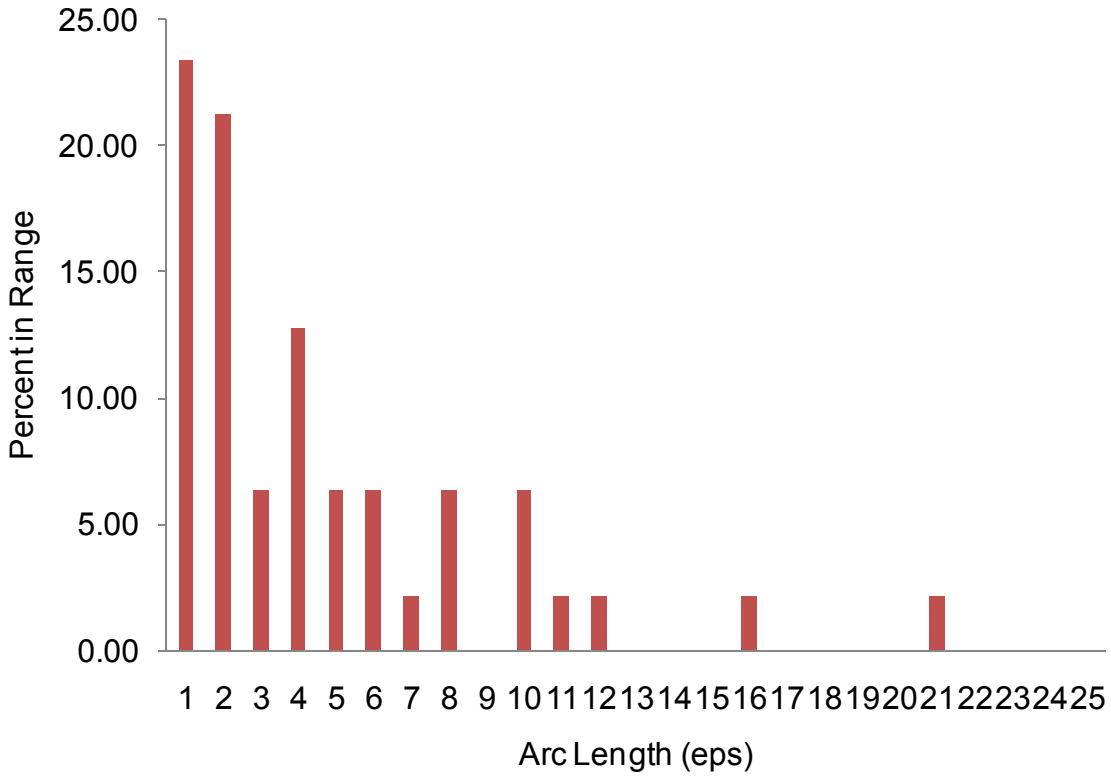

FIGURE 29. Percentage of arc lengths at each length for the Arc Compensation Method.

From Figure 29, the percentage of arc lengths is greater for the smaller arc lengths for the Arc Compensation Method. Figure 30 is the percentage of occurrences at each arc length for the Arc Extension Method.

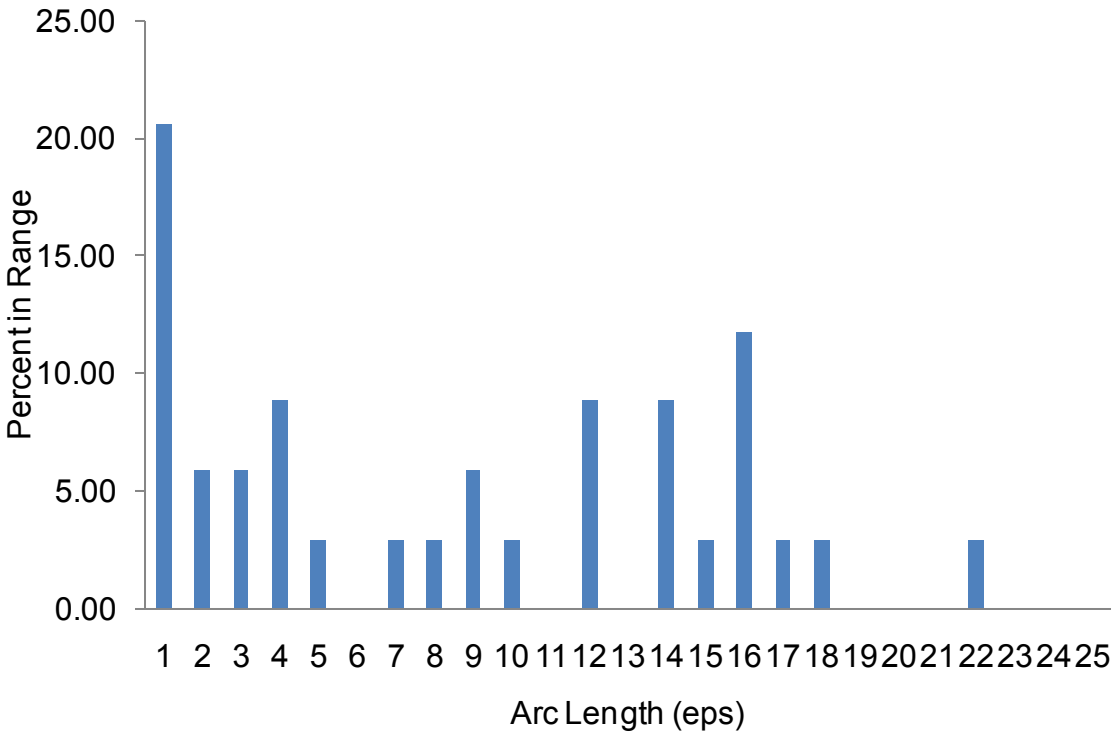

FIGURE 30. Percentage of arc lengths at each length for the Arc Extension Method.

From Figure 30, the distribution in arc lengths shifts for the Arc Extension Method. This is expected since this method calculates a longer arc because it considers the arc prior to crossing the slip threshold. Figures 29 and 30 also show a significant amount of slips resulting in and arc length of one ep (10.89 mm). This may is a result of both methods ability to detect when the tracked vehicle has finished slipping. Currently, the algorithm requires the compass to return two identical readings consecutively while slipping. If the vehicle is not rotating quickly enough between readings, the algorithms may prematurely signal that the vehicle is done slipping. Figures 27 and 28 support this theory, they illustrate that the tracked vehicle stops at highly varying distances from the target

location when the arc length is equal to one ep for both methods. It seems to me that if the slip was measured less that actual, the vehicle would fall short of the target. Figures 27 and 28 do not show this, but all of your figures later (those showing xy plots from destination) do. Indicate that as shown later, these two methods almost always (use percentage rather than my relative grammar here) fall short of the destination.

After each trial, the distance to the target location to the point that the vehicle stopped was measured in the *x* and *y* direction and recorded using a standard tape measure. From this data, information and statistics for each algorithm were tabulated and modeled with scatter plots. For each series of test, the distance that the vehicle had to travel was varied from 0.9 m (3 ft), 1.2 m (4 ft) and 1.8 mm (6 ft). Overall, the Arc Compensation Method averaged a radial distance of 60 mm from the target location to the point that the vehicle stopped. Figure 31 plots where the tracked vehicle stopped for each trial relative to the target location for all three methods.

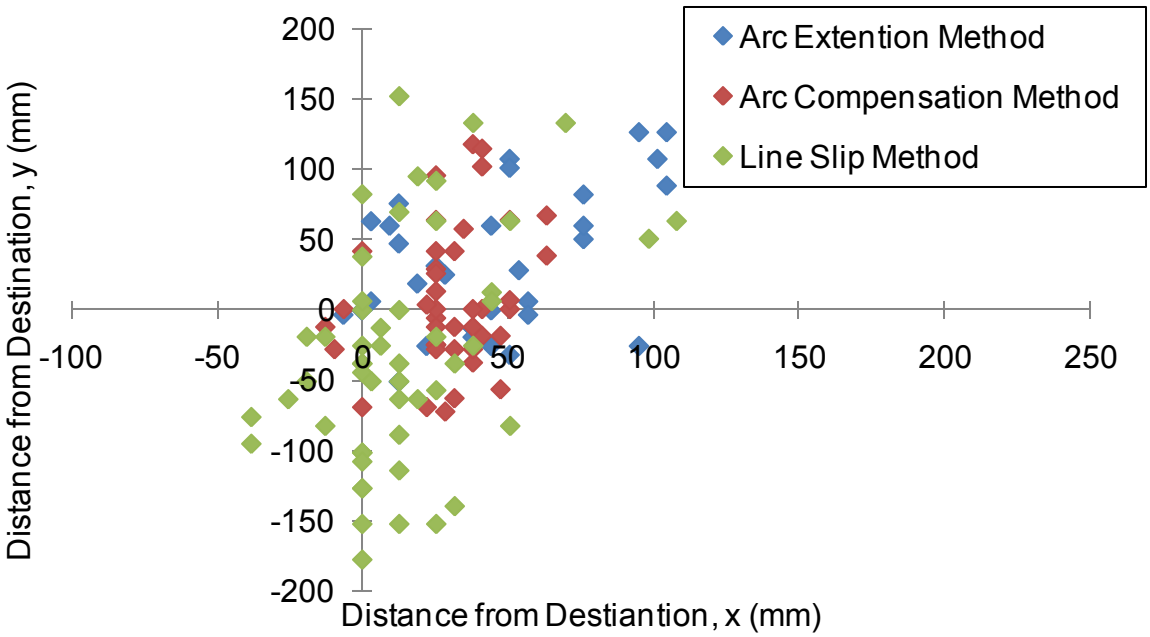

FIGURE 31. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for all test data and for all three methods. The direction of travel relative to the graph is from right to left.

From Figure 31, each method displays varying levels of precision and accuracy. The Line Slip Method has a wide range of data points whereas the Arc Compensation Method has a more concentrated grouping of data points. The Arc Extension Method tends to be concentrated in the first quadrant. Figure 32 plots where the tracked vehicle stopped for each trial relative to the target location for the Arc Compensation Method alone.

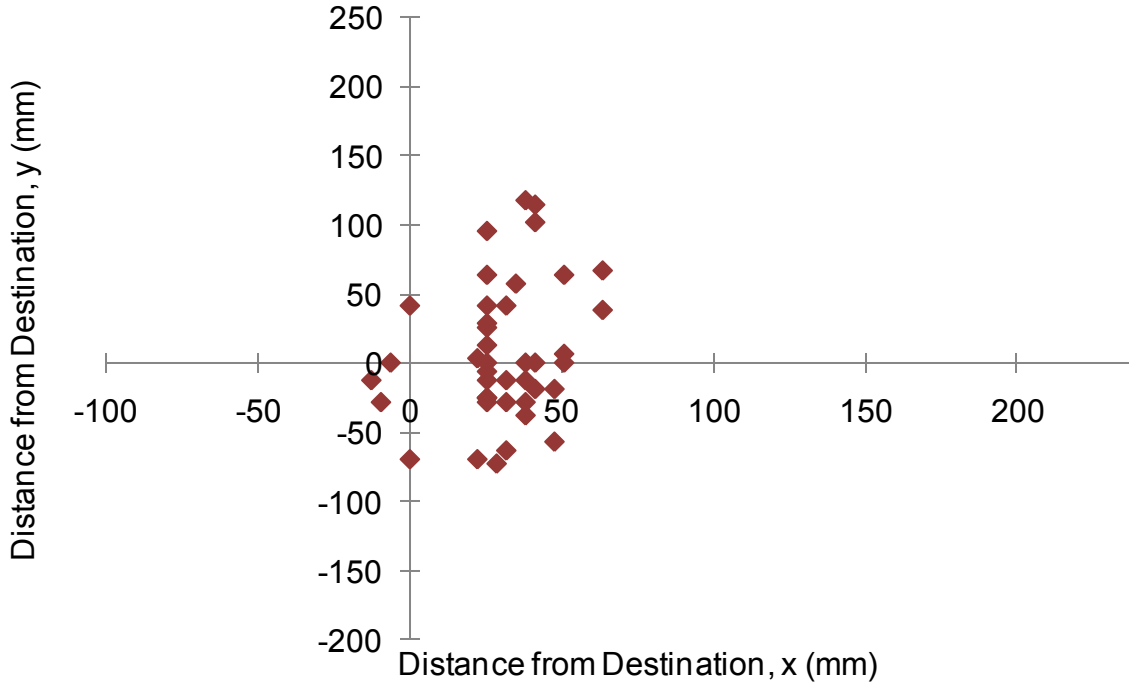

FIGURE 32. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for all tests of the Arc Compensation Method. The direction of travel relative to the graph is from right to left.

From this figure, the tracked vehicle consistently came short of the target location and never stopped at the coordinate (0,0) which is the target location. The Arc Compensation method had an average distance to the target location from the point that the vehicle stopped of 53 mm. Figure 33 illustrates the distance to the target location for the short range test.

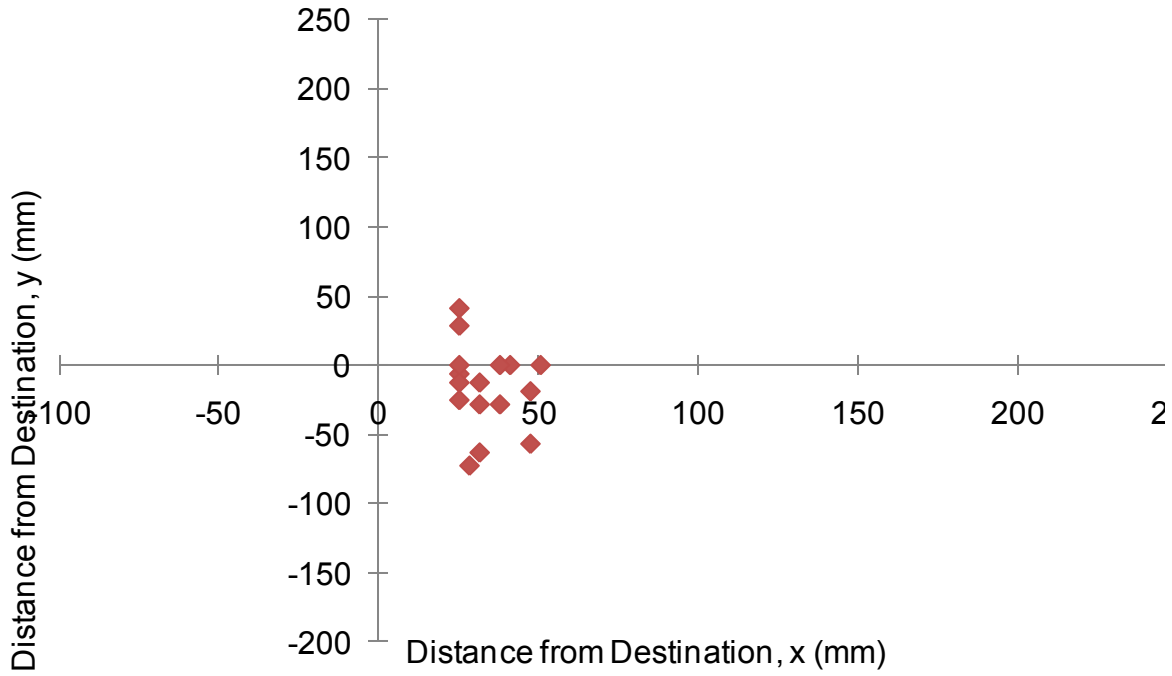

FIGURE 33. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 91.4 cm (3 ft) test of the Arc Compensation Method. The direction of travel relative to the graph is from right to left.

From Figure 33, the algorithm generally directed the tracked vehicle not to travel as far as necessary to reach the target location along the *x* axis and directed the tracked vehicle to travel too far along the *y* axis. From this information, it can be suggested that that algorithm over calculated the return angle in conjunction with operating within a 10 degree range (+/- 5 degrees from measured) when obtaining the return angle, (this range does not include the up to three degree error of the compass) causing the vehicle to underperform. The 10 degree range is necessary because of the low resolution of the compass.

Figure 34 plots the distance to the target location for the 121.9 cm total distance tests.

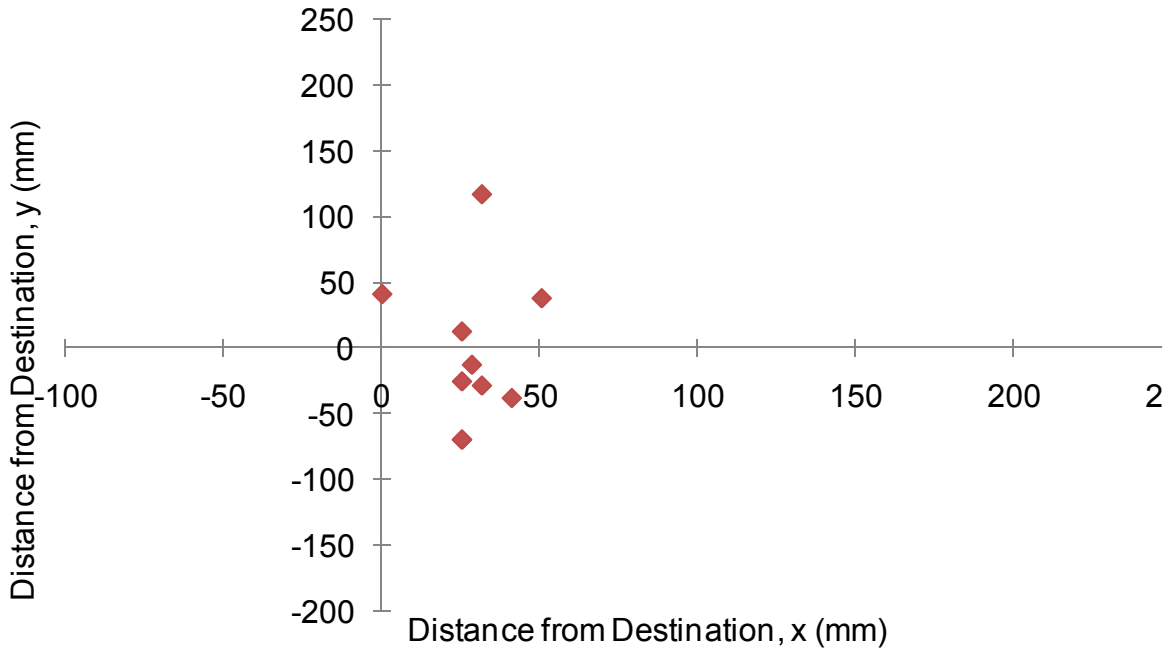

FIGURE 34. The distance from where the vehicle stopped with respect to the target location at coordinate (0, 0) for the 121.9 cm (4 ft) test of the Arc Compensation Method. The direction of travel relative to the graph is from right to left.

