2016

Multithreaded Sensor Analysis for a Redundant Dynamic Positioning 2 Project

Stephen A. Palmieri

Adam D. Kline

Tooran Emami

Ali Reza

Peter F. Swaszek
University of Rhode Island, swaszek@uri.edu

Citation/Publisher Attribution

Palmieri, S. A., Kline, A. D., Emami, T., Reza, A., & Swaszek, P. (2016). Multithreaded Sensor Analysis for a Redundant Dynamic Positioning 2 Project. *Proceedings of the ASEE NE 2016 Conference*. Kingston, RI: American Society for Engineering Education.
Available at: http://egr.uri.edu/wp-uploads/asee2016/47-526-1-PB.pdf

Multithreaded Sensor Analysis for a Redundant Dynamic Positioning 2 Project

Multithreaded Sensor Analysis for a Redundant Dynamic Positioning 2 Project

Stephen A. Palmieri[1], Adam D. Kline[1], Tooran Emami[1], Ali Reza[1], and Peter Swaszek[2]

[1]United States Coast Guard Academy, [2]University of Rhode Island

Abstract- The goal of the electrical engineering capstone project, Dynamic Positioning 2 (DP2), is to prototype a controlled dynamic positioning system that has a single-failure safe capability. The primary objective is for the platform to maintain a desired heading and position within ten degrees and three hundred millimeters, respectively, using data obtained from three light detection and ranging (LIDAR) sensors. The secondary objective is for the control system to compensate for the failure of a single sensor or motor. The platform for the DP2 project is a salvage drum that encases the electronic equipment and an inner tube for buoyancy. The internal construction consists of three tiers containing batteries at the lowest level, an onboard computer at the second level, and control hardware at the top level. The platform is tested in an indoor tank with an area of sixteen meters squared. The vessel's position is calculated from the LIDAR data (bearing and range) to eight stationary poles that mark the outside of the tank using an overdetermined least squares matrix solution. The heading is calculated using the bearings and ranges to specific pairs of poles. An ad-hoc wireless network is used to communicate with the onboard computer while it is operating. All programming was completed in the *NET Framework* and *MATLAB®*. Students complete the project milestones through the application of material from past courses in computer control systems, software engineering, and electronic navigation at the U.S. Coast Guard Academy. The DP2 project is sponsored by the Marine Safety Center (MSC).

I. INTRODUCTION

From operating a buoy tender to regulating the plans for an offshore oil rig, the United States Coast Guard has a vested interest in understanding the operation of dynamic positioning (DP) systems. During the course of the dynamic positioning 2 capstone project, students have expanded their knowledge of DP systems by developing and integrating the different components necessary to operate one. In addition to technical knowledge gained during the project, students also were exposed to a variety of engineering project management ideas. The duration, depth, and breadth of the dynamic positioning 2 (DP2) project has provided students with a great opportunity to prepare for future careers in an engineering field.

The DP2 project is a senior capstone design project in the Electrical Engineering major at the United States Coast Guard Academy. All work for the DP2 project was carried out in the fall semester of 2015 and the spring semester of 2016. The Coast Guard's Marine Safety Center (MSC) sponsors the project, due to its responsibility for developing regulations for the dynamic positioning industry.

The dynamic positioning 2 project is a continuation of the previous year's senior design project [1-3]. The hardware platform from 2014-2015 was mostly retained due to its well thought out design and configuration. The focus of the DP2 project was overhauling the methods and programming framework used to calculate position and control the platform. The previous year's project used only a single LIDAR sensor and two navigation aids two calculate position. This placed severe constraints on the system. This year three LIDAR sensors were used with eight navigation aids, which allows the vessel to operate at any heading and position. Also, another major constraint encountered by the previous year's project was a high sampling time, to rectify this all programming was done in C# in the .NET framework allowing for faster processing and serial port communications.

