Smoothed A* Algorithm for Practical Unmanned Surface Vehicle Path Planning

Rui Song, Yuanchang Liu, Richard Bucknall
(*Department of Mechanical Engineering, University College London, Torrington Place, London WC1E 7JE, UK*)
Corresponding author: Yuanchang Liu. Tel: +44(0)2091089405. E-mail: yuanchang.liu.10@ucl.ac.uk

An effective path planning or route planning algorithm is essential for guiding unmanned surface vehicles (USVs) between way points or along a trajectory. The A* algorithm is one of the most efficient algorithms for calculating a safe route with the shortest distance cost. However, the route generated by the conventional A* algorithm is constrained by the resolution of the map and it may not be compatible with the non-holonomic constraint of the USV. In this paper an improved A* algorithm has been proposed and applied to the *Springer* USV. A new path smoothing process with three path smoothers has been developed to improve the performance of the generated route, reducing unnecessary 'jags', having no redundant waypoints and offering a more continuous route. Both simulation and experimental results show that the smoothed A* algorithm outperforms the conventional algorithm in both sparse and cluttered environments that have been uniformly rasterised. It has been demonstrated that the proposed improved A* route planning algorithm can be applied to the *Springer* USV providing promising results when tracking trajectories.

KEY WORDS

1. USV navigation. 2. Path planning. 3. A* algorithm. 4. Autonomous

1. INTRODUCTION: The benefits of deploying unmanned surface vehicles (USVs) in both civil and military applications have raised worldwide interest. USVs offer the benefits of reduced casualty risk and lower costs, as they can operate with minimal personnel involvement. Moreover, when compared with autonomous underwater vehicles (AUVs), USVs do not suffer from the constraint of GPS fixing as AUVs are required to surface regularly to correct errors introduced from having to utilise on-board dead reckoning navigation systems (Naeem et al., 2012). As a result of these benefits, USVs can be used in a myriad of tasks requiring high positional accuracy, such as weapons delivery, force multipliers, oceanography, environmental monitoring, shallow water surveying and supporting AUV operations by acting as a communication relay. A number of USV programmes have emerged in the last decade for a variety of these aforementioned purposes. *Springer* USV is one such research project highlighted in this paper.

The *Springer* USV project commenced in 2004 by the Marine and Industrial Dynamic Analysis (MIDAS) Research Group at Plymouth University (Xu, 2007). *Springer* is designed for undertaking shallow water missions, such as pollutant tracking, and is a test bed platform for other academic and scientific institutions' research (Naeem et al., 2006). To accomplish these missions, a robust navigation, guidance and control (NGC) system has been developed for *Springer*. Several research work programs have been undertaken to improve the reliability and accuracy of the NGC system. For example, a fuzzy logic based multi-sensor data fusion algorithm (Xu et al.,

2007) and an interval Kalman filter (Motwani et al., 2013) have been implemented for accurate positioning to navigate the vehicle. In addition, a linear quadratic Gaussian theory based controller (Naeem et al., 2006) and a genetic algorithm based model predictive controller (Sharma and Sutton, 2013) have been developed for the autopilot in the NGC system. However, from the previous reported papers, *Springer* either follows a predefined route or uses a line-of-sight (LOS) waypoint tracking strategy for route guidance. An intelligent path planning subsystem is currently absent and thus needs to be developed and integrated with the NGC system to better assure the safety of the USV.

Path planning algorithms can be categorised by two general approaches, viz. the stochastic approach (Smierzchalski 1999; Tam and Bucknall 2010; Tsou et al., 2010) and the deterministic approach (Naeem et al., 2012; Xue et al., 2011; Tam and Bucknall 2013). The deterministic approach, also known as the exact approach, follows a set of rigorously defined steps to generate a unique navigation path; while the stochastic approach, widely accepted as an approximate approach, only searches for an acceptable solution. Therefore, the output from the stochastic approach does not necessarily provide the best solution that satisfies the design requirements. As a consequence of its better consistency and completeness, the deterministic approach has become the dominant solution for maritime navigation systems. However, employing the deterministic approach could lead to several other practical problems. The generated path consists of a sequence of waypoints but may contain a significant number of course changes. Such a path may not be practical in real-time maritime navigation, where the path with the least course changes is generally preferred. Waypoints connected by line segments usually result in the need for sharp angled course changes, which can prove difficult for a USV to track precisely. In addition, the deterministic approach is costly in terms of the computational resources required to run. In (Goldberg and Harrelson, 2005), it has been shown that it takes substantial memory space for saving the significant number of solutions generated when applying the deterministic approach.