From the above figure, the algorithm again consistently did not direct the vehicle to travel far enough in the parallel direction. However, the algorithm equally missed the target along the perpendicular axis by traveling both too far and too short. From this information, it can be determined that the algorithm does not consistently over calculate or under calculate the return angle or that the compass operating range that is allowed is too much. It is thought that methods for determining how far the tracked vehicle traveled during the slip may not be

accurate enough to achieve extremely high resolution results but are very good at low resolution estimation.

Figure 35 plots the distance to the target location for the 182.9 cm total distance tests.

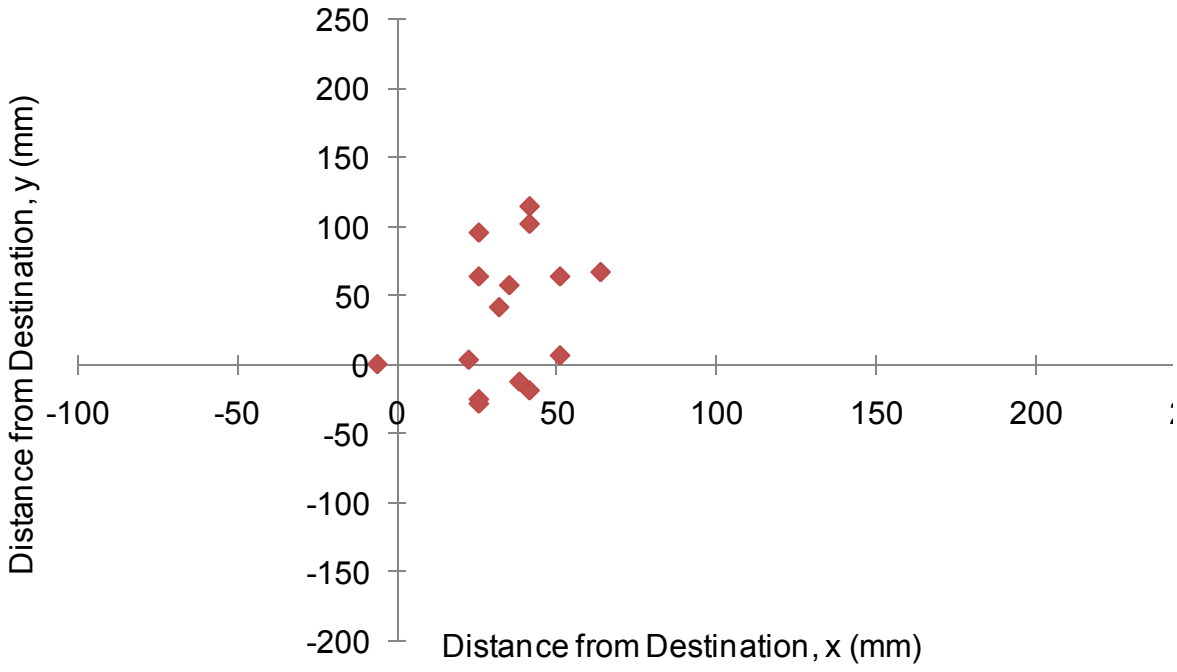

FIGURE 35. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 182.9 cm (6 ft) test of the Arc Compensation Method. The direction of travel relative to the graph is from right to left.

From Figure 35, the suggestion that the algorithm always over calculates the return angle can be completely ruled out. Another factor may be the distance traveled. The short range test consistently over calculated the angle giving

results in the fourth quadrant, the medium range was centered along the *y* axis and the long range test consistently under calculated the angle and gave results in the first quadrant. Figures 33 – 35 show that in each series of tests, the tracked vehicle did not travel the correct distance in the *x* direction. The error does not lie in the angle calculation but in the algorithms ability to measure for how far the vehicle traveled in the *x* direction during a slip. The long range test added more distance to travel and therefore resulted in smaller return angles, where as the short range test had shorter distance to travel and produced larger return angles. Therefore it is the algorithms ability to measure slip travel in the x direction that is in error and not the algorithms ability to place the vehicle on the correct return trajectory. This can also be seen in the series of test in which the Arc Extension Method is used to control the tracked autonomous vehicle.

The series of test with the autonomous tracked vehicle running the Arc Extension Method performed much more poorly than the Arc Compensation Method at achieving the target location. Overall, the Arc Extension Method averaged a radial distance of 80 mm from the target location to the point that the tracked vehicle stopped. After eliminating the outliers, the averaged dropped to 70 mm which is 32% farther than the Arc Extension Method. Figure 36 plots the distance to the target location for all of the tests using the Arc Extension Method.

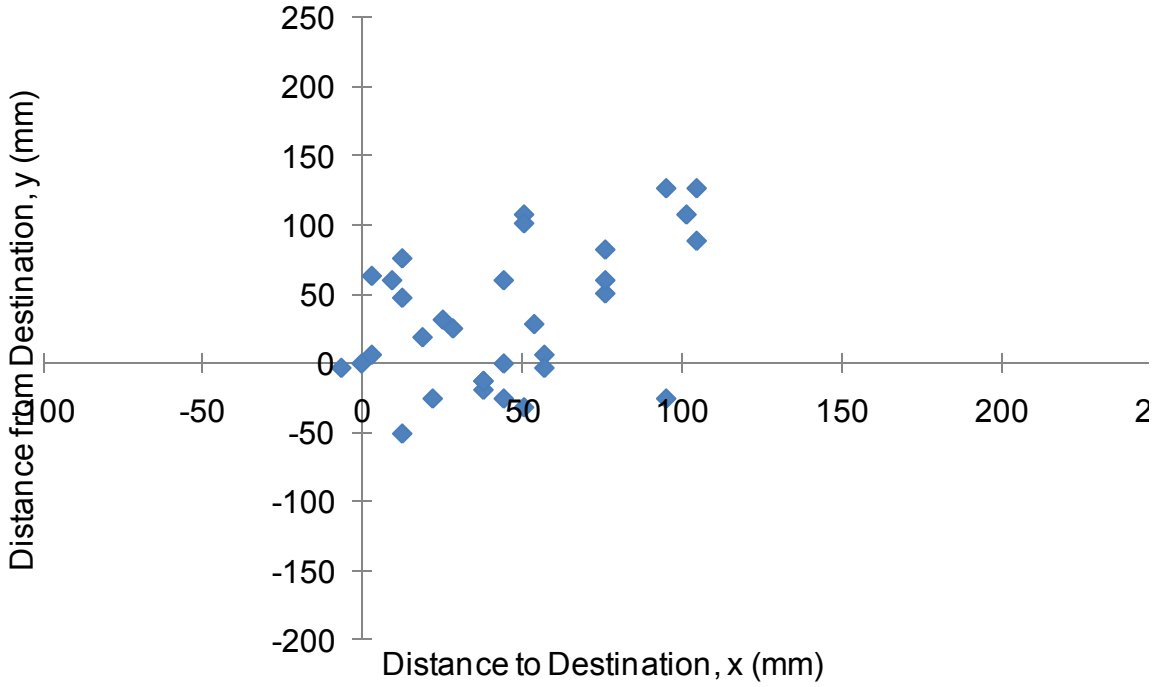

FIGURE 36. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for all tests of the Arc Extension Method. The direction of travel relative to the graph is from right to left.

The values for the Arc Extension Method are less consistent than the Arc Compensation Method. The tracked vehicle most often ended up in the first quadrant and generally stopped short of the target location. However, the algorithm did stop at the (0,0) coordinate. Figure 37 plots the distance to the target location for the 91.4 cm total distance tests for the Arc Extension Method.

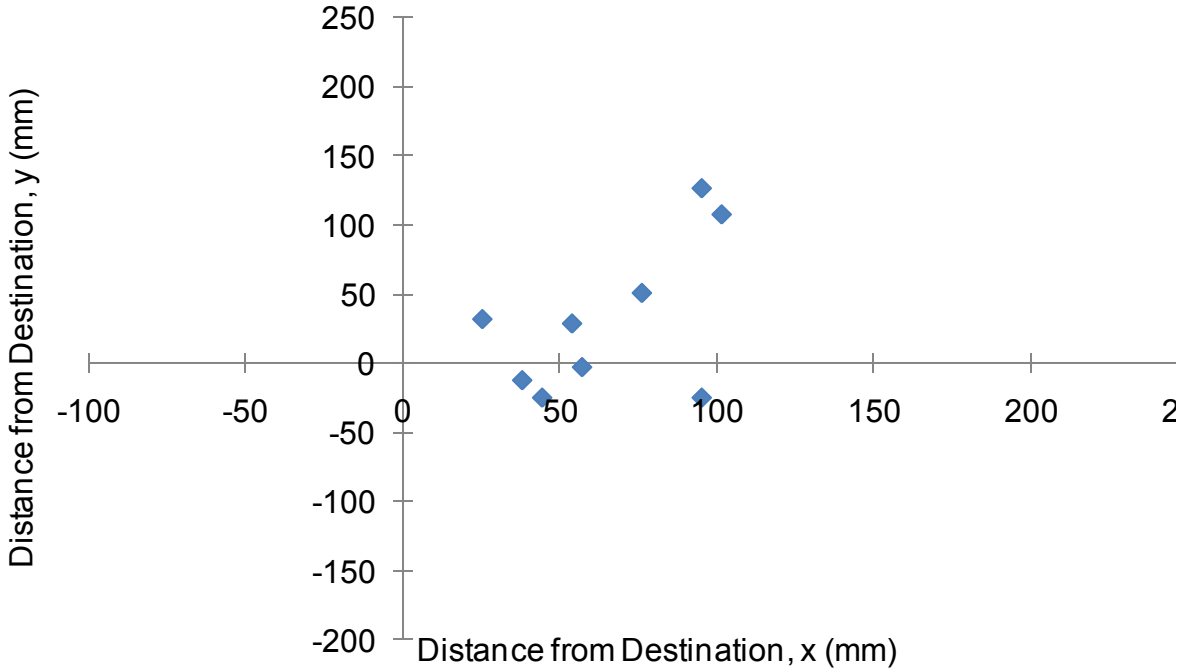

FIGURE 37. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 91.4 cm (3 ft) test of the Arc Extension Method. The direction of travel relative to the graph is from right to left.

From Figure 37, the tracked vehicle consistently did not travel far enough along the *x* direction. For the series of tests in which the vehicle traveled the short initial distance under control of the Arc Extension Method, the vehicle stopped an average of 65 mm away from the target location in the *x* direction and 46 mm away in the *y* direction. Comparatively, when the autonomous tracked vehicle was controlled by the Arc Compensation Method, the tracked vehicle stopped and average of 34 mm away from the target location in the *x* direction and 25 mm away in the *y* direction. The tracked vehicle stopped an average of 83 mm radial distance from the target location when under control of the Arc Extension

Method. By comparison, the autonomous tracked vehicle when controlled by the Arc Compensation Method stopped an average of 46 mm radial distance from the target location, an 80% difference.

Figure 38 plots the distance to the target location for the 121.9 cm total distance tests for the Arc Extension Method..

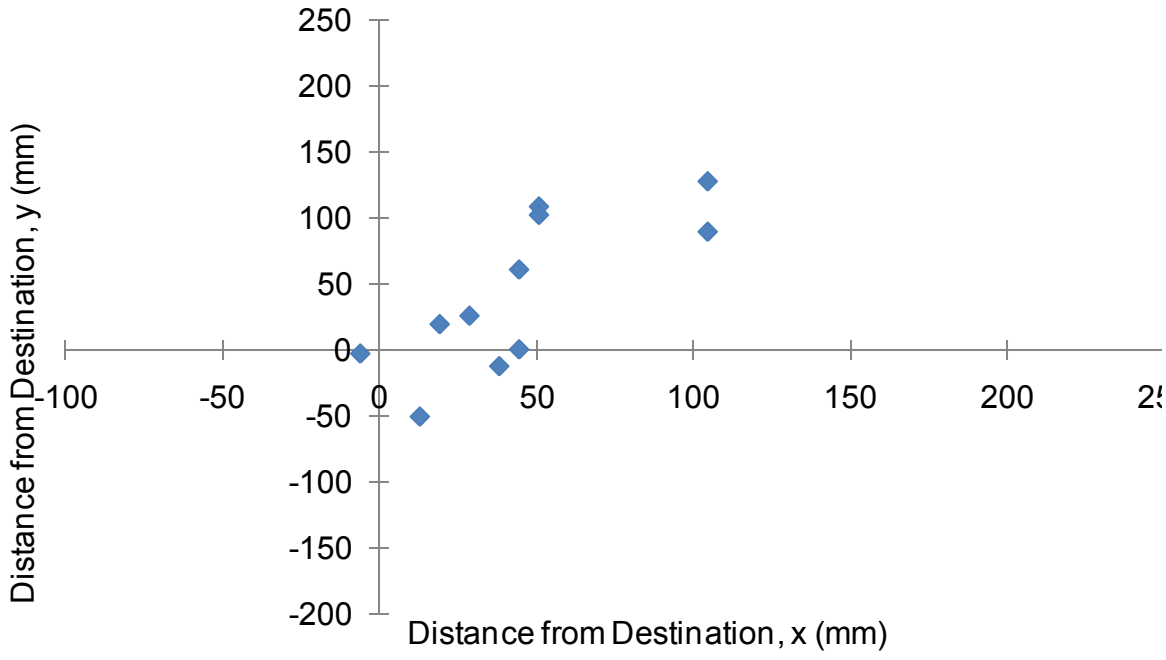

FIGURE 38. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 121.9 cm (4 ft) test of the Arc Extension Method. The direction of travel relative to the graph is from right to left.

From Figure 38, the tracked vehicle stopped an average of 46 mm from the target location in the *x* direction and an average of 54 mm in the *y* direction. An average radial distance less than 74 mm was recorded for this series of test, with a range of 165 mm and a standard deviation of 51 mm. The standard deviation

for the Arc Compensation Method during the 121.9 cm series of test is 30 mm, a 70% difference to the respective Arc Extension Method series of testing.

Figure 39 plots the distance to the target location for the 182.9 cm total distance tests for the Arc Extension Method.

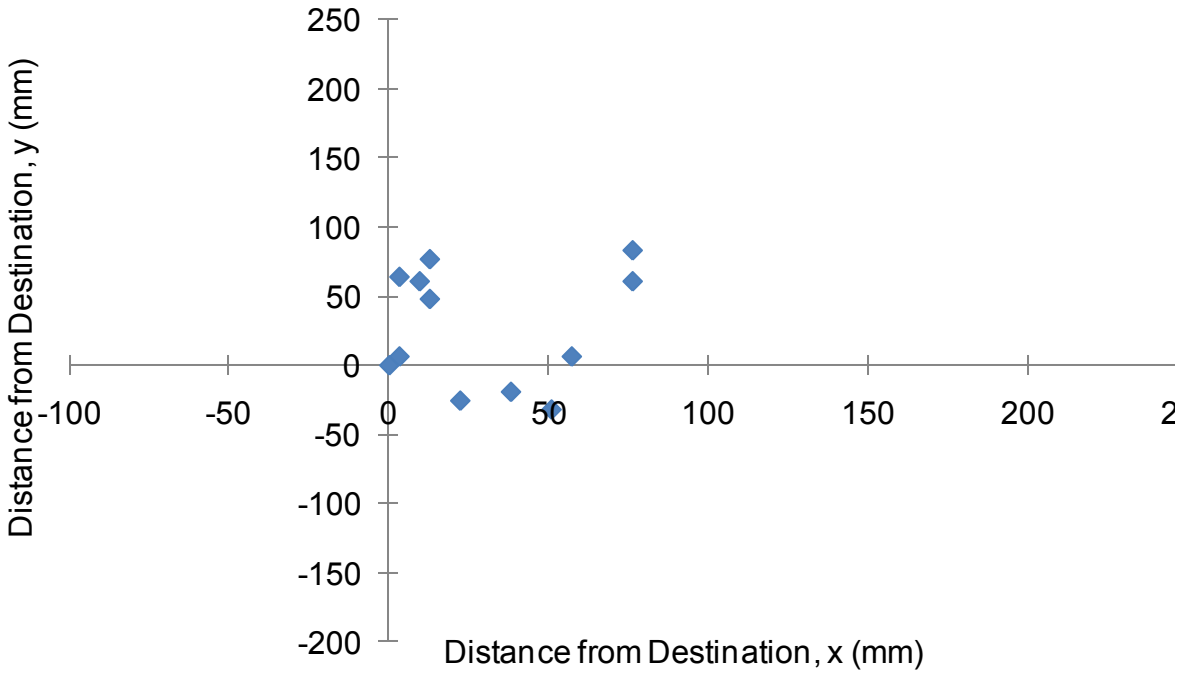

FIGURE 39. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 182.9 cm (6 ft) test of the Arc Extension Method. The direction of travel relative to the graph is from right to left.

For the series of tests, the Arc Extension Method averaged a radial distance of 55 mm, which is smaller than the 62 mm average for the respective series of test for the Arc Compensation Method. The range and the standard deviation for the Arc Extension Method and the Arc Compensation Method are very similar for the 182.9 cm total distance test. The range for the Arc Extension Method is 112 mm,

doing better than the Arc Compensation Method by 10 mm. The standard deviation is 33 mm for both methods.

The Straight Line Slip Method had an average radial distance from where the vehicle stopped to the target location of 78 mm. This is only 8 mm further than the Arc Extension Method and is 25 mm further than the Arc Compensation Method. The Straight Line Slip Method is also the simplest of the three methods. Being the simplest, the Line Slip Method has no ability to measure change in the y direction during a slip. Figure 40 plots the distance to the target location for all of the tests using the Straight Line Slip Method.