II. BACKGROUND

Both the usage of and applications for dynamic positioning system are continuing to expand, and as DP systems enter environments that require increasing precision and accuracy, the need to understand and effectively regulate them becomes incredibly pertinent. Organizations such as the American Bureau of Shipping (ABS) and the Nautical Institute have begun reviewing and certifying DP systems and operators respectively, in order to prevent DP failures such as drive-offs, blow-outs, and loss of station keeping. The risk failures are mitigated by including redundant systems as fail safes, such as multiple sources for positioning both relative and absolute, and multiple mechanical systems for ensuring the systems always have propulsion [4].

A. Dynamic Positioning Systems

In its simplest form, a dynamic positioning system consist of two components: A position calculation method, and a controller. These two components work together in order to allow the DP system to maintain position and heading without any user input. First the user inputs a desired position and heading, which is then compared to the calculated position to produce and error. The error is then fed into the controller which outputs propulsion commands in order to move the platform from the calculated position to the desired position. The effectiveness of the system is dependent both on the precision of the position calculation method and the quality of the controller. Effective DP systems will be able to maintain position even with the failure of various sensors used for position calculation and thrusters used by the controller.

B. Light Detection and Ranging

Light Detection and Ranging (LIDAR) sensors operate in a similar manner as RADARs, except instead of reflecting radio waves off objects they reflect light. By timing how long it takes the light to return the LIDAR sensor can calculate the distance to objects at various angles. The sensors used for the DP2 project are three Hokuyo URG-04LX-UG01 LIDAR sensors, which interface through a USB 2.0 port. These sensors return 682 ranges over a 240 degree span, with a max range of approximately 4 meters.

C. H-Bridge and Pulse Width Modulation

H-Bridges are used to convert pulse width modulation (PWM) signals into DC voltages to power the thrusters. A pulse width modulation signal is a digitally encoded analog signal that stores its value in the duty cycle of a square wave. The H-Bridge receives one PWM signal and a directional bit, as the duty cycle of the PWM signal increases the output voltage of the H-Bridge increases, and the directional bit switches the H-Bridge output from a positive to negative DC voltage. Since the H-Bridge is fed by a 12V battery, it will return +12V for a 100% duty cycle and a direction bit of 1 and -6V for a 50% duty cycle and a direction bit of 0.

D. Control Theory

Control theory is implemented in many automated processes. It consists of automatically controlling a system, monitoring the output, and the comparison of a desired set point to its output. There are both open-loop and closed-loop systems. An open-loop system consists of no feedback and human intervention in order to reach a desired set point, whereas a closed-loop system will dynamically adjust the plant's input in order to reach its desired output. Closed-loop systems prove much more advantageous because they will automatically control a system. These systems must be accurately and can be mathematically described using differential equations. Modeling a system mathematically allows for an accurate and appropriate design of a linear controller. A common type of linear controller is a proportional-integral-derivative controller (PID). PID controllers allow for a system to adjust by putting emphasis on different requirements, such as a quick response, reaching the desired outcome, and reducing overshoot. Each part of the PID controller is used for a specific problem with a system. The proportional part allows for a quick response of the system, the integral part ensures that the desired set point is reached, and lastly, the derivative part minimizes overshoot. PID controllers are a common linear controller and were used in implementation of a controller.

III. Methodology

The methodology section will describe how the different components onboard the dynamic positioning platform interact with each other in order to maintain a desired position and heading. The effective operation of the dynamic position platform requires that the onboard hardware and software work effectively independently as well as interface both quickly and accurately. The overall goals of the project had to be incorporated into the design and testing of each system.

A. Platform Construction

The platform is mechanically comprised of a steel salvage drum with 6 welded thrusters on the keel of the hull. In order to control this vessel, three LIDAR sensors are attached above in a vertical stack physically offset by 120 degrees. These sensors feed information into the system which is located within the steel drum. Four of the DC brushed thrusters are used for translational movement, which is in the x-y plane relative to the surface of the water. In order to control rotational movement, two thrusters are placed tangentially to the steel drum, facing opposite directions. This design allows for complete, redundant control of the vessel in the tank even if single-failures were to occur.