The path planning can further be classified into three stages, route planning, trajectory planning and motion planning, according to properties of autonomous unmanned systems. The intention here is to address the aforementioned problems by developing a practical route planning algorithm to be integrated with *Springer*'s NGC system, and report field testing results from data acquired from on-board sensors. The design of the route planning algorithm is developed from the well-known deterministic algorithm - A* algorithm (Hart et al., 1972). As a heuristic search, the A* algorithm is more efficient than other deterministic approaches (Oriolo et al., 1995) because it returns an optimal path with minimum cost, whenever one exists, thus guaranteeing completeness. The conventional A* algorithm is however limited to generate piecewise-linear path (Dolgov et al., 2010), neither smooth nor continuous. To implement the conventional A* algorithm on USV platform, (Zhu et al., 2013) developed a self-adaptive environmental model for inland water environment. The A* algorithm is applied on subdivided effective grids, which are selected by a presegmentation process. Also, to realise the avoidance of both static and dynamic obstacles, (Casalino et al., 2009) developed a three-layered hierarchical architecture path planning algorithm, where the obstacle is bounded in a box. With the A* algorithm being applied locally in the second layer, the start point when searching for a path is the USV's current position while the end point is one of the vertex of the obstacle bounding box. Although, these studies have succeeded in applying the conventional A* algorithm to generate the shortest trajectory in a short time, they did not consider the trajectory tracking problems and neglected the disadvantages such as the generated route being unsmooth.

To optimise the generated route in terms of smoothness, (Yang et al., 2015) proposed a finite angle A* algorithm, where the neighbouring nodes are increased to 16 and turning options are increased (0^o, 22.5^o, 45^o, 67.5^o and 90^o). Although such options can avoid sharp turnings, it sacrifices the computational memory space at each iteration. (Wang et al., 2018) developed a global path planning algorithm based on improved A* algorithm, where hexagonal grids (6 neighbours) are used instead of rectangular grids to increase the safety and rapidity. The generated path was smoothed by deletion of the second node among three consecutive nodes that clear of collision. However, evaluating collision for every node is redundant. (Kim et al., 2013) improved the conventional A* algorithm that considered the dynamic constraint of USV in a non-uniform map, but such variable scales will make locating become difficult when integrating with the data acquisition subsystem.

Therefore, to best retain the advantages of the conventional A* algorithm and integrate it into the Springer USV seamlessly, an improved waypoint based, A* algorithm has been developed, named the smoothed A* algorithm. The conventional A* algorithm has the advantage of being able to search for an optimal trajectory fast and efficiently. To retain this characteristic, the 4-cell geometry connection style is applied. In addition, to make the algorithm more suitable for USV navigation, three novel path smoothers are introduced. These are the line-of-sight path smoother (LOPS), the waypoint refining path smoother (WRPS) and the interpolation based path smoother (IPS). The LOPS is used to reduce unnecessary 'jags' produced by the conventional A* algorithm, the WRPS removes redundant waypoints along a straight line within the constraints of *Springer*'s dynamic behaviour, and the IPS uses a cubic spline interpolation method to make the path more continuous.

This paper is organised as follows: Section 2 outlines the NGC system and describes each subsystem in detail. The structure and implementation of the smoothed A* algorithm are discussed in Section 3. Section 4 briefly introduces the dynamic model of the *Springer* USV. Simulation and experimental results for off-line and real time path planning are presented in Section 5. Section 6 is the conclusion.