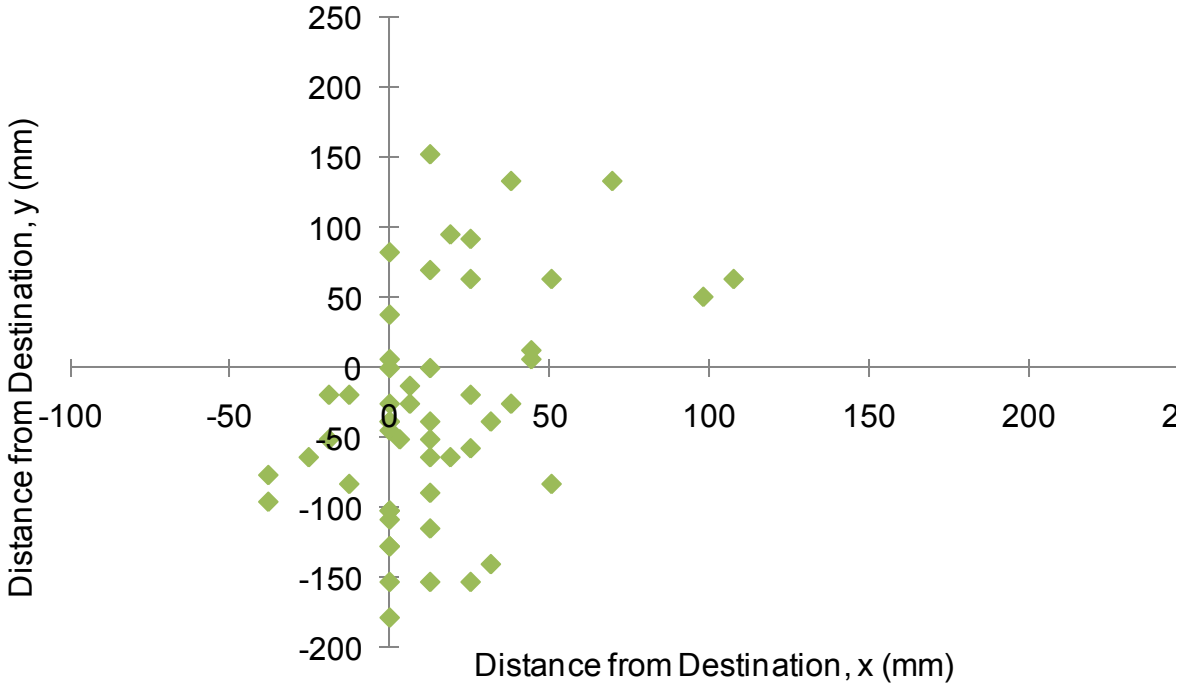

FIGURE 40. The distance from where the vehicle stopped with respect to the target location at coordinate (0, 0) for all tests of the Straight Line Slip Method. The direction of travel relative to the graph is from right to left.

The consistency of the Straight Line Slip Method is poorer than the Arc Compensation Method and the Arc Extension Method. One of the most notable aspects of these results is how often the algorithm directed the tracked vehicle to travel the appropriate distance in the *x* direction, this could because the method does not perform any numerical calculation in relation to the distance traveled in the *x* direction that would shorten that distance such as sine and cosine operations. The average distance in the *x* direction that the tracked vehicle traveled when controlled by the Straight Line Slip Method is 21 mm compared to the 32 mm of the Arc Compensation Method. Although the Straight Line Slip

Method performed very well in directing the tracked vehicle to travel the appropriate distances in the *x* direction, the algorithm performed the worst in determining the distance to travel in the *y* direction. The algorithm had an average distance to the target location in the *y* direction of 71 mm compared to the 37 mm average of the Arc Compensation Method, a 56% difference. The Arc Extension Method averaged 45 mm in the y-direction, a 37% difference compared to the Straight Line Slip Method.

Figure 41 plots the distance to the target location for the 91.4 cm total distance tests for the Straight Line Slip Method.

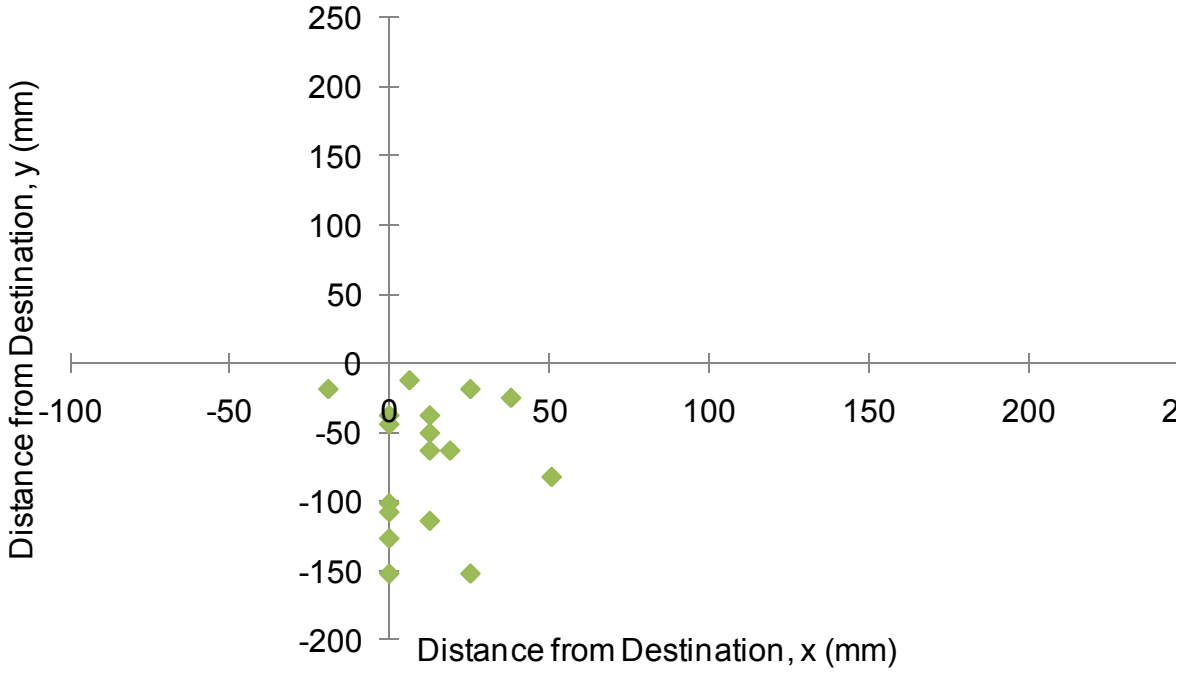

FIGURE 41. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 91.4 cm (3 ft) test of the Straight Line Slip Method. The direction of travel relative to the graph is from right to left.

The Straight Line Slip Method differs the most with its results by having more results in quadrant IV than the other two algorithms. During the 91.4 cm total distance series of test, 100% of the data is in quadrants III and IV indicating that the Straight Line Slip Method routinely measured a larger return angle than was necessary. The reason for this is because the algorithm does not consider how far it travels in the *y* direction during a slip.

Figure 42 plots the distance to the target location for the 121.9 cm total distance tests for the Straight Line Slip Method.

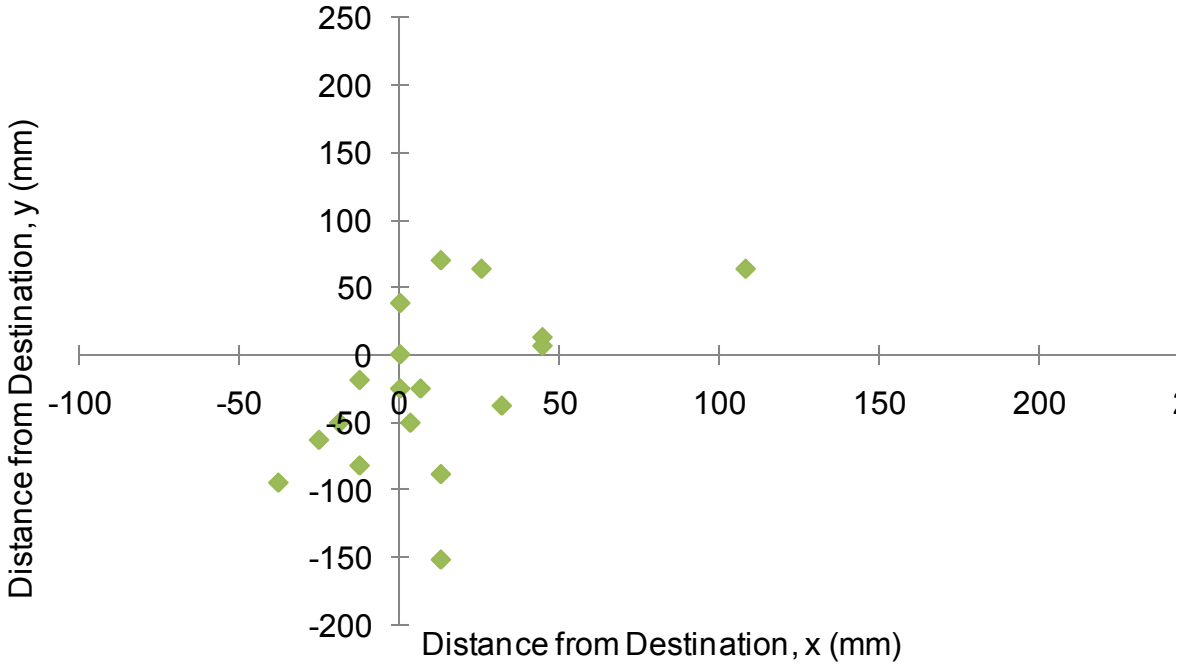

FIGURE 42. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 121.9 cm (4 ft) test of the Straight Line Slip Method. The direction of travel relative to the graph is from right to left.

From Figure 42, the majority of the results are in quadrants III and IV. For all the series of tests with the Straight Line Slip Method controlling the tracked vehicle, 66% of the results are in quadrants III and IV. For the test series with a total travel distance of 121.9 cm, 60% of the data are in quadrants III and IV. One of the most notable pieces of information from Figure 42 is the fact that the vehicle stopped at the coordinate (0,0) during one of its test while under the control of this method.

Figure 43 plots the distance to the target location for the 182.9 cm total distance tests for the Straight Line Slip Method.

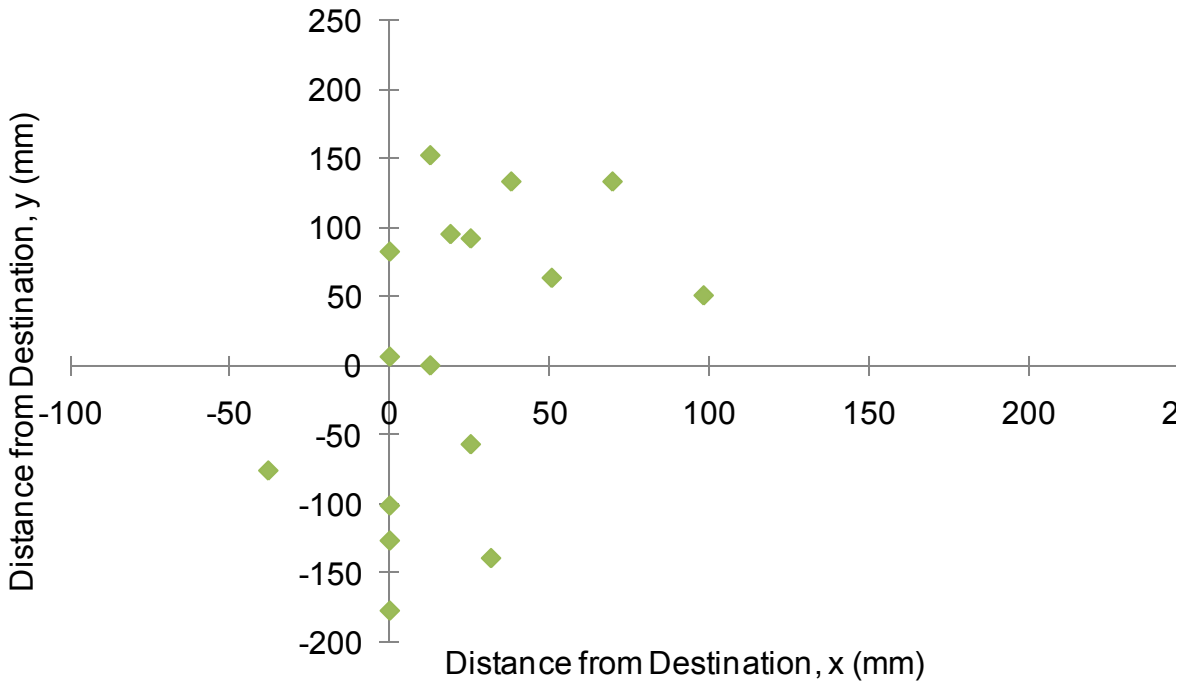

FIGURE 43. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for the 182.9 cm (6 ft) test of the Straight Line Slip Method. The direction of travel relative to the graph is from right to left.

The assumption that the Straight Line Slip Method usually calculates the angle of return as larger than necessary is incorrect. The variance along the *y* axis is due to not measuring or calculating the change in position along the *y* axis during a slip.

From the Arc Extension Method, it was discovered that attempting to interpolate how far the tracked vehicle slipped under the assumption that the radius of the slipping arc is constant will result in inconsistent results. The Straight Line Slip Method generated good insight to estimating the how far the tracked vehicle needs to travel in the *x* direction to reach the target location. This

insight is that it may be better to not tamper with the slip in the x direction measurement. From the Arc Compensation Method, consistent and high accuracy results were produced, but showed the need for better estimating the distance to travel in the *x* direction. All the results demonstrated the need for a higher resolution compass so that the tolerances for the compass could be tightened.

From this data, information and statistics for each algorithm were tabulated and modeled. The following figures document the precision and accuracy of the algorithms by generating circles for varying levels of accuracy. The smaller circles represent a higher level of accuracy. This type of figure will be referenced as a bull's-eye plot. Figures 42-44 plot the distance from the target location to the point that the tracked vehicle stopped within a bull's-eye plot. The plots document the percentage of data that lies within each level of accuracy. The Arc Compensation Method is represented in Figure 44, the Arc Extension Method is represented in Figure 45 and the Straight Line Slip Method is represented in Figure 46.

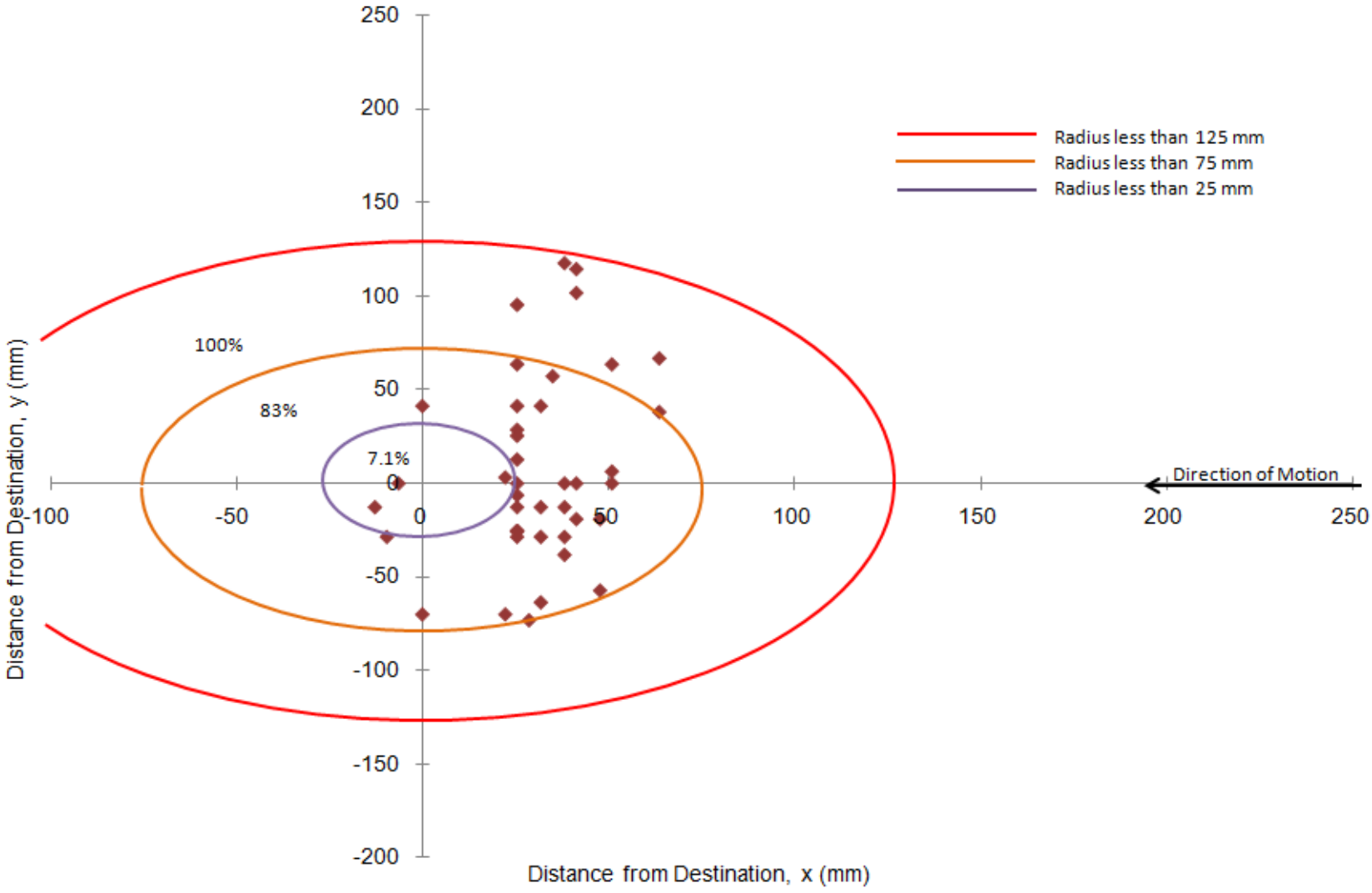

FIGURE 44. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for all tests of the Arc Compensation Method. Five percent of the data lies within 25 mm of the target location, 75% lies within 75 mm of the target location and 100% of the data lies within 125 mm of the target location.