Figure 1. Dynamic positioning platform, in test tank, with 3 LIDAR sensors offset by 120°

A deeper understanding of the platform can be learned from observing the components within the platform. The platform consists of three internal tiers. On the first and bottom tier lies the power for the vessel. Two Optima Dual-Marine Purpose 12 VDC lead-acid six cell batteries are used in order to power both control equipment and the thrusters of the vessel. In order to ensure that all electrical equipment onboard is protected, the batteries are fed into two fuseboxes, which are located on the top tier. On the second tier resides the onboard computer, which is used as the controller for the system. This computer is running all the software that controls input from the sensors and the output to the thrusters. The computer also has external equipment connected, such as the WiFi-adapter, three sensors, and a microcontroller. On the upper tier of the platform resides the control interface equipment. This includes the fuse boxes for both electronic equipment and each thruster, the H-bridges that are used to control direction and strength of each thruster, and lastly the microcontroller, specifically an Arduino, that is used to send the control commands to the H-bridges. Lastly, auxiliary equipment includes four DC fans that cool the second and third tiers of the platform.

B. LIDAR Interface

The first step in operating the dynamic positioning platform is calculating an accurate position. For the DP2 project LIDAR sensors are used as the means of position calculation, making the first step of position calculation the interface between the LIDAR sensors and onboard computer using serial port communications.

For position calculation three Hokuyo URG-04LX-UG01 Light Detection and Ranging (LIDAR) Sensors were used in order to observe the surroundings of the system through a USB 2.0 connection with Windows 7 OS. These specific sensors send infrared light pulses in a scan area of 240 degrees and can detect objects within 4 meters of distance, though they are most accurate from 20-2000 millimeters. A total of 682 ranges are obtained from the sensor at a frequency of 10 Hz. Hokuyo has created a specific protocol, SCIP 2.0, in order to properly communicate with the sensors through software. The design began with students reverse engineering a properly working Matlab script used for obtaining the ranges and then improved on it using the SCIP 2.0 protocol.

The process for communicating with the sensor using C# was determined by the SCIP 2.0 protocol. First, students have to create an object of the SerialPort Class in C# in order to communicate, while also making sure default specifications, such as baudrate and port name are correct. After the sensor has been initialized in the executable, a command is written to turn on the laser and then get the current scan. When the sensor responds, the data is read in. After reading in the data in ASCII text it must be converted into an array with 682 ranges in order to be useful. To convert this text, irrelevant data must first be erased, such as the header and status of the sensor. Once this data is erased, the text is parsed into three columns and decoded according to the SCIP 2.0 protocol encoding scheme. After communication with one sensor was successful, the next step was to simultaneously communicate with 2 additional sensors in order to meet the redundancy requirements of the project. A method, named Threading, was used to implement this requirement specification. Threading allows the software to run parallel processes so that each sensor is queried at the same time and responds at the same time. As a consequence the dynamic position system can operate about two to three times as fast when threading. Allowing a processed scans to be returned at a rate of approximately 5Hz.

C. Interpreting LIDAR Output

The final output of the LIDAR sensors after a single scan is three integer arrays with 682 values. Each integer array contains the range values in millimeters observed by the LIDAR sensor at each step. Since the sensor scans 240 degrees in 682 steps, each step is 0.3519 degrees. This raw range data from the LIDAR sensors must be processed into useful information that is capable of determining the position of the platform in a known environment.

To use the LIDAR scans effectively the tank testing environment must be configured to provide the sensors with useful information. To aid position calculation a Cartesian coordinate system was developed with the center of the tank being (0,0), the x-axis extending +/- 1900 millimeters left and right, and the y-axis extending 950 millimeters forwards and backwards. To calculate

position eight vertical polls (navigation aids) were placed outside the tank in known positions. Using ranges to distinct navigation aids in known positions allows the platform to calculate an accurate position within the tank environment.

