HIGHLY INTEGRATED NAVIGATION ALGORITHM FOR ALFRESCO MOBILE ROBOTS

Christopher Stephen Tingley

MEng. Computer Science and Cybernetics, siu01cst@rdg.ac.uk

ABSTRACT

A team of six were formed to compete in the 2005 MEng. Robot Challenge and as such build a robot to autonomously navigate between a series of GPS (WGS84) coordinates. This paper discuses the author's work in designing and implementing a highly integrated navigation algorithm to marshal data between numerous types of sensors and actuators. The intermediate results of the project will be explored and a successful start to the project will be ascertained.

1. INTRODUCTION

1.1. The MEng. Robot Challenge

The MEng. Robot Challenge consists of three challenges evenly spaced throughout the year long duration of the project. Each subsequent challenge increases in difficulty to encourage improvements and additions to the project. Each challenge is timed, with the shortest total time for all three challenges determining the winning team. The project is loosely based upon the DARPA Grand Challenge [1].

The author's team consists of six members: Alex Sanchez, Christopher Forster, Daniel Aldridge, Peter Curd, Simon Cross and Christopher Tingley all of which have both distinct administrative and technical roles. The six administrative roles are (respectively) as follows: Technical Director, Financial Director, Health and Safety Consultant, Public Relations Consultant, Secretary and Project Manager. The technical role of the author was to design and implement the navigation algorithm used by the robot to move between coordinates.

1.2. The DARPA Challenge

The DARPA Grand Challenge is an annual event where a prize of $2 million (U.S.) is offered to the team which can develop an autonomous ground vehicle capable of completing a 175 mile off-road desert route in the fastest time.

Natural and man-made obstacles litter the route, which must be successfully navigated in order to complete the course and as such, the vehicle must be capable to determine, plan and execute suitable paths. Along the course lie a series of waypoints (checkpoints) which the vehicle must pass within a given distance. Each waypoint has this lateral offset associated with it, defining how close the vehicle must get within.

The MEng. Robot Challenge is very similar in many ways, except that the scale of the route is vastly smaller and the terrain is quite different. As a result; the areas of difficulty are slightly different from that of the Grand Challenge, in particular the accuracy required of the navigation system.

2. DESIGN

As introduced above, the navigation system must be able to accurately travel between two (or more) waypoints. In order to achieve this, the robot must have an accurate representation of its own location and be able to calculate the distance and bearing to its current target. To make this possible, the robot has been equipped with several hardware and software tools produced by the MEng Robot Challenge team:

2.1. Hardware Tools

2.1.1. Motor Drivers

Along with various other components, the motor driver boards were provided by the Department. These boards have a variety of different modes and through the use of optical encoders mounted to the motors, allow extremely precise control of the motors.

Knowing the number of encoder counts per wheel revolution and the circumference of the wheel, using simple mathematics it is possible to instruct the motor drivers to turn the wheels by a certain number of encoder counts (distance mode).

Another feature provided by the motor drivers is the ability to instruct the motor drivers to power the wheels at a certain number of encoder counts per control loop (velocity mode). Again, using a simple formula it becomes trivial to convert from meters per second into encoder counts per control loop and of course, visa-versa. Couple this with the ability to periodically return and calculate the current velocity and distance moved by each wheel, it is possible to use velocity mode to accurately control the robot's orientation and distance travelled in any given direction. This fine control over the robot is key to the successful implementation of the navigation algorithm, as discussed in 3.2.

2.1.2. GPS

An A12 GPS [2] board, provided by Thales Navigation provides a number of functions to determine the units position on the earth (latitude/longitude), the units orientation (degrees from north), speed, altitude (ellipsoidal height) etc. The unit is able to send this information up to once a second.

2.1.3. Electronic Compass

The CMPS03 electronic magnetic compass [3] is used in order to provide the most accurate orientation (bearing from magnetic north) of the robot.

2.1.4. Ultrasonics

A total of three SRF08 ultrasonic range finders [4] are used to provide information as to the robot's immediate environment and as such, are used for local object avoidance (LOA).