2. NAVIGATION, GUIDANCE and CONTROL SYSTEM: The *Springer* USV, as shown in Figure 1, is designed and constructed by the Marine and Industrial Dynamic Analysis Research (MIDAS) group of Plymouth University. *Springer* has a two-hull catamaran structure with the dimensions of 4.3 m in length and 2.3 m in width. Eight 12 V 135 A h batteries, which are placed within the hulls with four in each side, provide power for the electrical motors to drive two propellers to propel the vehicle and all the electronic devices. Two Peli-cases with guaranteed waterproof capability are placed on the hulls to contain the Navigation, Guidance and Control (NGC) system. Figure 2 illustrates the autonomous NGC system structure of *Springer* USV. It consists of three modular subsystems:

1) **Navigation data processing subsystem**. This module acquires the necessary navigational data, including that of the USV itself and other ships, from various sensors. Common navigational devices such as GPS, electronic compass, inertial measurement unit and gyroscope have been implemented. In addition, to increase the robustness, a complement module consisting of IMU and stereo camera has also been developed to determine the USV's location if other sensors malfunction. All the information obtained is processed and merged via a data fusion process to improve information accuracy. As this paper focuses on the development of path planning algorithms, details of the data fusion process will not be introduced here, but can refer to work presented in (Liu et al., 2015 and Motwani et al., 2016). The processed information is then passed to the path planning subsystem to assist with the path planning task.

2) **Path planning subsystem**. This subsystem is a new addition to the NGC system of *Springer*. With the information provided from the navigation data subsystem, the smoothed A* algorithm is applied by this subsystem to generate an optimal collision free path as the reference trajectory route to inform the initial navigation.

3) **Autopilot subsystem**. The autopilot employs a closed loop controller to calculate the USV's heading according to the generated waypoints. The generated manoeuvres include the changing of the heading as well as the speed of USV. By conducting tests, the PID controller was tuned to produce the most robust trajectory tracking results for *Springer's* autopilot (Motwani, 2015).

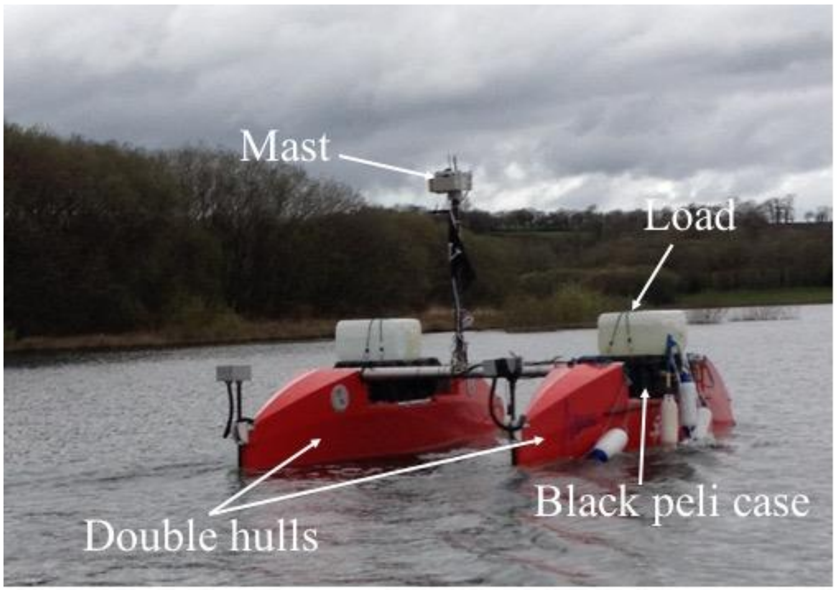


Figure 1. *Springer* USV.

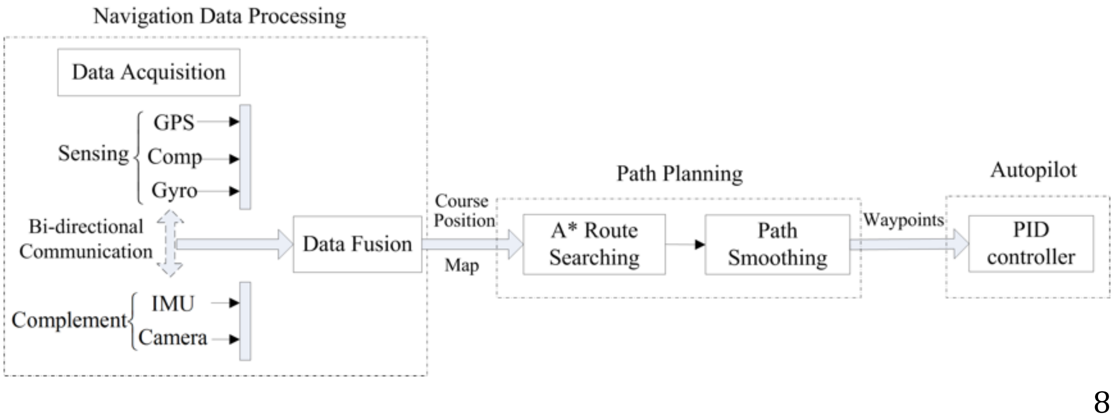


Figure 2. The NGC system structure of *Springer* USV. There are three subsystems, namely the Navigation Data Processing subsystem, the Path Planning subsystem, and the Autopilot subsystem.