However, converting 682 ranges into distinct ranges to specific navigation aids requires extensive processing. The first challenge is to differentiate between different navigation aids. There are two options for implementing this, the first would be to make navigation aids different sizes. In order to calculate the size of the aid the first and last sample would have to be used to calculate the total width of the object, which then must be compared to the widths of the other aids. The problem with this method is that its accuracy is dependent on accurately observing the leading and trailing edge of the navigation aid. This accuracy decreases as the distance from the aid increases, which results in greater distances between steps. The second method of differentiating navigation aids is using eight identical aids, but positioning them asymmetrically so that in relation to each other they appear distinct. The advantage of this method is that it only needs to see the center of the aid, which is more accurate at greater distances. The final navigation aid configuration used for the test tank environment was eight navigation aids, placed in four pairs, with each pair being a set distance apart. Since the LIDAR sensor scans counter clockwise, it will always see the second aid in the pair before the first. Allowing all navigation aids to be identified as long as at least three are seen (which should occur in the majority of cases).

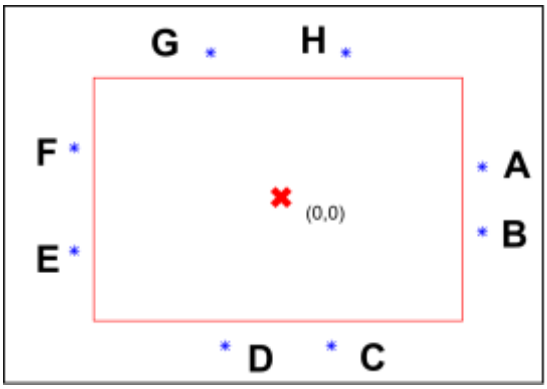

Figure 2. Testing environment coordinate system. Navigation aids are uniquely spaced for easy identification (AB = 600mm, CD = 1120mm, EF = 860mm, GH = 1380mm)


Figure 3. Actual tank configuration, utilizing 8 navigation aids for position calculation.

The key to the effective implementation of the eight navigation aids is ensuring that only the navigation aids are seen. Any other ranges observed by the sensors would invalidate the calculation process. Since the range of the LIDAR sensors is much greater than the distance to the perimeter of the tank the surrounding area must either be cleared of any objects that may be observed or the sensor data must process out objects outside the immediate area of the tank. Since the tank environment does not allow the removal of all possible interfering objects the 682 ranges must be masked to exclude any ranges that are greater than the possible distances to the navigation aids. However, since the platform is moving within the tank, the maximum allowable range must be updated depending on the position and heading of the platform. The dynamic mask requires an initial input position, and then uses the last calculated position to mask the ranges effectively for each scan.

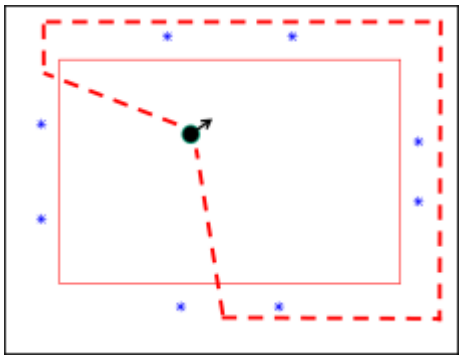
Figure 4. Dynamic mask for single sensor positioned in the tank. Mask updates for each scan based on input position

With a dynamic mask and a unique navigation aid configuration the dynamic positioning platform can identify unique navigation aids in known positions. Allowing the platform to convert an array of 682 ranges to ranges and bearings to navigation aids in known positions

D. Position and Heading Calculation

When calculating position in the test tank, the dynamic positioning platform is solving for two unknowns, its x-position and its y-position. With all three LIDAR sensors working the dynamic positioning platform should have 720 degrees of coverage allowing it to see each of the eight navigation aids twice, resulting in a total of sixteen observed navigation aids. With only two or one sensor operating the resulting number of navigation aids observed is variable depending on the location in the tank, but in the majority of cases even a single sensor should see at least four navigation aids. When using ranges to calculate position, two distinct ranges are necessary to calculate position. So in all standard operating cases with two sensors functioning, there will be more ranges acquired than needed to calculate position.