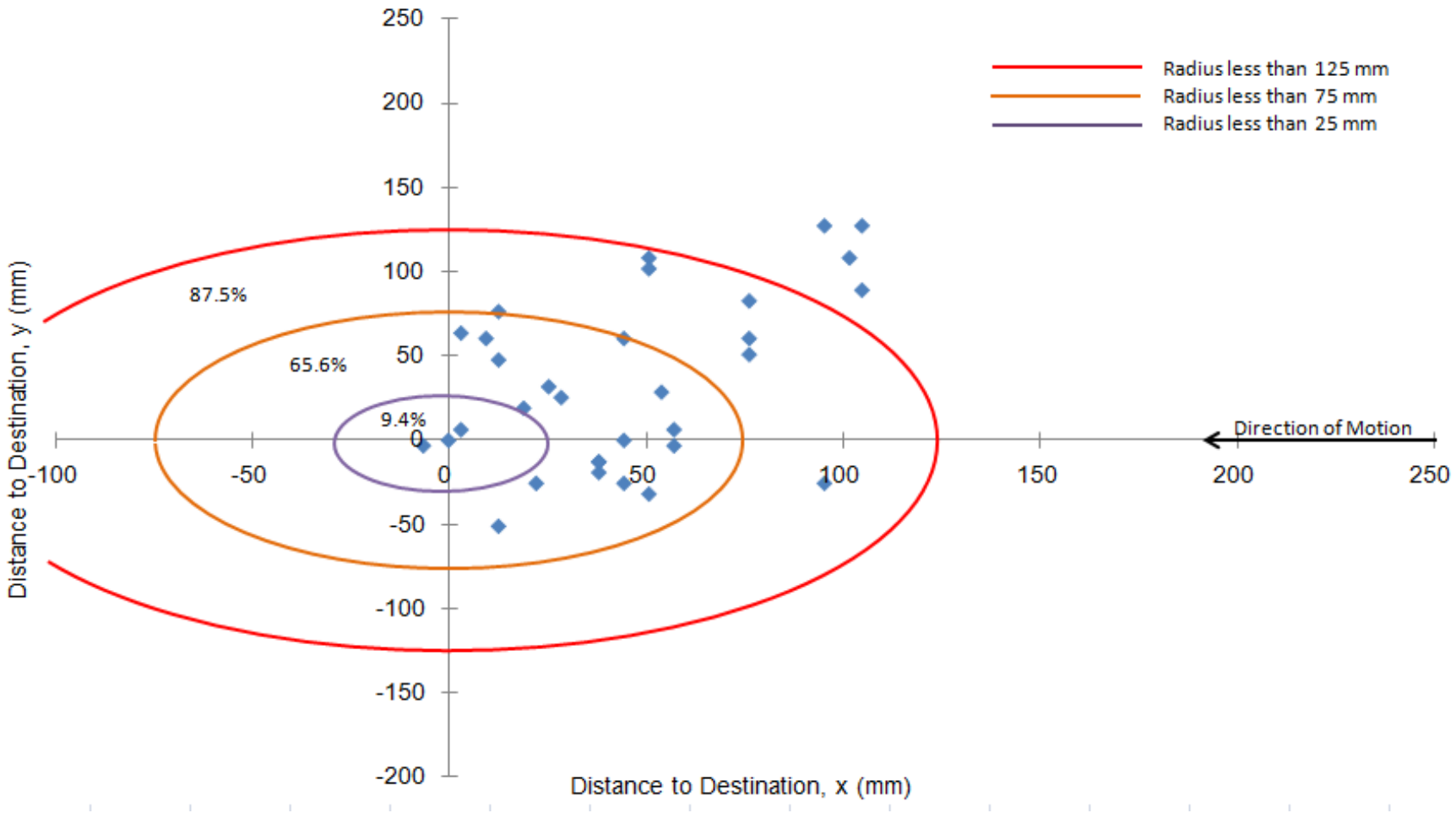

FIGURE 45. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for all tests of the Arc Extension Method. From the data,6.7% lies within 25 mm of the target location, 60% lies within 75 mm of the target location and 81.7% of the data lies within 125 mm of the target location.

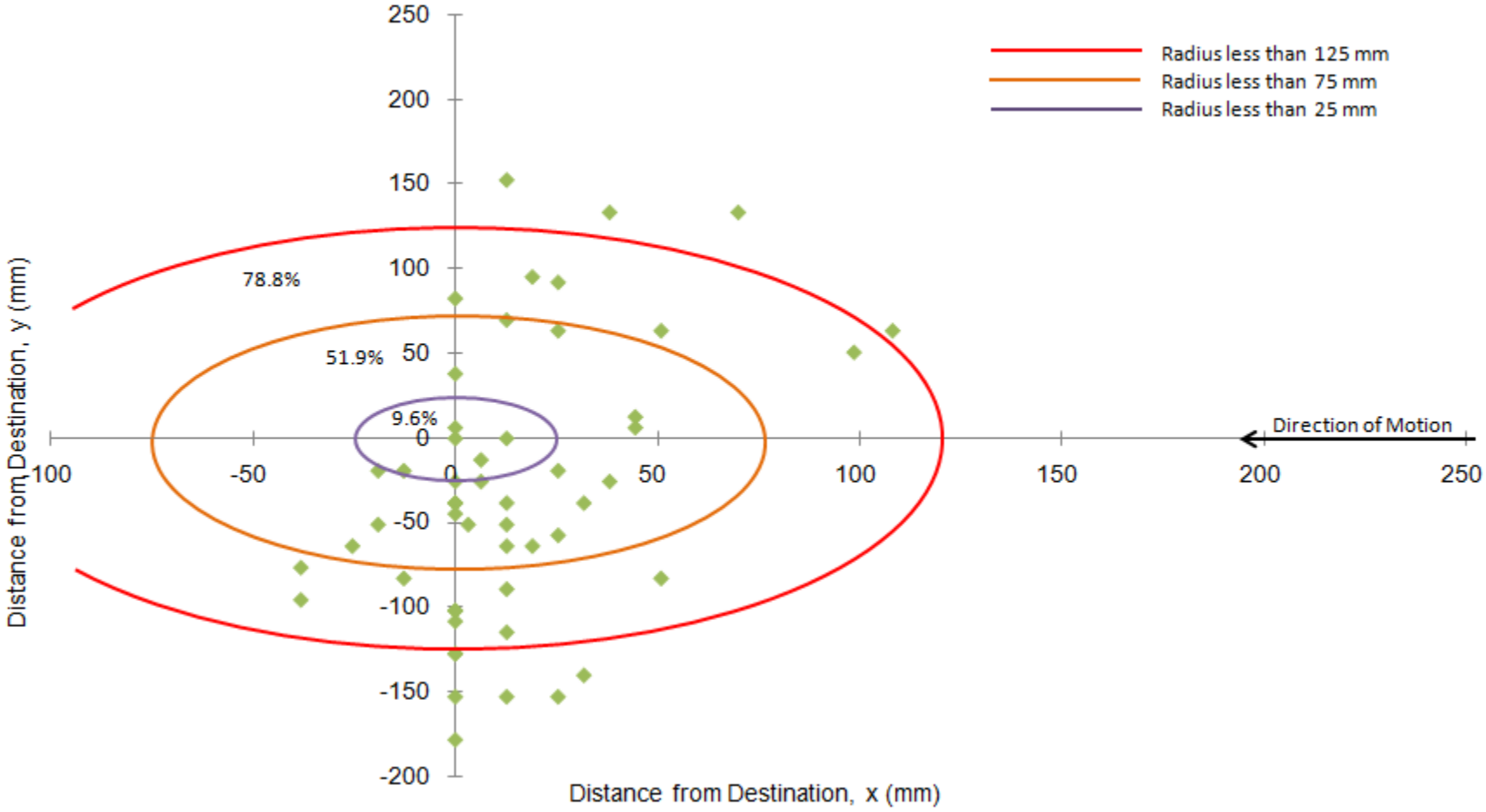

FIGURE 46. The distance from where the tracked vehicle stopped with respect to the target location at coordinate (0, 0) for all tests of the Straight Line Slip Method. From the data, 10% lies within 25 mm of the target location, 53.3% lies within 75 mm of the target location and 78.3% of the data lies within 125 mm of the target location.

The Arc Compensation Method performed the best of the three algorithms at directing the tracked vehicle to the target location. Although the algorithm had the least amount of data points within the circle for the highest level of accuracy, only 7.1% of the data points, the algorithm had 100% of its data points within the circle for the lowest level of accuracy and 83% of its data points within the circle for the middle level of accuracy.

Comparatively from Figure 45, the Arc Extension Method had the same amount data points within the circle for the highest level of accuracy as the Arc Compensation Method. However, the algorithm had only 66% of its data points within the circle for the middle range of accuracy, 18% less than the Arc Compensation Method. When all three tiers of accuracy are considered, 87.5% of the data points lie within this region on the bull's-eye plot. This is a less percentage than the Arc Compensation Method.

The Straight Line Slip Method had the most results in the region for the highest accuracy with 9.6% of the data points at this level. However, this algorithm produced the fewest data points at the two subsequent levels of accuracy with 51.9% of the data points within the circle for the middle level and 78.8% within the circle for the lowest level of accuracy. The level of performance for the Straight Line Slip Method was only slightly lower than the performance level of the Arc Extension Method.

The single most informative statistic that can be measured about each algorithm is the distance to the target location from the point that the tracked

vehicle stopped during each trial. Table III compares each algorithm on the basis of the average, the minimum, the maximum and standard deviation for the distance to the target location from the point that the vehicle stopped.

TABLE III

PERFORMANCE METRICS FOR ALL THREE ALGORITHMS COMPARING THE DISTNACE TO THE TARGET LOCATION TO THE POINT THAT THE VEHICLE STOPPED. ALL UNITS IN MM

	Avg.	Min	Max	StdDev
Arc Extension Method	70	0	165	43
Arc Compensation Method	53	6	123	27
Straight Line Slip Method	78	0	173	46

The Arc Compensation Method has a smaller number in each category, indicating a higher level of performance than the other algorithms except for the minimum distance to the target location from the point that the vehicle stopped.

Although the algorithms are able to meet their objectives, and the tracked vehicle and its instruments provide a very capable platform, all still experience some sources of error. The first notable source of error is the tendency to the tracked vehicle to bear right when traversing a straight trajectory. This is due to the mechanical setup of the tracked vehicle and will vary for each setup. There are two primary drivers to the tendency of bearing right for the tracked vehicle.

The first is the variation in the servo motors that drive each track of the vehicle. Although great care was taken to make them as similar as possible, they cannot be identical and will not drive each track equally. This will also change based on the power level of the batteries driving the servos. The second driver is the track of the vehicle itself. The guide wheels for each track are attached to the chassis by two nuts, a bolt and a washer. The amount of torque on each bolt affects how freely the guide wheels rotate. Each wheel was setup to rotate as freely as possible, however, the freedom of rotation for each wheel varies.

To measure how much this affected the tracked vehicle, a series of 10 test where run in which the vehicle traveled a distance of 182.9 cm and it was recorded how far the vehicle had varied from the straight trajectory at 305 mm increments. Figure 47 documents this tendency by plotting the amount of deviation from a straight line for each measurement during each test.

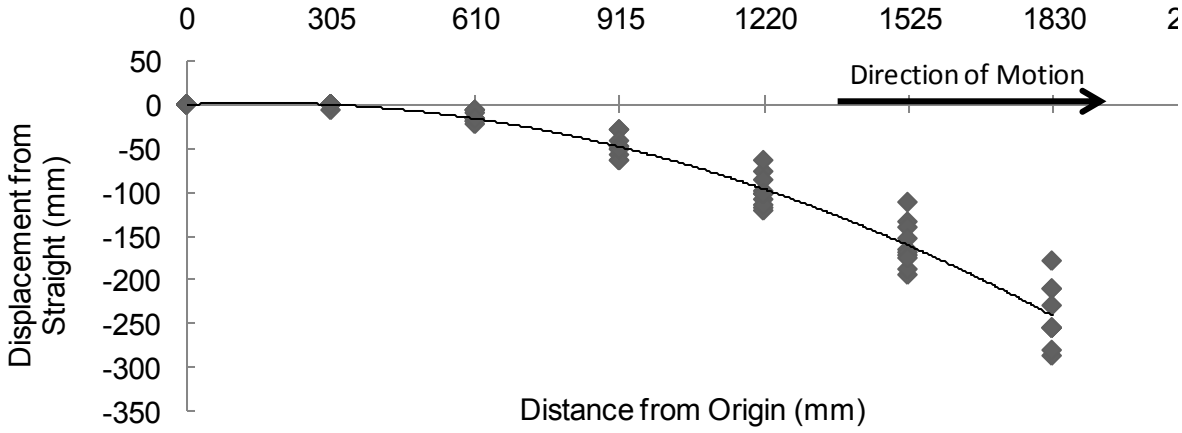

FIGURE 47. Measurements of tracked vehicle position relative to true during a straight line test at 305 mm (1 ft) increments with best fit line for the data. The equation of the best fit line is Y= -9E-05x2+0.0262x+0.2268. The line has an R2 value of 0.9599.

The tracked vehicle's tendency to bear right is slight up to 915 mm (3 ft), where according to the best line equation; the variance is at 51 mm. At the 1219 mm mark, the variance doubles to 102 mm of variance. At 1830 mm (6 ft), the best fit equation predicts a variance of 253 mm. This information suggests that the tracked vehicle is subject to extreme straight line variations when traveling long distances. However, if the tracked vehicle is limited to distances of less than 1 meter during straight line travel, then variances will be slight. These figures will not be the same for each vehicle setup and the results for this project have not been adjusted to account for any straight line variance.

The second notable source of error is the tolerance limits that are built into the algorithm and are the result of a combination of two sources. The compass provides acceptable results but is not precise enough to provide high levels of accuracy. Therefore tolerances had to be built into the algorithms to allow for the

variations resulting from the degree changes that the compass measured and the true amount of change in orientation actually existed. For each algorithm, the tolerance is +/- 5 degrees, which allows for a 10 degree range in which the tracked vehicle can operate in. It is thought that this source of error is the primary limitation on improving the accuracy of the algorithms.

A third source of error is due to the microcontroller used for the tracked vehicle. The Basic Stamp 2, which has its limitations stated above, only allows for whole number math and therefore cannot perform division and multiplication that results in parts of number. The microcontroller will perform the math and then round down to the next whole number and report that number as its result. This will affect such calculations as determining how far the tracked vehicle must travel and cause small errors while covering distances.

Overall, all three algorithms met the objective of directing the tracked vehicle to the target location. The Arc Compensation Method proved to be the most capable algorithm for directing the tracked vehicle to the target location. The tracked vehicle has a length dimension of 210 mm. For the Arc Compensation Method, 92.8% of the trials resulted with at least 50% of the length of the tracked vehicle being within a half body length radius (105 mm) of the target location.

VIII. CONCLUSIONS

Three methods were created and tested to find which method was better able to direct a robot to the target location, detect changes in position and orientation as a result of a slip condition and correct for changes associate with a slip. The Arc Compensation Method produced the best results at achieving the target destination. The Line Slip Method produced the worst results at achieving the target location but was the most capable at calculating the correct distance to travel in the *x*-direction. The Arc Extension Method produced results that were in between the other two methods; however, this method produced more realistic results for detecting the arc length.

As the arc increases, the vehicles ability to achieve the target destination decreases. This applies to both the Arc Extension Method and the Arc Compensation Method and was expected since the greater the arc length, the greater the displacement of the slip. The Straight Line Slip Method does not consider the arc length to determine the amount of slip incurred.

As the change in orientation increases for a slip, the ability to achieve the target location will increase. This proves that the initial assumption of a constant

radius arc formed during a slip is incorrect. Therefore, the use of this assumption results in poor arc length estimations for slips with varying rates of rotation.

With the elimination of arc length outliers, the Arc Extension Method gives a more realistic distribution of arc lengths. The Arc Compensation Method tends to report shorter arc lengths; this was expected since the Arc Extension Method includes the arc length prior to crossing the slip threshold. Without the elimination of arc length outliers, the Arc Extension Method produces many values that exceed expectation and are not realistic. These results that exceed expectation are due to the initial assumption of a constant radius arc. Both methods produced a significant amount of zero length arc lengths and is a result of each methods ability to determine when the tracked vehicle has ceased to slip.

IX. FUTURE WORK

For future applications of the Arc Compensation Method, the platform should include a compass with significantly better resolution than the one used for this project. This is so the orientation tolerances can have a smaller operating range and provide greater accuracy in orientation measurement, especially when the autonomous tracked vehicle is moving. In conjunction with this recommendation, the ability for the compass to operate in a noisy environment with multiple strong magnetic fields should be increased with shielding or by some other method. One of the most difficult aspects of this project was to find a suitable testing place that was void of a lot of electronics and far from large quantities of metal that altered the Earth's magnetic field. Another suggested implementation is the use of a microcontroller that is capable of math using partial numbers to reduce any error associated with not being able to use partial numbers. A method of detecting slip should be created that does not rely on the constant radius arc assumption to improve the accuracy of the arc length detection. Lastly, a method should be created from the best components of the three methods presented.

X. WORKS CITED

Ashtech Technology. (2010). *RTK (Real Time Kinematic)*. Retrieved April 20, 2010, from Magellangps.com: http://pro.magellangps.com/en/products/aboutgps/rtk.asp

Bian, H., Jin, Z., & Tian, W. (2005). Study on GPS attitude determination system aided INS using adaptive Kalman filter. *Measurement Science and Technology* , 2072-2079.

Byun, S., Hajj, G., & Young, L. (2002). *Assessment of GPS Signal Multipath Interface.* San Diego, CA: Jet Propulsion Laboratory, California Institute of Technology.

Earth Measurement Consulting. (n.d.). *GPS Accuracy and Limitations*. Retrieved Feb 16, 2010, from Earth Measurement and Cooling: www.earthmeasurement.com/GPS_accuracy.html

Martin, J., Williams, J., Gracey, K., Alvarez, A., & Lindsay, S. (2005). *BASIC Stamp Syntax and Reference Manual Version 2.2.* Rocklin, CA: Parallax.

Ojeda, L., Cruz, D., Reina, G., & Borenstein, J. (2006). Current-Based Slippage Detection and Odometry Correction for Mobile Robots and Planetary Rovers. *IEEE Transactions on Robotics* , 366-378.

Parallax. (2010). *Parallax.com*. Retrieved March 21, 2010, from www.parallax.com

Parallax, Inc. (2005, May). *Hitachi® HM55B Compass Module (#29123).* Retrieved April 11, 2010, from Parallax.com: http://www.parallax.com/Portals/0/Downloads/docs/prod/compshop/HM55BModDocs.pdf

Pilgrim, P. C. (2004, April 11). *Applying the Boe-Bot Digital Encoder Kit (#28107).* Retrieved April 11, 2010, from Parallax.com: http://www.parallax.com/Portals/0/Downloads/docs/prod/datast/ApplyEncoder.pdf

Reina, G., Ojeda, L., Milella, A., & Borenstein, J. (2006). Wheel Slippage and Sinkage Detection for Planetary Rovers. *IEEE/ASME Transactions on Mechatronics* , 185-195.

The Robocup Federation. (2010). *RoboCup*. Retrieved April 11, 2010, from www.robocup.org

The Robocup Federation. (2010). *RoboCup Rescue*. Retrieved March 7, 2010, from http://www.robocuprescue.org/

Thurn, S. (2007). *Junior: The Stanford Entry in the Urban Grand Challenge.* Stanford Artifical Intelligence Lab, Stanford University.

Thurn, S. (2006). Stanley: The Robot that won the DARPA Grand Challenge. *Journal of Field Robotics* .

Thurn, S., & Montemerlo, M. (2005). GraphSLAM Algorithm with Applications to Large Scale Mapping of Urban Structures. *The Internation Journal of Robotics Research* (25), 403-429.

Yang, Q., & Jianmin, S. (2007, Nov 3). A Location Method for Autonomous Vehicle Based on Integrated GPS/INS. *IEEE Xplore* .

APPENDIX I

FINDING ε_X AND ε_Y LENGTHS

$$\beta = \theta \tag{I1}$$

where β is the arc angle and θ is measured from the compass readings. The radius of the arc in degrees is found using

$$r = \left(\frac{L}{\theta}\right)\left(\frac{180}{\pi}\right) \tag{I2}$$

where *L* can be either the total arc length as used by the Arc Extension Method or the partial arc length, $L_{p,}$, as used by the Arc Compensation Method. The length is calculated using

$$z = r\cos\theta \tag{I3}$$

Since the radius of the arc is already known, ε_x and ε_y lengths are found using

$$\varepsilon_y = r - z = r - r\cos\theta \quad (14)$$

$$\varepsilon_x = r\sin\theta \quad (15)$$

The resultant is seen in Figure 7. By subtracting ε_x and the distance traveled before the slip from the distance from the origin to the destination, the distance to the target location in the *x* direction is determined. Using Equation 5 and the distance to the destination in the *x* direction, a new orientation to the destination is set and a new traveling distance is calculated using Pythagorean's Theorem.