2.2. Software Tools

2.2.1. Coordinates

Within the robot, a number of different measurement units are used. The GPS unit returns WGS84 [5] coordinates, the motors and ultrasonics return distances in meters (after conversion) and the internal map's native units are Ordinance Survey OSGB36 [6] coordinates. As a result, the robot ends up basing calculations on a mish-mash of different units. For this reason, it was decided that only one measurement/coordinate system should be used. The coordinate system decided upon was OSGB36 which conveniently works in meters. This was for a number of reasons, the primary of which being the internal map as this already had OSGB36 reference lines. Secondly, OSGB36 coordinates are simply the number of meters from a known datum. This makes calculating distances between two points on the map a much simpler task.

Most of the internal units can be easily converted to meters, except the GPS (WGS84) coordinates. For this, the Grid InQuest DLL [7] provided by Quest Geo Solutions Ltd was used. This allowed extremely accurate conversions (within 2cm) between the ETRS89 [8] and OSGB36 coordinate systems. The ETRS89 conversion is adequate, as the only difference between this system and WGS84 is the datum point. As all readings are relative to a datum on the internal map, the discrepancy between the WGS84 and ETRS89 datum points (<37.5cm) is irrelevant.

2.2.2. Waypoints

An internal data structure is used to record information about coordinates. Whether this be a start point, an end point, or simply a strategic point on the map, it is stored within the software as a waypoint. Waypoints store the following pieces of information: x, y and z (latitude, longitude, ellipsoidal height), coordinate type (ETRS89 or OSGB36), maximum speed to safely approach the waypoint, lateral distance from current position, lateral offset, a Boolean value indicating if the waypoint has been reached and a string value containing a friendly name for the waypoint. Waypoints also store which waypoints neighbour it and also the costs, (c.f. 3.1) to travel to its neighbours.

2.2.3. The Map

To allow the robot to travel in a more 'intelligent' way, it is provided with a map of its environment. The map is a scale representation of the area in which the robot operates. It contains information as to where roads, paths and 'off-road' areas are located, along with a large number of strategic waypoints which provide information as to how to successfully navigate roads and paths. The map is also augmented in real-time with sensory information gathered from the robot. This information is used in LOA and is discussed in further detail in 3.2.3.

3. IMPLEMENTATION

The navigation algorithm requires the following information: an accurate position of the robot at that point in time and also the current destination. The prior is a topic outside the bounds of this paper, but the implementation of the latter can be broken down into two distinct areas, namely path planning and execution. Figure 1 is a graphical representation of the flow of data within the control software. As seen from the diagram, the input and output data is all mapped through the navigation algorithm at some stage.

3.1. Path Planning

To facilitate the path planning, the A* algorithm [9] (pronounced 'A' star) is used. This algorithm allows an optimal path (route) through a weighted graph (map containing connected waypoints) to be found. Once a route is found, the intermediate waypoints are saved and traversed in order.

3.1.1. The A Algorithm*

The A* algorithm uses all available waypoints on the map in order to generate an optimal path. This is done by evaluating the waypoints connected to the starting waypoint and picking the best waypoints to go via. The decision on which route to pick is based on adding both the cost to travel to a particular waypoint and the heuristic cost to get from there to the destination. The route with the lowest overall cost is selected.

The A* algorithm used is based upon a java tutorial [10], which provided the core functionality. Of course, the data structure and cost calculations had to be devised from scratch, as this implementation of A* is far from standard. In a standard application (usually computer gaming), the connections between nodes in the graph (waypoints on the map) are usually very simple. Imagine a chessboard, where a piece can move to any connected square. With this setup, the neighbouring squares are easy to find, and are all of fixed cost. With the map however, neighbouring waypoints are not at