The fact that more information is available than necessary to solve for the equation makes the system overdetermined. Also since the LIDAR sensors output is real data the probability of all the ranges converging exactly on a single point without any offset is for all practical purposes equal to zero. GPS receivers face the same issue when solving for position with information from multiple satellites. To get the most accurate position the GPS receivers use an overdetermined least squares solution so solve for x-position, y-position, z-position, and time offset. A least squares solution finds the position with the minimum offset from the overdetermined set of data, resulting in the most probable position being calculated. Since the dynamic positioning platform is only required to solve for x-position and y-position the platform uses a simplified overdetermined matrix solution. This solution utilizes all the information available to calculate the best position solution for the platform.

The matrices used in the calculation are shown below in equations 1-5. First the user assumes a position, and then equation 5 is used to calculate the difference between the assumed position and actual position. The difference is then added to the assumed position to create a new assumed position, this new assumed position is then used in equation 5 again, generating a new difference. This process is repeated 5 times at which point, the difference will collapse to 0 resulting in the assumed position being equal to the actual position of the platform.

$$A = Hx$$

Eq 1. Linear solution to matrix equation

$$H = \begin{bmatrix} \frac{X_{assumed} - X_{aid1}}{R_{aid1}} & \frac{Y_{assumed} - Y_{aid1}}{R_{aid1}} \\ \vdots & \vdots \\ \frac{X_{assumed} - X_{aidn}}{R_{aidn}} & \frac{Y_{assumed} - Y_{aidn}}{R_{aidn}} \end{bmatrix}$$

Eq 2. H matrix for position calculation, represents the values of cosine and sine of the angles to the navigation aid.

$$A = \begin{bmatrix} R_{assumed} - R_{aid1} \\ \vdots \\ R_{assumed} - R_{aidn} \end{bmatrix}$$

Eq 3. A matrix for position calculation, represents the difference between the assumed ranges and the measured ranges.

$$x = \begin{bmatrix} \Delta X \\ \Delta Y \end{bmatrix}$$

Eq 4. x matrix for position calculation, represents the distance between assumed position and actual position.

$$x = (H^T H)^{-1} H^T A$$

Eq 5. Moore-Penrose pseudoinverse, solves for x for the matrix system.

After calculating the position of the platform the heading must also be calculated. To calculate the ranges and angles to navigation aid pairs are used. In the test tank environment a heading of zero degrees parallels the positive y-axis. Whenever a sensor see both navigation aids one of the four navigation aid pairs heading is calculated. The heading calculation method functions by comparing the ranges and angles to both observed aids in a pair and then using trigonometry to solve for specific angles. The equations used vary based on which aid is closer to the platform, resulting in two equations for each pair of navigation aids and eight total. The calculated headings are then averaged to output a final heading.

Referencing figure 5, the angle θ_h can be calculated by first determine the length of the perpendicular bisector, L, using equation 6 below. Then the angle θ_z can be calculated using equation 7. Once θ_z is known, the angle θ_A and θ_z's compliment can be subtracted from 180° as seen in equation 8 to determine the heading of the platform.

$$L = r_B \cos(\theta_{B-A})$$

Eq 6. Length of the perpendicular bisector

$$\theta_z = cos^{-1}(L/600)$$

Eq 7. Angle of θ_z, 600 is the spacing between the navigation aids.

$$\theta_h = -\theta_a - (180 - \theta_z) = -\theta_a + \theta_z$$

Eq 8. Solving for heading, equation for navigation aid pari AB below the pair.

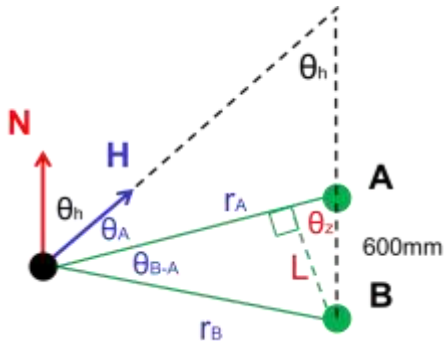

Figure 5. Heading calculation method of navigation aid pair AB. Relative North for the tank environment is labeled in red, and the measured values are in blue.