APPENDIX II

Test Data

		Equation 1?	Distance to Slip (mm)	Distance from Origin	Nominal Angle (deg)
Test #	1	Y	50.8	3' (84 eps)	82
Test #	2	Y	50.8	3' (84 eps)	76
Test #	3	Y	50.8	3' (84 eps)	76
Test #	4	Y	50.8	3' (84 eps)	78
Test #	5	Y	50.8	3' (84 eps)	79
Test #	6	Y	50.8	3' (84 eps)	80
Test #	7	Y	50.8	3' (84 eps)	80
Test #	8	Y	50.8	3' (84 eps)	77
Test #	9	Y	50.8	3' (84 eps)	79
Test #	10	Y	50.8	3' (84 eps)	79
Test #	11	Y	50.8	4' (112 eps)	78
Test #	12	Y	50.8	4' (112 eps)	79
Test #	13	Y	50.8	4' (112 eps)	79
Test #	14	Y	50.8	4' (112 eps)	80
Test #	15	Y	50.8	4' (112 eps)	76
Test #	16	Y	50.8	4' (112 eps)	82
Test #	17	Y	50.8	4' (112 eps)	77
Test #	18	Y	50.8	4' (112 eps)	77
Test #	19	Y	50.8	4' (112 eps)	79
Test #	20	Y	50.8	4' (112 eps)	79
Test #	21	Y	152.4	6' (168 eps)	77
Test #	22	Y	152.4	6' (168 eps)	81
Test #	23	Y	152.4	6' (168 eps)	78
Test #	24	Y	152.4	6' (168 eps)	79
Test #	25	Y	152.4	6' (168 eps)	78
Test #	26	Y	152.4	6' (168 eps)	79
Test #	27	Y	152.4	6' (168 eps)	78
Test #	28	Y	152.4	6' (168 eps)	79
Test #	29	Y	152.4	6' (168 eps)	79
Test #	30	Y	152.4	6' (168 eps)	78
Test #	31	Y	152.4	3' (84 eps)	81
Test #	32	Y	152.4	3' (84 eps)	79
Test #	33	Y	152.4	3' (84 eps)	78
Test #	34	Y	152.4	3' (84 eps)	79
Test #	35	Y	152.4	3' (84 eps)	78
Test #	36	Y	152.4	3' (84 eps)	78
Test #	37	Y	152.4	3' (84 eps)	77
Test #	38	Y	152.4	3' (84 eps)	78
Test #	39	Y	152.4	3' (84 eps)	80
Test #	40	Y	152.4	3' (84 eps)	80
Test #	41	Y	152.4	4' (112 eps)	82
Test #	42	Y	152.4	4' (112 eps)	82
Test #	43	Y	152.4	4' (112 eps)	80

Test #	44	Y	152.4	4' (112 eps)	80
Test #	45	Y	152.4	4' (112 eps)	75
Test #	46	Y	152.4	4' (112 eps)	76
Test #	47	Y	152.4	4' (112 eps)	74
Test #	48	Y	152.4	4' (112 eps)	76
Test #	49	Y	152.4	4' (112 eps)	73
Test #	50	Y	152.4	4' (112 eps)	76
Test #	51	Y	254	6' (168 eps)	76
Test #	52	Y	254	6' (168 eps)	77
Test #	53	Y	254	6' (168 eps)	79
Test #	54	Y	254	6' (168 eps)	75
Test #	55	Y	254	6' (168 eps)	75
Test #	56	Y	254	6' (168 eps)	74
Test #	57	Y	254	6' (168 eps)	75
Test #	58	Y	254	6' (168 eps)	75
Test #	59	Y	254	6' (168 eps)	74
Test #	60	Y	254	6' (168 eps)	74
Test #	61	N	50.8	3' (84 eps)	78
Test #	62	N	50.8	3' (84 eps)	80
Test #	63	N	50.8	3' (84 eps)	77
Test #	64	N	50.8	3' (84 eps)	78
Test #	65	N	50.8	3' (84 eps)	77
Test #	66	N	50.8	3' (84 eps)	80
Test #	67	N	50.8	3' (84 eps)	78
Test #	68	N	50.8	3' (84 eps)	77
Test #	69	N	50.8	3' (84 eps)	80
Test #	70	N	50.8	3' (84 eps)	78
Test #	71	N	50.8	4' (112 eps)	81
Test #	72	N	50.8	4' (112 eps)	77
Test #	73	N	50.8	4' (112 eps)	80
Test #	74	N	50.8	4' (112 eps)	81
Test #	75	N	50.8	4' (112 eps)	78
Test #	76	N	50.8	4' (112 eps)	80
Test #	77	N	50.8	4' (112 eps)	78
Test #	78	N	50.8	4' (112 eps)	81
Test #	79	N	50.8	4' (112 eps)	77
Test #	80	N	50.8	4' (112 eps)	81
Test #	81	N	152.4	6' (168 eps)	80
Test #	82	N	152.4	6' (168 eps)	79
Test #	83	N	152.4	6' (168 eps)	80
Test #	84	N	152.4	6' (168 eps)	79
Test #	85	N	152.4	6' (168 eps)	79
Test #	86	N	152.4	6' (168 eps)	79
Test #	87	N	152.4	6' (168 eps)	79
Test #	88	N	152.4	6' (168 eps)	79
Test #	89	N	152.4	6' (168 eps)	81
Test #	90	N	152.4	6' (168 eps)	78

Test #	91	N	152.4	3' (84 eps)	77
Test #	92	N	152.4	3' (84 eps)	78
Test #	93	N	152.4	3' (84 eps)	79
Test #	94	N	152.4	3' (84 eps)	78
Test #	95	N	152.4	3' (84 eps)	78
Test #	96	N	152.4	3' (84 eps)	78
Test #	97	N	152.4	3' (84 eps)	81
Test #	98	N	152.4	3' (84 eps)	81
Test #	99	N	152.4	3' (84 eps)	78
Test #	100	N	152.4	3' (84 eps)	79
Test #	101	N	152.4	4' (112 eps)	78
Test #	102	N	152.4	4' (112 eps)	79
Test #	103	N	152.4	4' (112 eps)	78
Test #	104	N	152.4	4' (112 eps)	79
Test #	105	N	152.4	4' (112 eps)	79
Test #	106	N	152.4	4' (112 eps)	78
Test #	107	N	152.4	4' (112 eps)	78
Test #	108	N	152.4	4' (112 eps)	76
Test #	109	N	152.4	4' (112 eps)	77
Test #	110	N	152.4	4' (112 eps)	78
Test #	111	N	254	6' (168 eps)	78
Test #	112	N	254	6' (168 eps)	79
Test #	113	N	254	6' (168 eps)	78
Test #	114	N	254	6' (168 eps)	77
Test #	115	N	254	6' (168 eps)	77
Test #	116	N	254	6' (168 eps)	77
Test #	117	N	254	6' (168 eps)	78
Test #	118	N	254	6' (168 eps)	79
Test #	119	N	254	6' (168 eps)	77
Test #	120	N	254	6' (168 eps)	77
Test #	121	Straight	50.8	3' (84 eps)	79
Test #	122	Straight	50.8	3' (84 eps)	79
Test #	123	Straight	50.8	3' (84 eps)	82
Test #	124	Straight	50.8	3' (84 eps)	82
Test #	125	Straight	50.8	3' (84 eps)	79
Test #	126	Straight	50.8	3' (84 eps)	80
Test #	127	Straight	50.8	3' (84 eps)	82
Test #	128	Straight	50.8	3' (84 eps)	79
Test #	129	Straight	50.8	3' (84 eps)	80
Test #	130	Straight	50.8	3' (84 eps)	78
Test #	131	Straight	50.8	4' (112 eps)	80
Test #	132	Straight	50.8	4' (112 eps)	79
Test #	133	Straight	50.8	4' (112 eps)	80
Test #	134	Straight	50.8	4' (112 eps)	81
Test #	135	Straight	50.8	4' (112 eps)	76
Test #	136	Straight	50.8	4' (112 eps)	78
Test #	137	Straight	50.8	4' (112 eps)	80

Test #	138	Straight	50.8	4' (112 eps)	79
Test #	139	Straight	50.8	4' (112 eps)	82
Test #	140	Straight	50.8	4' (112 eps)	79
Test #	141	Straight	152.4	6' (168 eps)	79
Test #	142	Straight	152.4	6' (168 eps)	81
Test #	143	Straight	152.4	6' (168 eps)	80
Test #	144	Straight	152.4	6' (168 eps)	79
Test #	145	Straight	152.4	6' (168 eps)	81
Test #	146	Straight	152.4	6' (168 eps)	81
Test #	147	Straight	152.4	6' (168 eps)	81
Test #	148	Straight	152.4	6' (168 eps)	80
Test #	149	Straight	152.4	6' (168 eps)	80
Test #	150	Straight	152.4	6' (168 eps)	77
Test #	151	Straight	152.4	3' (84 eps)	78
Test #	152	Straight	152.4	3' (84 eps)	78
Test #	153	Straight	152.4	3' (84 eps)	77
Test #	154	Straight	152.4	3' (84 eps)	78
Test #	155	Straight	152.4	3' (84 eps)	78
Test #	156	Straight	152.4	3' (84 eps)	78
Test #	157	Straight	152.4	3' (84 eps)	79
Test #	158	Straight	152.4	3' (84 eps)	79
Test #	159	Straight	152.4	3' (84 eps)	77
Test #	160	Straight	152.4	3' (84 eps)	76
Test #	161	Straight	152.4	4' (112 eps)	76
Test #	162	Straight	152.4	4' (112 eps)	82
Test #	163	Straight	152.4	4' (112 eps)	80
Test #	164	Straight	152.4	4' (112 eps)	78
Test #	165	Straight	152.4	4' (112 eps)	78
Test #	166	Straight	152.4	4' (112 eps)	78
Test #	167	Straight	152.4	4' (112 eps)	79
Test #	168	Straight	152.4	4' (112 eps)	78
Test #	169	Straight	152.4	4' (112 eps)	78
Test #	170	Straight	152.4	4' (112 eps)	77
Test #	171	Straight	254	6' (168 eps)	77
Test #	172	Straight	254	6' (168 eps)	79
Test #	173	Straight	254	6' (168 eps)	75
Test #	174	Straight	254	6' (168 eps)	77
Test #	175	Straight	254	6' (168 eps)	77
Test #	176	Straight	254	6' (168 eps)	77
Test #	177	Straight	254	6' (168 eps)	76
Test #	178	Straight	254	6' (168 eps)	77
Test #	179	Straight	254	6' (168 eps)	77
Test #	180	Straight	254	6' (168 eps)	78

Arc Length (eps)	Measured Radius	Measured x length	Measured y length	Actual Slip Angle (deg)	Measured Slip Angle (deg)	Difference btwn A/M
0	0	0	0	13	9	4
1	4	2	1	15	15	0
80	286	82	12	15	16	1
16	57	16	2	16	16	0
26	62	56	36	20	24	4
0	0	0	0	12	11	1
17	57	55	42	19	17	2
15	39	0	0	25	22	3
25	84	81	61	20	17	3
7	16	2	0	20	25	5
4	16	16	14	11	14	3
0	0	0	0	14	13	1
0	0	0	0	16	14	2
18	38	37	27	18	27	9
1	4	2	1	15	15	0
9	27	4	0	20	19	1
4	19	10	3	12	12	0
12	38	29	13	16	18	2
2	8	5	2	18	15	3
9	27	4	0	19	19	0
0	0	0	0	12	13	1
32	115	33	5	16	16	0
14	38	32	17	16	21	5
4	14	4	1	16	16	0
48	172	49	7	15	16	1
1	4	2	1	15	15	0
3	17	9	3	18	10	8
25	84	81	61	17	17	0
1	4	2	1	12	15	3
0	0	0	0	11	13	2
22	57	1	0	29	22	7
0	0	0	0	18	16	2
5	29	16	5	8	10	2
0	0	0	0	17	10	7
64	229	66	10	16	16	0
0	0	0	0	9	12	3
10	27	23	12	15	21	6
48	172	49	7	15	16	1
0	0	0	0	16	14	2
0	0	0	0	14	14	0
1	4	2	0	16	13	3
8	23	21	14	29	20	9
1	5	5	5	16	11	5

16	38	35	22	18	24	6
29	62	59	44	22	27	5
16	57	16	2	15	16	1
25	57	8	1	22	25	3
48	120	101	56	21	23	2
1	4	2	0	15	13	2
2	5	1	0	22	25	3
14	28	18	7	28	29	1
12	24	16	6	25	29	4
16	46	42	27	18	20	2
0	0	0	0	15	15	0
32	92	84	54	18	20	2
14	73	73	73	9	11	2
0	0	0	0	15	17	2
3	12	12	11	14	14	0
12	28	4	0	23	25	2
33	59	33	10	28	32	4
0	0	0	0	16	14	2
6	18	3	0	20	19	1
1	7	7	6	16	8	8
0	0	0	0	19	14	5
12	33	27	15	24	21	3
1	3	0	0	20	19	1
0	0	0	0	19	18	1
4	18	7	2	15	13	2
3	9	1	0	21	19	2
6	29	15	4	14	12	2
1	4	1	0	19	16	3
1	3	0	0	22	22	0
21	33	33	29	32	36	4
0	0	0	0	16	14	2
8	29	8	1	16	16	0
0	0	0	0	12	9	3
2	7	2	0	18	16	2
5	24	13	4	16	12	4
0	0	0	0	11	12	1
1	4	4	4	17	14	3
5	24	13	4	15	12	3
0	0	0	0	15	16	1
2	6	5	2	16	18	2
0	0	0	0	15	12	3
4	13	13	10	15	17	2
2	5	0	0	21	22	1
1	4	1	0	18	16	2
1	4	2	1	15	15	0
6	25	24	21	14	14	0
0	0	0	0	8	7	1

5	13	0	0	21	22	1
3	14	8	2	14	12	2
8	23	21	14	19	20	1
4	19	10	3	11	12	1
1	8	5	2	12	7	5
4	11	10	7	18	20	2
2	7	2	0	16	16	0
8	22	18	10	23	21	2
11	33	5	0	19	19	0
2	9	4	1	15	13	2
1	14	11	5	15	4	11
4	29	28	24	11	8	3
2	38	5	0	9	3	6
7	22	17	8	22	18	4
1	14	11	5	9	4	5
10	26	0	0	24	22	2
10	20	13	5	24	29	5
2	8	5	2	15	15	0
0	0	0	0	15	8	7
10	34	32	24	19	17	2
2	7	2	0	14	16	2
2	13	5	1	11	9	2
3	10	10	7	20	17	3
0	0	0	0	11	9	2
0	0	0	0	11	9	2
0	0	0	0	15	14	1
4	21	21	21	11	11	0
16	40	34	19	21	23	2
2	11	6	2	11	10	1
1	14	11	5	9	4	5
				20	20	0
				26	24	2
				22	20	2
				18	17	1
				20	23	3
				22	21	1
				13	11	2
				23	23	0
				20	21	1
				30	37	7
				25	25	0
				26	25	1
				16	16	0
				20	17	3
				14	14	0
				12	9	3
				18	16	2

17	16	1
18	17	1
20	17	3
20	19	1
23	25	2
11	10	1
15	13	2
15	16	1
15	14	1
15	12	3
25	22	3
18	16	2
18	17	1
16	15	1
14	13	1
20	20	0
20	20	0
18	20	2
20	21	1
25	26	1
15	16	1
16	14	2
20	19	1
15	12	3
23	25	2
17	15	2
17	17	0
19	17	2
21	17	4
18	11	7
15	13	2
15	14	1
12	15	3
24	24	0
26	12	14
20	18	2
17	14	3
28	29	1
31	33	2
19	14	5
17	14	3
13	13	0
14	11	3

Actual Final Orientation (deg)	Measured Final Orientation (deg)	Difference btwn A/M	x-cord (mm)	y-cord (mm)	Radius (mm)
5	4	1	25	-6	26
7	8	1	102	108	148
11	11	0	76	57	95
11	11	0	76	51	92
22	22	0	25	38	46
6	4	2	25	32	41
11	12	1	95	127	159
12	16	4	102	140	173
11	14	3	92	76	120
13	14	1	64	51	81
5	5	0	51	108	119
5	4	1	105	127	165
6	4	2	51	102	114
13	12	1	127	200	237
8	5	3	105	89	137
9	2	7	105	76	130
9	4	5	51	102	114
8	7	1	44	60	75
5	4	1	29	25	38
6	7	1	44	0	44
4	2	2	76	60	97
5	5	0	25	44	51
6	5	1	70	-44	83
3	2	1	76	83	112
5	4	1	22	51	55
2	2	0	38	-19	43
8	2	6	102	191	216
4	5	1	48	25	54
3	2	1	57	6	58
4	2	2	0	0	0
20	25	5	70	-25	74
10	9	1	38	-13	40
6	8	2	44	-25	51
8	5	3	25	0	25
17	16	1	64	-16	65
7	7	0	95	-25	99
20	15	5	51	25	57
15	12	3	25	-6	26
5	7	2	54	29	61
10	7	3	57	-3	57
7	5	2	38	-13	40
8	8	0	41	-10	42
3	4	1	102	0	102

10	12	2	175	-19	176
18	18	0	213	-51	219
10	8	2	-6	-3	7
15	16	1	0	0	0
17	15	2	48	35	59
7	5	2	13	-51	52
11	9	2	19	19	27
8	8	0	13	76	77
9	8	1	6	-29	29
7	5	2	3	64	64
3	2	1	3	6	7
8	9	1	32	51	60
5	4	1	51	-32	60
2	5	3	10	60	61
5	4	1	13	48	49
7	7	0	22	-25	34
14	12	2	95	117	151
9	7	2	41	0	41
12	12	0	29	-73	78
7	5	2	0	-64	64
7	7	0	19	-70	72
10	18	8	64	-76	99
13	11	2	25	-6	26
8	8	0	51	0	51
8	8	0	32	-64	71
12	11	1	32	-13	34
9	8	1	25	-25	36
7	5	2	38	-38	54
8	8	0	-13	-13	18
21	19	2	-38	-114	120
5	4	1	22	-70	73
7	7	0	64	38	74
2	2	0	38	117	123
7	7	0	-10	-29	30
6	5	1	13	-102	102
6	4	2	25	-25	36
8	5	3	0	41	41
5	2	3	64	67	92
3	2	1	41	-19	45
3	4	1	38	-13	40
6	1	5	38	51	64
6	4	2	25	-25	36
6	4	2	25	-29	38
5	2	3	25	95	99
6	2	4	51	95	108
4	2	2	25	64	68
3	0	3	41	102	110