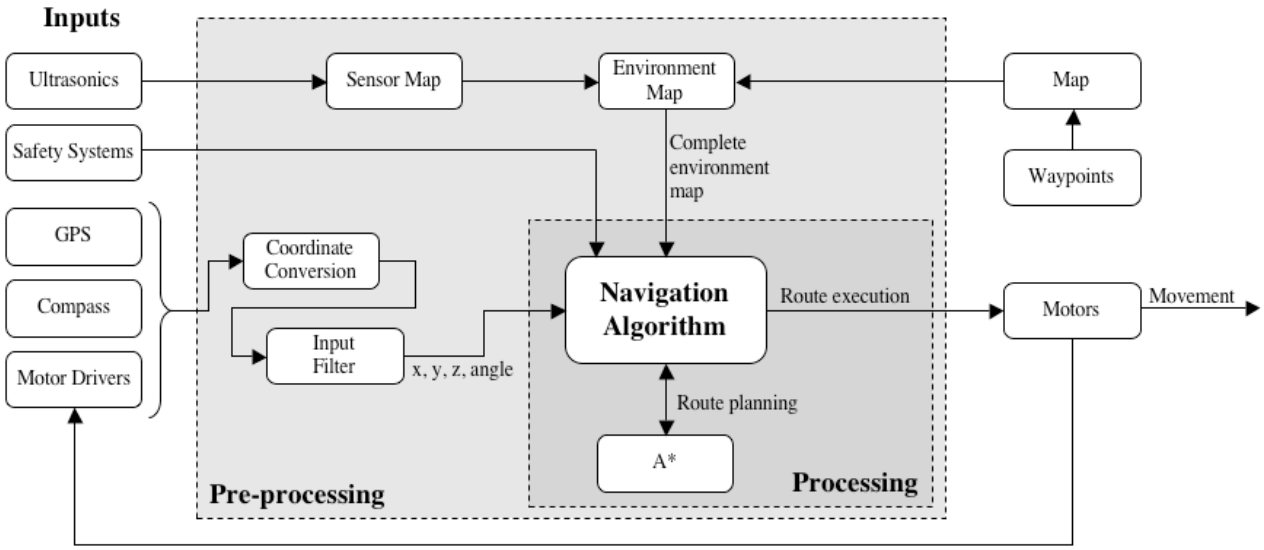

Figure 1. Navigation software architecture

discrete intervals and the costs are anything but fixed. To solve this, each waypoint stored on the map needs to know which other waypoints are its neighbours and also the cost to travel to each of its neighbours (see 2.2.2).

3.2. Path Execution

Once an intermediate waypoint is found, the job of the navigation system moves to one of getting to the desired waypoint. As introduced earlier; for the navigation algorithm to get to its destination, it must calculate three things: where the robot is, how it will get there and how it will avoid obstacles along the way (see Figure 3).

3.2.1. Where am I? - Data Filtering

The robot is able to calculate its position on the map by using the data available from the GPS, the motor drivers and the electronic compass.

When each new piece of location-based data is received, it is fed to the filter which provides an updated position of the robot. This happens in parallel with the navigation algorithm and facilitates the robot's ability to identify its position on the map.

3.2.2. How do I get there? - Dynamic Velocity Control (DVC)

To create a smooth trajectory, the velocity of both wheels are constantly updated. The velocities for both motors are calculated using a proportional controller. If a divergence from the desired path to the current waypoint is detected, the DVC will subtly change the wheel velocities in order to steer the robot back towards the target.

As the DVC uses a proportional controller, the further the deviation from the desired bearing, the faster the robot will turn. This enables the robot to make sharp turns when appropriate and is especially useful after the robot completes a waypoint, as the next waypoint is often at a completely different bearing.

3.2.3. How do I avoid things on the way? - Local Obstacle Avoidance (LOA)

While navigating between waypoints, it is vitally important that the robot does not come in contact with any stationary or non-stationary obstacles. For this reason, the environment map is augmented with ultrasonic data taken from the direct vicinity of the robot.

Object range from each of the three ultrasonic range finders is put through a sensor model and then an algorithm to create a certainty plane [11] (see figure 2). This certainty plane is then overlaid onto the map and is used in conjunction with the original map data to finely adjust the route in order to steer around any locally detected obstacles.

Figure 2. Sensor data shown on both a high-granularity [1cm resolution], sensor model (left) and then the same data on low-granularity [25cm resolution] (right) sensor plane. Dark colours represent a high probability of obstacles, whereas light colours represent a lower probability. White areas indicate where there are no obstacles.