Both the heading and position calculation methods have met the project requirements of calculating heading within +/- 5 degrees and +/- 120 millimeters. When all processing is complete the platform can return position and heading at a rate of approximately 4 hertz. Which provides the controller with the timely and accurate information required to maintain position effectively.

E. Controller Implementation

Once position and heading is calculated it is fed into the controller. Once the controller receives a position and heading it compares it with the user input desired position and heading and calculates the error between the two. The next step is converting the error into appropriate x, y and rotational forces to move to the desired position.

To output appropriate commands a controller must accurately predict the platform's response to forces both translationally and rotationally. In order to do this the system must be realized. In order to realize this system, open loop testing must be completed. By testing how the platform moves through the water translationally and rotationally in both thruster directions, the most accurate depiction of the system will be realized. To expedite controller design coefficients from last year's project were used in the updated controller. These coefficients represent how the system should react depending on how far the system is from the desired position. By identifying proportional, integral, and derivative coefficients separately, a controller is capable of correcting both minor and major errors in position while avoiding overshoot. These coefficients are calculated separately in order to design a controller that would only use the proportional, integral, or derivative part depending on where the system is in relation to the set point. This P, I, or D discrete controller is polished so that it can be as responsive as possible.

One problem with the previous year's controller is that it assumed that the heading of the vessel was constant, and therefore two thrusters would always operate in the x-direction and two thrusters would always operate in the y-direction, however with the more effective position and heading calculation method the dynamic position platform can operate at any heading. This requires the controller to convert the desired x-y forces to thruster commands for any heading of the platform. This is accomplished by treating the heading of the platform, which is aligned with thrusters 1 and 4 as a different coordinate system. This allows the x and y translational forces to be projected onto a coordinate system aligned with the heading, resulting in two force vectors aligned with the thruster pair 1 and 4 and thruster pair 2 and 5. This allows a quick conversion from desired translation forces to thruster forces.

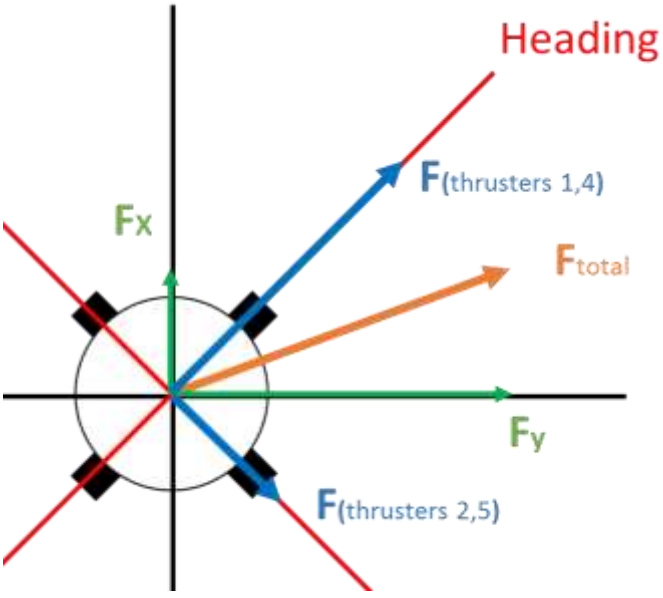

Figure 6. Two coordinate systems, the x-y system with desired x and y thrust and the system based on the heading of the platform, allowing for the total force to be projected onto the coordinate system aligned with the thruster pairs.

F. Hardware Interface

Direct Current Seabotix thrusters are used in order to move the platform translationally and rotationally. H-bridges connected to the DC thrusters and Programmable Logical Controller (PLC) are used in order to determine the direction of rotation of the DC thrusters. The H-bridges also allow the Pulse Width Modulation (PWM) duty cycle to be sent to the thrusters for force of thrust control, though the H-bridge only interprets commands. In order to send these commands to the H-bridges for further interpretation, a PLC is used for its PWM capabilities. We use an Arduino so that each thruster direction and PWM is controlled by a separate pin.