15	15	0	25	29	38
9	11	2	32	-29	43
22	19	3	25	-13	28
8	8	0	48	-19	51
6	4	2	51	-3	51
13	15	2	25	0	25
10	12	2	38	0	38
23	21	2	38	-29	48
21	19	2	48	-57	74
6	5	1	25	41	48
6	0	6	48	73	87
6	4	2	0	-70	70
1	0	1	0	-102	102
9	8	1	51	51	72
6	0	6	38	13	40
10	11	1	25	13	28
13	14	1	0	51	51
9	5	4	48	25	54
2	1	1	38	-79	88
9	8	1	25	25	36
6	4	2	51	6	51
1	1	0	-6	0	6
6	4	2	32	41	52
4	1	4	41	64	76
5	1	2	51	64	81
4	2	2	41	114	122
2	11	9	25	-25	36
7	7	0	22	3	22
1	1	0	35	57	67
5	0	5	32	48	57
11	13	2	-19	-19	27
15	15	0	51	-83	97
14	19	5	25	-6	26
7	6	1	0	-127	127
8	9	1	38	-25	46
7	9	2	25	-152	155
5	4	1	0	-152	152
16	14	2	13	-114	115
8	7	1	0	-102	102
19	19	0	0	-51	51
8	7	1	44	13	46
8	7	1	25	64	68
4	5	1	-19	-51	54
6	5	1	44	6	45
4	4	0	13	-152	153
5	2	3	-25	-64	68
6	4	2	13	-89	90

7	4	3	-13	-19	23
7	9	2	-13	-83	84
4	5	1	-38	-95	103
6	2	4	25	-95	99
5	4	1	0	-102	102
4	0	4	13	25	28
4	2	2	32	-140	143
5	2	3	0	83	83
1	2	1	-38	-76	85
3	1	2	0	-127	127
3	4	1	0	-178	178
3	2	1	25	-57	63
3	2	1	13	152	153
5	7	2	13	-64	65
5	7	2	13	-38	40
10	10	0	25	-19	32
13	12	1	0	-38	38
13	11	2	19	-64	66
12	11	1	6	-13	14
13	16	3	0	-38	38
7	9	2	0	-44	44
7	10	3	13	-51	52
10	12	2	0	-108	108
7	4	3	108	64	125
8	8	0	0	38	38
5	4	1	32	-38	50
8	5	3	0	-25	25
6	5	1	0	0	0
6	5	1	13	0	13
3	2	1	6	-127	127
5	4	1	3	-51	51
4	4	0	6	-25	26
6	4	2	13	70	71
6	5	1	70	133	151
5	1	4	16	-140	141
7	4	3	0	6	6
4	2	2	19	95	97
4	5	1	13	0	13
4	5	1	98	51	111
3	2	1	19	25	32
3	2	1	25	92	96
4	2	2	51	64	81
2	1	1	38	133	139