The map is built upon an empty background, with coloured areas depicting regions where the robot is not permitted to travel and leaving empty regions as locations where the robot is allowed to be. The map is augmented with the sensor data, populating otherwise empty areas of the map with real-time information as to

the immediate environment (4 meter radius). The enhanced map is then used to assess the optimum route between the current point and the target point (straight line) and determine if the robot is able to traverse this route without colliding into any obstacles. If this is not possible, a new route as similar to the optimum route is generated. If this new route is deemed to be impossible, or indeed too costly, the route-planning algorithm is run again (v. inf. Fig. 3, L4-9).

```
L1.  repeat while not at destination
L2.      determine waypoints using A*
L3.      repeat while not at waypoint
L4.          update map
L5.          if path to waypoint is clear
L6.                move to waypoint using DVC
L7.           else if path to waypoint is possible
L8.                adjust route to waypoint using LOA
L9.           else find a new route using A*
```

Figure 3. Pseudo-code describing the navigation algorithm.

Due to the subsumption based architecture [12] of the software, the sensor data is also interpreted at a much lower level. This allows the range finders to 'over-ride' the map and slow down the robot immediately if a collision is imminent. This is known as "reactive" control [13] and is implemented (along with several other features), as a fail-safe to the navigation algorithm in accordance with the safety requirements of the project.

4. RESULTS

The first challenge consisted of autonomously traversing between three coordinates in a flat open area with no obstacles. The coordinates were published in advance of the challenge. The first generation navigation algorithm implemented was found to work very well as the robot was able to traverse between the points with no difficulty. At the time of publication the first challenge had been completed in 2 minutes and 43 seconds and preparations for the second challenge were well underway. Amidst these preparations is the development of the second generation algorithm which will include the remaining features discussed in this paper which were not required of the first generation algorithm.

5. CONCLUSION

At the time of publishing, the project was approximately half way through the allotted time frame. As such, any strong conclusions are difficult to draw. However, even at this stage of development, it is apparent that the different technologies and somewhat novel approaches discussed in 2, are able to be brought together in such a manner as to exhibit the ability to control an alfresco mobile robot in a partially known, dynamic environment.

Acknowledgments: As this project is being conducted as part of a larger team, the author would like to acknowledge the work done by the other five members of the MEng. Project team. The author would also like to thank the support staff within the Department of Cybernetics, namely: Mr. Iain Goodhew, Dr. Ben Hutt and Mr. Geoff Peace for their invaluable help with the more difficult (and frustrating!) aspects of the project.

6. REFERENCES

[1] (2004, Jan.). DARPA Grand Challenge, DARPA [Online]. Available: http://www.darpa.mil/grandchallenge/

[2] (2004). Thales Navigation A12 Board, Thales Navigation. UK. [Online]. Available: http://products.thalesnavigation.com/en/products/product.asp?PRODID=1003

[3] (2004). CMPS03 Electronic Magnetic Compass. Devantech, UK. [Online]. Available: http://www.robotelectronics.co.uk/html/cmps3doc.htm

[4] (2004). SRF08 Ultrasonic Range Finder. Devantech, UK. [Online]. Available: http://www.robot-electronics.co.uk/html/srf08tech.htm

[5] (1984). World Geodetic System 1984. Ordnance Survey, UK. [Online]. Available: http://www.gps.gov.uk/guide4.asp

[6] (1936). Ordnance Survey Great Britain. Ordnance Survey, UK. [Online]. Available: http://www.gps.gov.uk/guide5.asp

[7] (2004). Grid InQuest v6.0.8 3D Datum Transformation DLL. Quest Geo Solutions Ltd., UK. [Online]. Available: http://www.qgsl.com/software/gridiq.php

[8] (1989). European Terrestrial Reference System. Ordnance Survey, UK. [Online]. Available: http://www.gps.gov.uk/guide4.asp

[9] (2004, Apr.). P. Lester 'A* Pathfinding for Beginners'. [Online]. Available: http://www.policyalmanac.org/games/aStarTutorial.htm

[10] (2003, Sep.). D. Brackeen, 'Game Character Path Finding in Java'. [Online]. Available: http://www.informit.com/articles/article.asp?p=101142

[11] Elfes, A. 'Sonar-based real-world mapping and navigation', *IEEE Journal of Robotics and Automation*, v. RA-3, no.3, pp. 249-265, June 1987.

[12] Brooks, R. A. 'A robust layered control system for a mobile robot'. *IEEE Journal of Robotics and Automation*, 2:14-23, 1986.

[13] Murphy, R. R. 'Introduction to AI Robotics'. Cambridge: The MIT Press, 2000. (p. 113-122).