In order to make sure the PLC sends the right commands to the H-bridges, we have programmed it to interpret serial commands in a 5 character string that is output from the platform's controller. The first character determines which thruster we want to control, the second character is for direction, and lastly, the remaining three characters are for an intensity value ranging from 0-255. The Arduino Integrated Development Environment (IDE) was used to program the Arduino specifically in anticipation of these commands from a C# executable.

The purpose of programming the PLC to interpret serial commands is that it allows us to create our own protocol of controlling the thrusters using only one other language, C#. Resulting in efficient communications between the controller and thrusters.

IV. ANALYSIS

The LIDAR sensors are effectively and reliably communicating with the on-board computer. All three sensors acquire data at a rate no slower than 5 Hertz, this increase of speed is the result of using C# instead of MATLAB and was predicted by the students.

In addition to the sensor data acquisition and processing, heading and position calculations, which are also implemented in C#, are also being carried out at a rate of approximately 4 Hertz. This is much quicker than the 1.5 Hertz speed of the previous year's system. Calculating heading exceeds the specifications with an accuracy of +/- 4 degrees, which is within the required +/-5 degrees. Horizontal and Vertical (x-y) position within the tank is calculated within +/-40 mm out which is much less than the required +/-100mm. This position calculation is extremely accurate due to its inclusion of all available information in the new testing environment. Position is calculated using every range from every navigation aid for each of the three sensors. Ideally, 16 aids will be seen from one scan for the three sensors so that all redundant information is utilized to maximize the accuracy of the position calculation.

The dynamic mask was also tested for the system. A concern with the mask was that since it depended on the feedback of the last known position the mask would be calculated for a position offset from the actual one at the time of scan. However, after testing the offset between samples was not large enough to cause the mask to remove navigation aids from the scan.

While no controller testing has been carried out it is expected that by updating the controller and shifting it into C# instead of MATLAB there will be a significant performance increase.

V. CONCLUSIONS

In conclusion, C# proves to be much more reliable and efficient in comparison to MATLAB when dealing with real-time data acquisition and processing. Acquiring data using C# instead of MATLAB by reverse engineering the sensor's protocol, increases sampling frequency seven fold in some instances, but on average is twice the speed. This increase in sampling frequency allows the system for much improvement in real-time, by being able to respond to data much faster. With a C# designed executable, there is also no overhead, whereas in MATLAB there is overhead from the application running. Faster sampling times also are important for not just system design, but for discrete controller applications. With shorter times between each sample, the discrete controller will be able to respond more accurately to the actual conditions the system is experiencing. Although MATLAB may have an advantage with its complex mathematical built-in functions, some complex functions can simply be used through third party libraries. By using third party libraries for mathematical operations such as the Moore-Penrose Pseudoinverse, an accurate x-y position is calculated without the use of MATLAB. In addition, MATLABb may be easier to analyse data and more flexible in dynamically allocating memory for variables, but other methods such as file output can be utilized as a solution to the less flexible debugging of C# code.

In addition, an overdetermined least squares matrix solution, similar to GPS methods, for position calculation proves very accurate. This overdetermined least squares matrix solution, will use information about the surroundings from each of the sensors, weighted equally between sensors in order to determine an accurate position and heading. This solution is optimal because it ensures there is no bias in the calculations and that all information is used and not discarded

References

1. J. Meyers and A. Hoburg, "A Capstone Project on Robust Dynamic Positioning and Data Acquisition Systems,"

Proceedings of the 2015 ASEE Northeast Section Conference, Boston, MA, 2015.

2. C. Palmieri, J. Hooymans, and C. Gingrich, T. Emami, A. Dahlen, J. Staier "Electrical Engineering Capstone Project on Dynamic Position System," *Proceedings of the 2015 Zone 1 Conference*, Bridgeport, CT, 2014.
3. J. Paquette, T. Cogley, and T. Emami, A. Dahlen, R. Hartnett "Architecture of a Dynamic Position Autonomous Vessel," *121st ASEE Annual Conference & Exposition*, Indianapolis, Indiana, 2014.
4. D. Bray, The DP operator's handbook. London: The Nautical Institute, 2010.