APPENDIX III

Source Code

```
' Slip_Detection_rev7_Arc Compensation Method_Final.bs2
' {$STAMP BS2}
' {$PBASIC 2.5}

'-----[Description]-------------------------------------------------
'
'This algorithim is designed to detect slip occurrences by sensing
'the orientation of the tracked vehicle with the use of a compass.
'If the compass senses a change in orientation, the robot will
'know that a slip has occurred and will correct for the slip by
'changing its path in order to reach the original destination.
'
'-----[Ownership]---------------------------------------------------
'
'Aaron Stewart
'Department of Mechanical Engineering
'J. B. Speed School of Engineering
'University of Louisville
'
'*****************************************************************************
'*-------------------------[Program]-------------------------------------------*
'*****************************************************************************
'
'-----[EEPROM Data]-------------------------------------------------
CompassOffsets DATA @ 0, (4)                'Stores x and y axis offsets
CompassLowVal  DATA      (1)                'Stores index of lowest angle
CompassCal     DATA      (16)               '16 reference compass angles

'-----[Encoder Variables]-------------------------------------------
REncoValue     VAR Bit                      'Right encoder reading
LEncoValue     VAR Bit                      'Left encoder reading
RCounter       VAR Byte                     'Right Counter, can oly count to 256
LCounter       VAR Byte                     'Left Counter, can only count to 256
PrevR          VAR Bit
PrevL          VAR Bit

'-----[Compass Variables]-------------------------------------------
x              VAR      Word                ' x-axis data
y              VAR      Word                ' y-axis data
status         VAR      Nib                 ' Status flags
angle          VAR      Word                ' Angle measurement
axisOffset     VAR      angle               ' Axis offset

index          VAR      Status              ' EEPROM index
table          VAR      Byte(2)             ' Stores EEPROM table values
span           VAR      x                   ' Span between table entries
angleOffset    VAR      y                   ' Offset btwn measured and table

'-----[Other Variables]---------------------------------------------
TotalDist      VAR      Word                'Begining distance to the final destination.
i              VAR      Byte                'Counter
nomAngle       VAR      Word                'Begining angle
Dist           VAR      Byte                'How far the robot has travled.
arcLength      VAR      Word                'Another distance.
b              CON      6                   'distance from center of robot to center of track.
  2.4 inches, or 5 eps
sigma          VAR      Word
xcord          VAR      Byte
ycord          VAR      Byte
CW             CON      0                   'Indicates Clockwise
CCW            CON      1                   'Indicates Counter Clockwise
dir            VAR      Bit                 'Direction indicator.
prevAngle      VAR      Word
thresh         CON      15                  'Slip Threshold

'-----[Compass Constants]-------------------------------------------
```

```
Reset           CON     %0000                   ' Reset command for HM55B
Measure         CON     %1000                   ' Start measurement command
Report          CON     %1100                   ' Get status/axis values command
Ready           CON     %1100                   ' 11 -> Done, 00 -> no errors
NegMask         CON     %1111100000000000       ' For 11-bit negative to 16-bits
current         CON     0                       ' Table array index
previous        CON     1                       ' Table array index


'-----[Pin Definitions]-------------------------------------------
REnco           CON     0                       'Right encoder pin
LEnco           CON     1                       'Left encoder pin

nInp            PIN     12                      'Define the input pin for ServoPal

DinDout         PIN     11                      'P11 transceives to/from Din/Dout
Clk             PIN     10                      'P10 sends pulses to HM55B's Clk
En              PIN      9                      'P9 controls HM55B's /EN(ABLE)

'-----[Initialization]--------------------------------------------
INPUT nInp                                      'Make sure nInp isn't being driven.
DO : LOOP UNTIL nInp                            'Wait for ServoPAL to power up.

LOW nInp                                        'Set pin to an output and hold it low
PAUSE 100                                       'for 100mS to reset ServoPAL.
HIGH nInp                                       'Raise the pin.
PAUSE 100

RCounter = 0                                    'Initialize encoder values
LCounter = 0

'-----[Centering Routine]-----------------------------------------------------
'DEBUG "Centering...",CR                         'Makes sure the the encoders start
PAUSE 500                                       'at the same position everytime.
GOSUB CheckEnco
IF (REncoValue = 1) THEN
 PULSOUT nInp, 500
 DO UNTIL (REncoValue = 0)                      'Pulse right servo
  GOSUB CheckEnco
 LOOP
 PULSOUT nInp, 2000                             'Stop Servos
ENDIF
IF (LEncoValue = 1) THEN
  PULSOUT nInp, 4
  PULSOUT nInp, 1000                            'Pulse left servo
 DO UNTIL (LEncoValue = 0)
  GOSUB CheckEnco
 LOOP
 PULSOUT nInp, 2000                             'Stop Servos
 PULSOUT nInp, 2000
ENDIF
'DEBUG "Centered."
PAUSE 1000

'-----[Main Program]-----------------------------------------------------------------------
nomAngle = 0                                    'Initializes nomAngle
FOR i = 0 TO 9
  GOSUB Compass
  nomAngle = nomAngle + angle                   'Compiles angles so an average can be taken.
NEXT
nomAngle = nomAngle/10                          'Averages angles and sets the nominal angle.
DEBUG DEC ? nomAngle

DEBUG "How far would you like to go (in eps)? "      'An ep is one "click" on the encoder. One
    ep is 1.089 cm
DEBUGIN DEC TotalDist
```

```
'TotalDist = 100
PAUSE 1000

PULSOUT nInp, 500                           'Pulse both servos to move forward
PULSOUT nInp, 1000                          '(left servo)

DO UNTIL ((angle > nomAngle + thresh) OR (angle < nomAngle - thresh)) 'Allows a small variance from nominal.
 GOSUB Compass
 GOSUB CheckEnco
 'DEBUG DEC ? RCounter
 IF Rcounter = TotalDist THEN GOTO ending    'If the robot encounters no slip and reaches destination then quit moving.
LOOP

Dist = Rcounter                             'Distance robot has traveled so far.
DEBUG "Distance to Slip: ", DEC Dist,CR
RCounter = 0                                'Reset encoder values
LCounter = 0

DEBUG "slipping...",CR

slipping:
DO UNTIL (angle = prevAngle)                'Do untill repeated results.
 GOSUB CheckEnco
 'DEBUG DEC ? RCounter, DEC ? LCounter
 IF (angle <> prevAngle) THEN               'Save old angle as new angle for comparison.
 prevAngle = angle
 ENDIF
 'DEBUG DEC ? prevAngle
 PAUSE 100                                  'Pause for 1/10 of a second to allow for any change in orientation
 GOSUB Compass                              'Get new angle
LOOP

PULSOUT nInp, 2000                          'Stop Servos
PULSOUT nInp, 2000

FOR i = 0 TO 9
  GOSUB Compass                             'Compiles angles so an average can be taken.
  Sigma = Sigma + angle                     'Here, Sigma is an average angle.
NEXT
angle = sigma/10                            'Averages angles.

'Doing some math for the slip...

IF angle > nomAngle THEN                    'Slipped and rotated CW
 angle = angle - nomAngle
 arcLength = (LCounter)
 dir = CW
ELSEIF angle < nomAngle THEN                'Slipped and rotated CCW
 angle = nomAngle - angle
 arcLength = (RCounter)
 dir = CCW
ENDIF
DEBUG SDEC ? angle, DEC ? arcLength
'arcLength = arcLength + ((thresh * arcLength)/ (angle - thresh))     'Arc modification
'DEBUG DEC ? arcLength

sigma = arcLength  * 1800 / (angle * 31)    'Here, sigma is the radius of the arc.
angle = angle * 32/45                       'Convert from degrees to brads.
ycord = sigma - (sigma*(COS angle) / 127)
xcord = sigma*(SIN angle) / 127

Totaldist = TotalDist - xcord - Dist        'Convert total distance to travel to distance remmaining.
```

```
193
194 DEBUG "Radius= ", SDEC sigma,CR, SDEC ? ycord, SDEC ? xcord, "Dist remaining= ", SDEC TotalDist,
    CR
195
196 RCounter = 0                                   'Reset encoder values
197 LCounter = 0
198 PULSOUT nInp, 500                              'Move forward
199 PULSOUT nInp, 1000
200
201 'DEBUG "Clearing slip.",CR
202
203 DO UNTIL (RCounter = 28)                       'Move forward 12 inches to clear slip.
204   GOSUB CheckEnco
205 LOOP
206
207 PULSOUT nInp, 2000                             'Stop Servos
208 PULSOUT nInp, 2000
209
210 DEBUG "Take slip angle measurement.",CR
211 PAUSE 15000                                    'Pause for 15 seconds to take measurements.
212
213 'DEBUG "Slip cleared.",CR
214 'Doing some math for the clearance...
215
216 ycord = ((28 * SIN angle) / 127) + ycord      'In brads
217 xcord = (28 * COS angle) / 127
218 DEBUG SDEC ? ycord, SDEC ? xcord
219
220 Totaldist = Totaldist - xcord                  'Total distance left in the x-dir.
221 DEBUG SDEC ? TotalDist
222 sigma = TotalDist ATN ycord                    'Redefine sigma to be the angle (in brads) that
    robot must rotate from nominal to correct from slip.
223 'DEBUG "Degree of slip (brad): ", SDEC sigma,CR
224 sigma = sigma */ 360                           'Convert sigma to degrees.
225 Totaldist = Totaldist HYP ycord                'Total distance from current position.
226 DEBUG "Degree of Correction: ", SDEC sigma,CR, SDEC ? TotalDist
227
228 'DEBUG "Aligning to original destination.",CR
229
230 IF dir = CW THEN
231 PULSOUT nInp, 700                              'Rotate CCW to recover from CW slip.
232 PULSOUT nInp, 700
233 sigma = nomAngle - sigma                       'Redefine sigma to be the complete angle that the
    robot must rotate to go to destination.
234 ELSEIF dir = CCW THEN
235 PULSOUT nInp, 800                              'Rotate CW to recover from CCW slip.
236 PULSOUT nInp, 800
237 sigma = nomAngle + sigma                       'Redefine sigma to be the complete angle that the
    robot must rotate to go to destination.
238 ENDIF
239 IF sigma.BIT15 = 1 THEN                        'Allows it to calculate across 360 degree boundary.
240 sigma = 360 + sigma
241 DEBUG "Sigma neg.",CR
242 ENDIF
243
244 DEBUG "Angle to Destination: ", SDEC sigma,CR
245
246 DO UNTIL ( (sigma + 5) >= angle) AND (angle >= (sigma - 5) )
247   GOSUB Compass
248   'DEBUG "Compass angle: ", DEC angle,CR
249 LOOP
250
251 PULSOUT nInp, 2000                             'Stop Servos
252 PULSOUT nInp, 2000
253
254 DEBUG "Take final orientation measurement.",CR
```

```
255 PAUSE 15000                                 'Pause for 15 seconds to take measurements.
256
257 PAUSE 100
258 RCounter = 0                                'Reset encoders
259 LCounter = 0
260
261 DEBUG "Go to original destination.",CR, DEC ? TotalDist
262
263 PULSOUT nInp, 500                           'Move forward
264 PULSOUT nInp, 1000
265
266 DO UNTIL (Rcounter = TotalDist)             'Go to original destination.
267   GOSUB CheckEnco
268 LOOP
269 'DEBUG "Arrived.",CR
270
271 ending:
272 PULSOUT nInp, 2000                          'Stop Servos
273 PULSOUT nInp, 2000
274
275 PAUSE 150
276
277 IF dir = CCW THEN
278 PULSOUT nInp, 700                          'Rotate CCW to begining orientation.
279 PULSOUT nInp, 700
280 ELSEIF dir = CW THEN
281 PULSOUT nInp, 800                          'Rotate CW to begining orientation.
282 PULSOUT nInp, 800
283 ENDIF
284
285 DO UNTIL (nomAngle + 5 > angle) AND (nomAngle - 5 < angle)
286   GOSUB Compass
287   'DEBUG "Compass angle: ", DEC angle,CR
288 LOOP
289 DEBUG "Robot has arrived.",CR
290
291
292 PULSOUT nInp, 2000                          'Stop Servos
293 PULSOUT nInp, 2000
294 END
295
296
297 '-----[Subroutines]-------------------------------------------------------
298
299 CheckEnco:                                  'Checks both encoders
300 REncoValue = INS.LOWBIT(REnco)
301 IF REncoValue ^ PrevR THEN
302 RCounter = Rcounter + 1
303 PrevR = REncoValue
304 ENDIF
305 LEncoValue = INS.LOWBIT(LEnco)
306 IF LEncoValue ^ PrevL THEN
307 LCounter = LCounter + 1
308 PrevL = LEncoValue
309 ENDIF
310 RETURN
311
312 Compass:                                   'Checks Compass
313 GOSUB Compass_Get_Axes                     ' Get x, and y values
314 GOSUB Compass_Correct_Offsets              ' Correct axis offsetes
315 angle = x ATN -y                           ' Convert x and y to brads
316 GOSUB Compass_Interpolate                  ' Linear interpolation
317 angle = angle */ 360                       ' Convert brads to degrees
318 'DEBUG DEC ? angle
319 RETURN
320
```

```
' -----[ Subroutine - Compass_Get_Axes ]-------------------------------------

' This subroutine handles BASIC Stamp - HM55B communication and stores the
' magnetic field strength measurements returned by the device in the x and
' y axis variables.

Compass_Get_Axes:                               ' Compass module subroutine

  HIGH En: LOW En                               ' Send reset command to HM55B
  SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]

  HIGH En: LOW En                               ' HM55B start measurement command
  SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
  status = 0                                    ' Clear previous status flags

  DO                                            ' Status flag checking loop
    HIGH En: LOW En                             ' Measurement status command
    SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
    SHIFTIN  DinDout,clk,MSBPOST,[Status\4]     ' Get Status
  LOOP UNTIL status = Ready                     ' Exit loop when status is ready

  SHIFTIN  DinDout,clk,MSBPOST,[x\11,y\11]      ' Get x & y axis values
  HIGH En                                       ' Disable module

  IF (y.BIT10 = 1) THEN y = y | NegMask         ' Store 11-bits as signed word
  IF (x.BIT10 = 1) THEN x = x | NegMask         ' Repeat for other axis

  RETURN

' -----[ Subroutine - Compass_Correct_Offsets ]-------------------------------

' This subroutine corrects cumulative magnetic field interference that can
' come from sources such as the PCB, jumper wires, a nearby battery, or a
' nearby current source.  This subroutine relies on values stored in
' the EEPROM space that was reserved by the CompassOffsets DATA directive.
' These EEPROM values were written by CalibrateHM55BCompass.bs2.

Compass_Correct_Offsets:

  READ CompassOffsets, Word axisOffset          ' Get x-axis offset
  x = x - axisOffset                            ' Correct x-axis
  READ CompassOffsets + 2, Word axisOffset      ' Get y-axis offset
  y = y - axisOffset                            ' Correct y-axis

  RETURN

' -----[ Subroutine - Compass_Interpolate ]-----------------------------------

' This subroutine applies linear interpolation to the refine the compass
' measurement.  This second level of refinement can be performed after the
' Compass_Correct_Offsets subroutine, and it can correct axis skew and other
' errors inherent to the HM55B chip.
'
' The subroutine relies on sixteen actual compass measurements that were stored
' in the sixteen EEPROM locations reserved by the CompassCal DATA directive.
' These measurements were stored by CalibrateHM55BCompass.bs2, and they
' represent the actual compass measurements for 0, 22.5, 45, 90,..., 337.5
' degrees.  The subroutine finds the two EEPROM measurements that the current
' angle measurement falls between.  It then updates the angle measurement
' based on where the angle measurement falls between the two known table values.

Compass_Interpolate:

  ' Start with the lowest value in the CompassCal table.

  READ CompassLowVal, index
```

```
387
388   ' Load current and previous table values.
389
390   READ CompassCal + index, table(current)
391   READ (CompassCal + (index - 1 & $F)), table(previous)
392
393
394   ' The IF...ELSEIF...ELSE...ENDIF code block finds the two EEPROM CompassCal
395   ' table values that the current angle measurement falls between and calculates
396   ' the difference between the current angle measurement and the lower of the
397   ' two table values.  The IF and ELSEIF blocks deal with values that are
398   ' greater than the highest or less than the lowest table values.  The ELSE
399   ' block everything between the highest and lowest table values.
400
401   IF (angle >= table(previous)) THEN
402     span = (255 - table(previous)) + table(current)
403     angleOffset = angle - table(previous)
404   ELSEIF (angle <= table(current)) THEN
405     span = table(current) + (255 - table(previous))
406     angleOffset = angle + (255 - table(previous))
407   ELSE
408     index = index - 1
409     READ CompassCal + index, table(current)
410     DO
411       table(previous) = table(current)
412       index = index + 1
413       READ CompassCal + index, table(current)
414       IF (angle <= table(current)) AND (angle > table(previous)) THEN
415         span = table(current) - table(previous)
416         angleOffset = angle - table(previous)
417         EXIT
418       ENDIF
419     LOOP
420   ENDIF
421
422   ' After the offset between the current angle measurement and the next lower
423   ' table measurement has been determined, this code block uses it along with
424   ' the span between the table entries above and below the angle measurement
425   ' to solve for: angle(corrected) = angle(offset) * 16 / span.
426   ' This code block also rounds up or down by comparing the remainder of
427   ' the angleOffset / span division to the value of (span / 2).
428
429   angleOffset = angleOffset * 16
430   angle = (angleOffset / span) + ((angleOffset // span) / (span / 2))
431   angle = ((index - 1 & $F) * 16) + angle
432   angle = angle & $ff
433
434   RETURN
```

```
' Slip_Detection_rev7_Arc Extension Method_Final.bs2
' {$STAMP BS2}
' {$PBASIC 2.5}

'-----[Description]-------------------------------------------------
'
'This algorithim is designed to detect slip occurrences by sensing
'the orientation of the tracked vehicle with the use of a compass.
'If the compass senses a change in orientation, the robot will
'know that a slip has occurred and will correct for the slip by
'changing its path in order to reach the original destination.
'
'-----[Ownership]---------------------------------------------------
'
'Aaron Stewart
'Department of Mechanical Engineering
'J. B. Speed School of Engineering
'University of Louisville
'
'*******************************************************************************
'*-------------------------[Program]-------------------------------------------*
'*******************************************************************************
'
'-----[EEPROM Data]--------------------------------------------------
CompassOffsets DATA @ 0, (4)                'Stores x and y axis offsets
CompassLowVal  DATA      (1)                'Stores index of lowest angle
CompassCal     DATA      (16)               '16 reference compass angles

'-----[Encoder Variables]-------------------------------------------
REncoValue     VAR Bit                      'Right encoder reading
LEncoValue     VAR Bit                      'Left encoder reading
RCounter       VAR Byte                     'Right Counter, can oly count to 256
LCounter       VAR Byte                     'Left Counter, can only count to 256
PrevR          VAR Bit
PrevL          VAR Bit

'-----[Compass Variables]-------------------------------------------
x              VAR      Word                ' x-axis data
y              VAR      Word                ' y-axis data
status         VAR      Nib                 ' Status flags
angle          VAR      Word                ' Angle measurement
axisOffset     VAR      angle               ' Axis offset

index          VAR      Status              ' EEPROM index
table          VAR      Byte(2)             ' Stores EEPROM table values
span           VAR      x                   ' Span between table entries
angleOffset    VAR      y                   ' Offset btwn measured and table

'-----[Other Variables]---------------------------------------------
TotalDist      VAR      Word                'Begining distance to the final destination.
i              VAR      Byte                'Counter
nomAngle       VAR      Word                'Begining angle
Dist           VAR      Byte                'How far the robot has travled.
arcLength      VAR      Word                'Another distance.
b              CON      6                   'distance from center of robot to center of track.
  2.4 inches, or 5 eps
sigma          VAR      Word
xcord          VAR      Byte
ycord          VAR      Byte
CW             CON      0                   'Indicates Clockwise
CCW            CON      1                   'Indicates Counter Clockwise
dir            VAR      Bit                 'Direction indicator.
prevAngle      VAR      Word
thresh         CON      15                  'Slip Threshold

'-----[Compass Constants]-------------------------------------------
```

```
66 Reset           CON      %0000                ' Reset command for HM55B
67 Measure         CON      %1000                ' Start measurement command
68 Report          CON      %1100                ' Get status/axis values command
69 Ready           CON      %1100                ' 11 -> Done, 00 -> no errors
70 NegMask         CON      %1111100000000000    ' For 11-bit negative to 16-bits
71 current         CON      0                    ' Table array index
72 previous        CON      1                    ' Table array index
73
74
75 '-----[Pin Definitions]-------------------------------------------
76 REnco           CON      0                   'Right encoder pin
77 LEnco           CON      1                   'Left encoder pin
78
79 nInp            PIN      12                  'Define the input pin for ServoPal
80
81 DinDout         PIN      11                  'P11 transceives to/from Din/Dout
82 Clk             PIN      10                  'P10 sends pulses to HM55B's Clk
83 En              PIN       9                  'P9 controls HM55B's /EN(ABLE)
84
85 '-----[Initialization]--------------------------------------------
86 INPUT nInp                                  'Make sure nInp isn't being driven.
87 DO : LOOP UNTIL nInp                        'Wait for ServoPAL to power up.
88
89 LOW nInp                                    'Set pin to an output and hold it low
90 PAUSE 100                                   'for 100mS to reset ServoPAL.
91 HIGH nInp                                   'Raise the pin.
92 PAUSE 100
93
94 RCounter = 0                                'Initialize encoder values
95 LCounter = 0
96
97 '-----[Centering Routine]-------------------------------------------------
98 'DEBUG "Centering...",CR                     'Makes sure the the encoders start
99 PAUSE 500                                   'at the same position everytime.
100 GOSUB CheckEnco
101 IF (REncoValue = 1) THEN
102  PULSOUT nInp, 500
103  DO UNTIL (REncoValue = 0)                 'Pulse right servo
104   GOSUB CheckEnco
105  LOOP
106  PULSOUT nInp, 2000                        'Stop Servos
107 ENDIF
108 IF (LEncoValue = 1) THEN
109   PULSOUT nInp, 4
110   PULSOUT nInp, 1000                       'Pulse left servo
111  DO UNTIL (LEncoValue = 0)
112   GOSUB CheckEnco
113  LOOP
114  PULSOUT nInp, 2000                        'Stop Servos
115  PULSOUT nInp, 2000
116 ENDIF
117 'DEBUG "Centered."
118 PAUSE 1000
119
120 '-----[Main Program]-----------------------------------------------------------------
121 nomAngle = 0                               'Initializes nomAngle
122 FOR i = 0 TO 9
123   GOSUB Compass
124   nomAngle = nomAngle + angle              'Compiles angles so an average can be taken.
125 NEXT
126 nomAngle = nomAngle/10                     'Averages angles and sets the nominal angle.
127 DEBUG DEC ? nomAngle
128
129 DEBUG "How far would you like to go (in eps)? "      'An ep is one "click" on the encoder. One
    ep is 0.4289 inches
130 DEBUGIN DEC TotalDist
```

```
'TotalDist = 100
PAUSE 1000

PULSOUT nInp, 500                           'Pulse both servos to move forward
PULSOUT nInp, 1000                          '(left servo)

DO UNTIL ((angle > nomAngle + thresh) OR (angle < nomAngle - thresh)) 'Allows a small variance
from nominal.
 GOSUB Compass
 GOSUB CheckEnco
 'DEBUG DEC ? RCounter
 IF Rcounter = TotalDist THEN GOTO ending    'If the robot encounters no slip and reaches
destination then quit moving.
LOOP

Dist = Rcounter                             'Distance robot has traveled so far.
DEBUG "Distance to Slip: ", DEC Dist,CR
RCounter = 0                                'Reset encoder values
LCounter = 0

DEBUG "slipping...",CR

slipping:
DO UNTIL (angle = prevAngle)                'Do unitll repeated results.
 GOSUB CheckEnco
 'DEBUG DEC ? RCounter, DEC ? LCounter
 IF (angle <> prevAngle) THEN               'Save old angle as new angle for comparison.
 prevAngle = angle
 ENDIF
 'DEBUG DEC ? prevAngle
 PAUSE 100                                  'Pause for 1/10 of a second to allow for any change
in orientation
 GOSUB Compass                              'Get new angle
LOOP

PULSOUT nInp, 2000                          'Stop Servos
PULSOUT nInp, 2000

FOR i = 0 TO 9
  GOSUB Compass                             'Compiles angles so an average can be taken.
  Sigma = Sigma + angle                     'Here, Sigma is an average angle.
NEXT
angle = sigma/10                            'Averages angles.

'Doing some math for the slip...

IF angle > nomAngle THEN                    'Slipped and rotated CW
 angle = angle - nomAngle
 arcLength = (LCounter)
 dir = CW
ELSEIF angle < nomAngle THEN                'Slipped and rotated CCW
 angle = nomAngle - angle
 arcLength = (RCounter)
 dir = CCW
ENDIF
DEBUG SDEC ? angle, DEC ? arcLength

sigma = arcLength  * 1800 / (angle * 31)    'Here, sigma is the measured radius of the arc.
xcord = sigma*(SIN angle) / 127

arcLength = arcLength + ((thresh * arcLength)/ (angle - thresh))     'Arc modification
DEBUG DEC ? arcLength

sigma = arcLength  * 1800 / (angle * 31)    'Here, sigma is the radius of the arc.
angle = angle * 32/45                       'Convert from degrees to brads.
ycord = sigma - (sigma*(COS angle) / 127)
```

```
xcord = (sigma*(SIN angle) / 127) - xcord    'Distance traveled while slipping prior to breaking threshold.
xcord = (sigma*(SIN angle) / 127) - xcord    'Distance traveled during entire slip.

Totaldist = TotalDist - xcord - Dist         'Convert total distance to travel to distance remmaining.

DEBUG "Radius= ", SDEC sigma,CR, SDEC ? ycord, SDEC ? xcord, "Dist remaining= ", SDEC TotalDist, CR

RCounter = 0                                 'Reset encoder values
LCounter = 0
PULSOUT nInp, 500                            'Move forward
PULSOUT nInp, 1000

'DEBUG "Clearing slip.",CR

DO UNTIL (RCounter = 28)                     'Move forward 12 inches to clear slip.
  GOSUB CheckEnco
LOOP

PULSOUT nInp, 2000                           'Stop Servos
PULSOUT nInp, 2000

DEBUG "Take slip angle measurement.",CR
PAUSE 15000                                  'Pause for 15 seconds to take measurements.

'DEBUG "Slip cleared.",CR
'Doing some math for the clearance...

ycord = ((28 * SIN angle) / 127) + ycord     'In brads
xcord = (28 * COS angle) / 127
DEBUG SDEC ? ycord, SDEC ? xcord

Totaldist = Totaldist - xcord                'Total distance left in the x-dir.
DEBUG SDEC ? TotalDist
sigma = TotalDist ATN ycord                  'Redefine sigma to be the angle (in brads) that robot must rotate from nominal to correct from slip.
'DEBUG "Degree of slip (brad): ", SDEC sigma,CR
sigma = sigma */ 360                         'Convert sigma to degrees.
Totaldist = Totaldist HYP ycord              'Total distance from current position.
DEBUG "Degree of Correction: ", SDEC sigma,CR, SDEC ? TotalDist

'DEBUG "Aligning to original destination.",CR

IF dir = CW THEN
PULSOUT nInp, 700                            'Rotate CCW to recover from CW slip.
PULSOUT nInp, 700
sigma = nomAngle - sigma                     'Redefine sigma to be the complete angle that the robot must rotate to go to destination.
ELSEIF dir = CCW THEN
PULSOUT nInp, 800                            'Rotate CW to recover from CCW slip.
PULSOUT nInp, 800
sigma = nomAngle + sigma                     'Redefine sigma to be the complete angle that the robot must rotate to go to destination.
ENDIF
IF sigma.BIT15 = 1 THEN                      'Allows it to calculate across 360 degree boundary.
sigma = 360 + sigma
DEBUG "Sigma neg.",CR
ENDIF

DEBUG "Angle to Destination: ", SDEC sigma,CR

DO UNTIL ( (sigma + 5) >= angle) AND (angle >= (sigma - 5) )
  GOSUB Compass
  'DEBUG "Compass angle: ", DEC angle,CR
```

```
LOOP

PULSOUT nInp, 2000                          'Stop Servos
PULSOUT nInp, 2000

DEBUG "Take final orientation measurement.",CR
PAUSE 15000                                 'Pause for 15 seconds to take measurements.

PAUSE 100
RCounter = 0                                'Reset encoders
LCounter = 0

DEBUG "Go to original destination.",CR, DEC ? TotalDist

PULSOUT nInp, 500                           'Move forward
PULSOUT nInp, 1000

DO UNTIL (Rcounter = TotalDist)             'Go to original destination.
  GOSUB CheckEnco
LOOP
'DEBUG "Arrived.",CR

ending:
PULSOUT nInp, 2000                          'Stop Servos
PULSOUT nInp, 2000

PAUSE 150

IF dir = CCW THEN
PULSOUT nInp, 700                          'Rotate CCW to begining orientation.
PULSOUT nInp, 700
ELSEIF dir = CW THEN
PULSOUT nInp, 800                          'Rotate CW to begining orientation.
PULSOUT nInp, 800
ENDIF

DO UNTIL (nomAngle + 5 > angle) AND (nomAngle - 5 < angle)
  GOSUB Compass
  'DEBUG "Compass angle: ", DEC angle,CR
LOOP
DEBUG "Robot has arrived.",CR


PULSOUT nInp, 2000                          'Stop Servos
PULSOUT nInp, 2000
END


'-----[Subroutines]-------------------------------------------------------

CheckEnco:                                  'Checks both encoders
REncoValue = INS.LOWBIT(REnco)
IF REncoValue ^ PrevR THEN
RCounter = Rcounter + 1
PrevR = REncoValue
ENDIF
LEncoValue = INS.LOWBIT(LEnco)
IF LEncoValue ^ PrevL THEN
LCounter = LCounter + 1
PrevL = LEncoValue
ENDIF
RETURN

Compass:                                    'Checks Compass
GOSUB Compass_Get_Axes                      ' Get x, and y values
GOSUB Compass_Correct_Offsets               ' Correct axis offsetes
```

```
angle = x ATN -y                            ' Convert x and y to brads
GOSUB Compass_Interpolate                   ' Linear interpolation
angle = angle */ 360                        ' Convert brads to degrees
'DEBUG DEC ? angle
RETURN

' -----[ Subroutine - Compass_Get_Axes ]-------------------------------------

' This subroutine handles BASIC Stamp - HM55B communication and stores the
' magnetic field strength measurements returned by the device in the x and
' y axis variables.

Compass_Get_Axes:                           ' Compass module subroutine

  HIGH En: LOW En                           ' Send reset command to HM55B
  SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]

  HIGH En: LOW En                           ' HM55B start measurement command
  SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
  status = 0                                ' Clear previous status flags

  DO                                        ' Status flag checking loop
    HIGH En: LOW En                         ' Measurement status command
    SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
    SHIFTIN  DinDout,clk,MSBPOST,[Status\4] ' Get Status
  LOOP UNTIL status = Ready                 ' Exit loop when status is ready

  SHIFTIN  DinDout,clk,MSBPOST,[x\11,y\11]  ' Get x & y axis values
  HIGH En                                   ' Disable module

  IF (y.BIT10 = 1) THEN y = y | NegMask     ' Store 11-bits as signed word
  IF (x.BIT10 = 1) THEN x = x | NegMask     ' Repeat for other axis

  RETURN

' -----[ Subroutine - Compass_Correct_Offsets ]-------------------------------

' This subroutine corrects cumulative magnetic field interference that can
' come from sources such as the PCB, jumper wires, a nearby battery, or a
' nearby current source.  This subroutine relies on values stored in
' the EEPROM space that was reserved by the CompassOffsets DATA directive.
' These EEPROM values were written by CalibrateHM55BCompass.bs2.

Compass_Correct_Offsets:

  READ CompassOffsets, Word axisOffset      ' Get x-axis offset
  x = x - axisOffset                        ' Correct x-axis
  READ CompassOffsets + 2, Word axisOffset  ' Get y-axis offset
  y = y - axisOffset                        ' Correct y-axis

  RETURN

' -----[ Subroutine - Compass_Interpolate ]----------------------------------

' This subroutine applies linear interpolation to the refine the compass
' measurement.  This second level of refinement can be performed after the
' Compass_Correct_Offsets subroutine, and it can correct axis skew and other
' errors inherent to the HM55B chip.
'
' The subroutine relies on sixteen actual compass measurements that were stored
' in the sixteen EEPROM locations reserved by the CompassCal DATA directive.
' These measurements were stored by CalibrateHM55BCompass.bs2, and they
' represent the actual compass measurements for 0, 22.5, 45, 90,..., 337.5
' degrees.  The subroutine finds the two EEPROM measurements that the current
' angle measurement falls between.  It then updates the angle measurement
' based on where the angle measurement falls between the two known table values.
```

```
386
387 Compass_Interpolate:
388
389   ' Start with the lowest value in the CompassCal table.
390
391   READ CompassLowVal, index
392
393   ' Load current and previous table values.
394
395   READ CompassCal + index, table(current)
396   READ (CompassCal + (index - 1 & $F)), table(previous)
397
398
399   ' The IF...ELSEIF...ELSE...ENDIF code block finds the two EEPROM CompassCal
400   ' table values that the current angle measurement falls between and calculates
401   ' the difference between the current angle measurement and the lower of the
402   ' two table values.  The IF and ELSEIF blocks deal with values that are
403   ' greater than the highest or less than the lowest table values.  The ELSE
404   ' block everything between the highest and lowest table values.
405
406   IF (angle >= table(previous)) THEN
407     span = (255 - table(previous)) + table(current)
408     angleOffset = angle - table(previous)
409   ELSEIF (angle <= table(current)) THEN
410     span = table(current) + (255 - table(previous))
411     angleOffset = angle + (255 - table(previous))
412   ELSE
413     index = index - 1
414     READ CompassCal + index, table(current)
415     DO
416       table(previous) = table(current)
417       index = index + 1
418       READ CompassCal + index, table(current)
419       IF (angle <= table(current)) AND (angle > table(previous)) THEN
420         span = table(current) - table(previous)
421         angleOffset = angle - table(previous)
422         EXIT
423       ENDIF
424     LOOP
425   ENDIF
426
427   ' After the offset between the current angle measurement and the next lower
428   ' table measurement has been determined, this code block uses it along with
429   ' the span between the table entries above and below the angle measurement
430   ' to solve for: angle(corrected) = angle(offset) * 16 / span.
431   ' This code block also rounds up or down by comparing the remainder of
432   ' the angleOffset / span division to the value of (span / 2).
433
434   angleOffset = angleOffset * 16
435   angle = (angleOffset / span) + ((angleOffset // span) / (span / 2))
436   angle = ((index - 1 & $F) * 16) + angle
437   angle = angle & $ff
438
439   RETURN
```

```
' Slip_Detection_rev7_Line Slip Method_Final.bs2
' {$STAMP BS2}
' {$PBASIC 2.5}

'-----[Description]-------------------------------------------------
'
'This algorithim is designed to detect slip occurrences by sensing
'the orientation of the tracked vehicle with the use of a compass.
'If the compass senses a change in orientation, the robot will
'know that a slip has occurred and will correct for the slip by
'changing its path in order to reach the original destination.
'
'-----[Ownership]---------------------------------------------------
'
'Aaron Stewart
'Department of Mechanical Engineering
'J. B. Speed School of Engineering
'University of Louisville
'
'*****************************************************************************
'*-------------------------[Program]-------------------------------------------*
'*****************************************************************************
'
'-----[EEPROM Data]-------------------------------------------------
CompassOffsets DATA @ 0, (4)                'Stores x and y axis offsets
CompassLowVal  DATA      (1)                'Stores index of lowest angle
CompassCal     DATA      (16)               '16 reference compass angles

'-----[Encoder Variables]-------------------------------------------
REncoValue     VAR Bit                      'Right encoder reading
LEncoValue     VAR Bit                      'Left encoder reading
RCounter       VAR Byte                     'Right Counter, can oly count to 256
LCounter       VAR Byte                     'Left Counter, can only count to 256
PrevR          VAR Bit
PrevL          VAR Bit

'-----[Compass Variables]-------------------------------------------
x              VAR      Word                ' x-axis data
y              VAR      Word                ' y-axis data
status         VAR      Nib                 ' Status flags
angle          VAR      Word                ' Angle measurement
axisOffset     VAR      angle               ' Axis offset

index          VAR      Status              ' EEPROM index
table          VAR      Byte(2)             ' Stores EEPROM table values
span           VAR      x                   ' Span between table entries
angleOffset    VAR      y                   ' Offset btwn measured and table

'-----[Other Variables]---------------------------------------------
TotalDist      VAR      Word                'Begining distance to the final destination.
i              VAR      Byte                'Counter
nomAngle       VAR      Word                'Begining angle
Dist           VAR      Byte                'How far the robot has travled.
arcLength      VAR      Word                'Another distance.
b              CON      6                   'distance from center of robot to center of track.
  2.4 inches, or 5 eps
sigma          VAR      Word
xcord          VAR      Byte
ycord          VAR      Byte
CW             CON      0                   'Indicates Clockwise
CCW            CON      1                   'Indicates Counter Clockwise
dir            VAR      Bit                 'Direction indicator.
prevAngle      VAR      Word
thresh         CON      15                  'Slip Threshold

'-----[Compass Constants]-------------------------------------------
```

```
66 Reset           CON      %0000                ' Reset command for HM55B
67 Measure         CON      %1000                ' Start measurement command
68 Report          CON      %1100                ' Get status/axis values command
69 Ready           CON      %1100                ' 11 -> Done, 00 -> no errors
70 NegMask         CON      %1111100000000000    ' For 11-bit negative to 16-bits
71 current         CON      0                    ' Table array index
72 previous        CON      1                    ' Table array index
73
74
75 '-----[Pin Definitions]-------------------------------------------
76 REnco           CON      0                   'Right encoder pin
77 LEnco           CON      1                   'Left encoder pin
78
79 nInp            PIN      12                  'Define the input pin for ServoPal
80
81 DinDout         PIN      11                  'P11 transceives to/from Din/Dout
82 Clk             PIN      10                  'P10 sends pulses to HM55B's Clk
83 En              PIN       9                  'P9 controls HM55B's /EN(ABLE)
84
85 '-----[Initialization]--------------------------------------------
86 INPUT nInp                                  'Make sure nInp isn't being driven.
87 DO : LOOP UNTIL nInp                        'Wait for ServoPAL to power up.
88
89 LOW nInp                                    'Set pin to an output and hold it low
90 PAUSE 100                                   'for 100mS to reset ServoPAL.
91 HIGH nInp                                   'Raise the pin.
92 PAUSE 100
93
94 RCounter = 0                                'Initialize encoder values
95 LCounter = 0
96
97 '-----[Centering Routine]-------------------------------------------------
98 'DEBUG "Centering...",CR                     'Makes sure the the encoders start
99 PAUSE 500                                   'at the same position everytime.
100 GOSUB CheckEnco
101 IF (REncoValue = 1) THEN
102  PULSOUT nInp, 500
103  DO UNTIL (REncoValue = 0)                 'Pulse right servo
104   GOSUB CheckEnco
105  LOOP
106  PULSOUT nInp, 2000                        'Stop Servos
107 ENDIF
108 IF (LEncoValue = 1) THEN
109   PULSOUT nInp, 4
110   PULSOUT nInp, 1000                       'Pulse left servo
111  DO UNTIL (LEncoValue = 0)
112   GOSUB CheckEnco
113  LOOP
114  PULSOUT nInp, 2000                        'Stop Servos
115  PULSOUT nInp, 2000
116 ENDIF
117 'DEBUG "Centered."
118 PAUSE 1000
119
120 '-----[Main Program]-----------------------------------------------------------------
121 nomAngle = 0                               'Initializes nomAngle
122 FOR i = 0 TO 9
123   GOSUB Compass
124   nomAngle = nomAngle + angle              'Compiles angles so an average can be taken.
125 NEXT
126 nomAngle = nomAngle/10                     'Averages angles and sets the nominal angle.
127 DEBUG DEC ? nomAngle
128
129 DEBUG "How far would you like to go (in eps)? "      'An ep is one "click" on the encoder. One
    ep is 0.4289 inches
130 DEBUGIN DEC TotalDist
```

```
'TotalDist = 100
PAUSE 1000

PULSOUT nInp, 500                           'Pulse both servos to move forward
PULSOUT nInp, 1000                          '(left servo)

DO UNTIL ((angle > nomAngle + thresh) OR (angle < nomAngle - thresh)) 'Allows a small variance
from nominal.
 GOSUB Compass
 GOSUB CheckEnco
 'DEBUG DEC ? RCounter
 IF Rcounter = TotalDist THEN GOTO ending    'If the robot encounters no slip and reaches
destination then quit moving.
LOOP

Dist = Rcounter                             'Distance robot has traveled so far.
DEBUG "Distance to Slip: ", DEC Dist,CR
RCounter = 0                                'Reset encoder values
LCounter = 0

'DEBUG "slipping...",CR

slipping:
DO UNTIL (angle = prevAngle)                'Do unitll repeated results.
 GOSUB CheckEnco
 'DEBUG DEC ? RCounter, DEC ? LCounter
 IF (angle <> prevAngle) THEN               'Save old angle as new angle for comparison.
 prevAngle = angle
 ENDIF
 'DEBUG DEC ? prevAngle
 PAUSE 100                                  'Pause for 1/10 of a second to allow for any change
in orientation
 GOSUB Compass                              'Get new angle
LOOP

PULSOUT nInp, 2000                          'Stop Servos
PULSOUT nInp, 2000

FOR i = 0 TO 9
  GOSUB Compass                             'Compiles angles so an average can be taken.
  Sigma = Sigma + angle                     'Here, Sigma is an average angle.
NEXT
angle = sigma/10                            'Averages angles.

'Doing some math for the slip...

IF angle > nomAngle THEN                    'Slipped and rotated CW
 angle = angle - nomAngle
 arcLength = (LCounter)
 dir = CW
ELSEIF angle < nomAngle THEN                'Slipped and rotated CCW
 angle = nomAngle - angle
 arcLength = (RCounter)
 dir = CCW
ENDIF
DEBUG SDEC ? angle
arcLength = 0                               'No arc length measurement
Dist = Dist + RCounter                      'Treat only as displacement in x direction

DEBUG DEC ? arcLength

sigma = arcLength  * 1800 / (angle * 31)    'Here, sigma is the radius of the arc.
angle = angle * 32/45                       'Convert from degrees to brads.
ycord = sigma - (sigma*(COS angle) / 127)
xcord = sigma*(SIN angle) / 127
xcord = 4
```

```
194
195 Totaldist = TotalDist + xcord - Dist          'Convert total distance to travel to distance
    remmaining.
196
197 DEBUG "Radius= ", SDEC sigma,CR, SDEC ? ycord, SDEC ? xcord, "Dist remaining= ", SDEC TotalDist,
    CR
198
199 RCounter = 0                                  'Reset encoder values
200 LCounter = 0
201 PULSOUT nInp, 500                             'Move forward
202 PULSOUT nInp, 1000
203
204 'DEBUG "Clearing slip.",CR
205
206 DO UNTIL (RCounter = 28)                      'Move forward 12 inches to clear slip.
207   GOSUB CheckEnco
208 LOOP
209
210 PULSOUT nInp, 2000                            'Stop Servos
211 PULSOUT nInp, 2000
212
213 DEBUG "Take slip angle measurement.",CR
214 PAUSE 15000                                   'Pause for 15 seconds to take measurements.
215
216
217 'DEBUG "Slip cleared.",CR
218 'Doing some math for the clearance...
219
220 ycord = ((28 * SIN angle) / 127) + ycord     'In brads
221 xcord = (28 * COS angle) / 127
222 DEBUG SDEC ? ycord, SDEC ? xcord
223
224 Totaldist = Totaldist - xcord                 'Total distance left in the x-dir.
225 DEBUG SDEC ? TotalDist
226 sigma = TotalDist ATN ycord                   'Redefine sigma to be the angle (in brads) that
    robot must rotate from nominal to correct from slip.
227 'DEBUG "Degree of slip (brad): ", SDEC sigma,CR
228 sigma = sigma */ 360                          'Convert sigma to degrees.
229 Totaldist = Totaldist HYP ycord               'Total distance from current position.
230 DEBUG "Degree of Correction: ", SDEC sigma,CR, SDEC ? TotalDist
231
232 'DEBUG "Aligning to original destination.",CR
233
234 IF dir = CW THEN
235 PULSOUT nInp, 700                             'Rotate CCW to recover from CW slip.
236 PULSOUT nInp, 700
237 sigma = nomAngle - sigma                      'Redefine sigma to be the complete angle that the
    robot must rotate to go to destination.
238 ELSEIF dir = CCW THEN
239 PULSOUT nInp, 800                             'Rotate CW to recover from CCW slip.
240 PULSOUT nInp, 800
241 sigma = nomAngle + sigma                      'Redefine sigma to be the complete angle that the
    robot must rotate to go to destination.
242 ENDIF
243 IF sigma.BIT15 = 1 THEN                       'Allows it to calculate across 360 degree boundary.
244 sigma = 360 + sigma
245 DEBUG "Sigma neg.",CR
246 ENDIF
247
248 DEBUG "Angle to Destination: ", SDEC sigma,CR
249
250 DO UNTIL ( (sigma + 3) >= angle) AND (angle >= (sigma - 3) )
251   GOSUB Compass
252   'DEBUG "Compass angle: ", DEC angle,CR
253 LOOP
254
```

```
255 PULSOUT nInp, 2000                          'Stop Servos
256 PULSOUT nInp, 2000
257
258 DEBUG "Take final orientation measurement.",CR
259 PAUSE 15000                                 'Pause for 15 seconds to take measurements.
260
261 PAUSE 100
262 RCounter = 0                                'Reset encoders
263 LCounter = 0
264
265 DEBUG "Go to original destination.",CR, DEC ? TotalDist
266
267 PULSOUT nInp, 500                           'Move forward
268 PULSOUT nInp, 1000
269
270 DO UNTIL (Rcounter = TotalDist)             'Go to original destination.
271   GOSUB CheckEnco
272 LOOP
273 'DEBUG "Arrived.",CR
274
275 ending:
276 PULSOUT nInp, 2000                          'Stop Servos
277 PULSOUT nInp, 2000
278
279 PAUSE 150
280
281 IF dir = CCW THEN
282 PULSOUT nInp, 700                          'Rotate CCW to begining orientation.
283 PULSOUT nInp, 700
284 ELSEIF dir = CW THEN
285 PULSOUT nInp, 800                          'Rotate CW to begining orientation.
286 PULSOUT nInp, 800
287 ENDIF
288
289 DO UNTIL (nomAngle + 3 > angle) AND (nomAngle - 3 < angle)
290   GOSUB Compass
291   'DEBUG "Compass angle: ", DEC angle,CR
292 LOOP
293 DEBUG "Robot has arrived.",CR
294
295
296 PULSOUT nInp, 2000                          'Stop Servos
297 PULSOUT nInp, 2000
298 END
299
300
301 '-----[Subroutines]-------------------------------------------------------
302
303 CheckEnco:                                  'Checks both encoders
304 REncoValue = INS.LOWBIT(REnco)
305 IF REncoValue ^ PrevR THEN
306 RCounter = Rcounter + 1
307 PrevR = REncoValue
308 ENDIF
309 LEncoValue = INS.LOWBIT(LEnco)
310 IF LEncoValue ^ PrevL THEN
311 LCounter = LCounter + 1
312 PrevL = LEncoValue
313 ENDIF
314 RETURN
315
316 Compass:                                   'Checks Compass
317 GOSUB Compass_Get_Axes                     ' Get x, and y values
318 GOSUB Compass_Correct_Offsets              ' Correct axis offsetes
319 angle = x ATN -y                           ' Convert x and y to brads
320 GOSUB Compass_Interpolate                  ' Linear interpolation
```

```
angle = angle */ 360                          ' Convert brads to degrees
'DEBUG DEC ? angle
RETURN

' -----[ Subroutine - Compass_Get_Axes ]-------------------------------------

' This subroutine handles BASIC Stamp - HM55B communication and stores the
' magnetic field strength measurements returned by the device in the x and
' y axis variables.

Compass_Get_Axes:                             ' Compass module subroutine

  HIGH En: LOW En                             ' Send reset command to HM55B
  SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]

  HIGH En: LOW En                             ' HM55B start measurement command
  SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
  status = 0                                  ' Clear previous status flags

  DO                                          ' Status flag checking loop
    HIGH En: LOW En                           ' Measurement status command
    SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
    SHIFTIN  DinDout,clk,MSBPOST,[Status\4]   ' Get Status
  LOOP UNTIL status = Ready                   ' Exit loop when status is ready

  SHIFTIN  DinDout,clk,MSBPOST,[x\11,y\11]    ' Get x & y axis values
  HIGH En                                     ' Disable module

  IF (y.BIT10 = 1) THEN y = y | NegMask       ' Store 11-bits as signed word
  IF (x.BIT10 = 1) THEN x = x | NegMask       ' Repeat for other axis

  RETURN

' -----[ Subroutine - Compass_Correct_Offsets ]-------------------------------

' This subroutine corrects cumulative magnetic field interference that can
' come from sources such as the PCB, jumper wires, a nearby battery, or a
' nearby current source.  This subroutine relies on values stored in
' the EEPROM space that was reserved by the CompassOffsets DATA directive.
' These EEPROM values were written by CalibrateHM55BCompass.bs2.

Compass_Correct_Offsets:

  READ CompassOffsets, Word axisOffset        ' Get x-axis offset
  x = x - axisOffset                          ' Correct x-axis
  READ CompassOffsets + 2, Word axisOffset    ' Get y-axis offset
  y = y - axisOffset                          ' Correct y-axis

  RETURN

' -----[ Subroutine - Compass_Interpolate ]-----------------------------------

' This subroutine applies linear interpolation to the refine the compass
' measurement.  This second level of refinement can be performed after the
' Compass_Correct_Offsets subroutine, and it can correct axis skew and other
' errors inherent to the HM55B chip.
'
' The subroutine relies on sixteen actual compass measurements that were stored
' in the sixteen EEPROM locations reserved by the CompassCal DATA directive.
' These measurements were stored by CalibrateHM55BCompass.bs2, and they
' represent the actual compass measurements for 0, 22.5, 45, 90,..., 337.5
' degrees.  The subroutine finds the two EEPROM measurements that the current
' angle measurement falls between.  It then updates the angle measurement
' based on where the angle measurement falls between the two known table values.

Compass_Interpolate:
```

```
387
388   ' Start with the lowest value in the CompassCal table.
389
390   READ CompassLowVal, index
391
392   ' Load current and previous table values.
393
394   READ CompassCal + index, table(current)
395   READ (CompassCal + (index - 1 & $F)), table(previous)
396
397
398   ' The IF...ELSEIF...ELSE...ENDIF code block finds the two EEPROM CompassCal
399   ' table values that the current angle measurement falls between and calculates
400   ' the difference between the current angle measurement and the lower of the
401   ' two table values.  The IF and ELSEIF blocks deal with values that are
402   ' greater than the highest or less than the lowest table values.  The ELSE
403   ' block everything between the highest AND lowest table values.
404
405   IF (angle >= table(previous)) THEN
406     span = (255 - table(previous)) + table(current)
407     angleOffset = angle - table(previous)
408   ELSEIF (angle <= table(current)) THEN
409     span = table(current) + (255 - table(previous))
410     angleOffset = angle + (255 - table(previous))
411   ELSE
412     index = index - 1
413     READ CompassCal + index, table(current)
414     DO
415       table(previous) = table(current)
416       index = index + 1
417       READ CompassCal + index, table(current)
418       IF (angle <= table(current)) AND (angle > table(previous)) THEN
419         span = table(current) - table(previous)
420         angleOffset = angle - table(previous)
421         EXIT
422       ENDIF
423     LOOP
424   ENDIF
425
426   ' After the offset between the current angle measurement and the next lower
427   ' table measurement has been determined, this code block uses it along with
428   ' the span between the table entries above and below the angle measurement
429   ' to solve for: angle(corrected) = angle(offset) * 16 / span.
430   ' This code block also rounds up or down by comparing the remainder of
431   ' the angleOffset / span division to the value of (span / 2).
432
433   angleOffset = angleOffset * 16
434   angle = (angleOffset / span) + ((angleOffset // span) / (span / 2))
435   angle = ((index - 1 & $F) * 16) + angle
436   angle = angle & $ff
437
438   RETURN
```

VITA

Aaron Stewart graduated with his Bachelor's Degree in Mechanical Engineering from the University of Louisville in the fall of 2008 and received his Masters Degree in Mechanical Engineering in the spring of 2010, also from the University of Louisville. While at the university Aaron was very active in various intramural sports including soccer, flag football, kickball, dodge ball and volleyball and was an intramural champion in both soccer and volleyball. While at the University of Louisville, Aaron worked as a Guest Service Supervisor at the Louisville Marriott Downtown and as the Senior Wedding Videographer for Complete Video where he and his team received a Best of Weddings award for the city of Louisville. Upon graduation Aaron will begin work for SABIC Innovative Plastics as an engineer in the Operational Development Program.