3. PATH PLANNING ALGORITHM: It has been proved that the conventional A* algorithm will find the shortest path in applications of robotics (Oriolo et al., 1995; Kala et al., 2010 and Boh´acs et al., 2016). However, the conventional A* algorithm uses a grid map where the resolution depends on the grid cell size and this can lead to several practical problems for USV navigation control:

1) The generated path may contain a significant number of turns which are not practicable in real-time maritime navigation, where the path with the least number of turns is generally preferred. Although the number of turns could be reduced by using a grid map with lower resolution, the resulting generated path may not be optimal, especially in a complex environment. This observation is supported by the performance comparisons when using the conventional A* algorithm with different grid map resolutions, as shown in (Chiang et al., 2007).

2) The generated path often consists of a sequence of line segments connected by waypoints where sharp turns are encountered. Such a path could be difficult for the piloting control dynamics of a USV to track precisely. Therefore, to facilitate and improve USV navigation, the conventional A* algorithm would need to be improved by the addition of a path smoothing process.

While developing the improved A* algorithm, not only do the optimal criteria, such as distance and time costs need be considered, but additional constraints, such as the USV dynamics, must also be taken into account to make the generated path more commensurate with practical requirements, constraints and limitations.

The structure of the improved A* algorithm is shown in Figure 3, which includes three steps: (1) Image processing to generate a feasible map; (2) applying the conventional A* algorithm to search for a route; and (3) path smoothing to produce a more practically feasible path. The algorithm first receives fused navigational information, i.e. the environment map from the upper level subsystem of the USV. The algorithm extracts the environment map information to generate a binary grid map and identifies the locations of static obstacles in the image processing step. Meanwhile, the fused navigational data is used for mapping the positions of the USV itself on to the generated grid map. Once all the information is processed, the conventional A* algorithm is applied in the second step to calculate the shortest safe path, which consists of a series of waypoints, including the mission start and mission end points. The path smoothing process is then applied to reduce any unnecessary waypoints making the path more continuous. Note that the path smoothing process is undertaken after the conventional A* algorithm finishes searching the least cost route to largely reserve the fast computational speed of the A* algorithm. Finally, the smoothed path with the information of waypoint coordinates is sent to the lower level subsystem, such as the autopilot subsystem, to control the USV and to track the generated path.

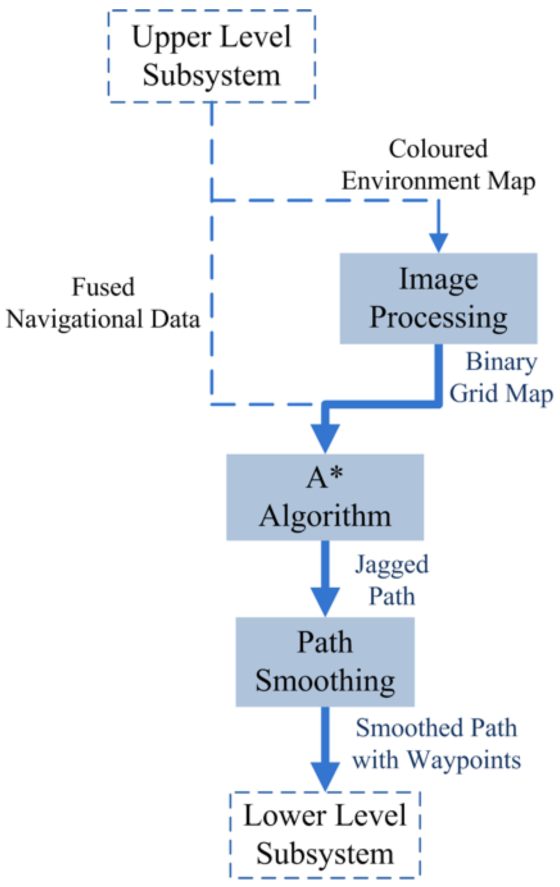


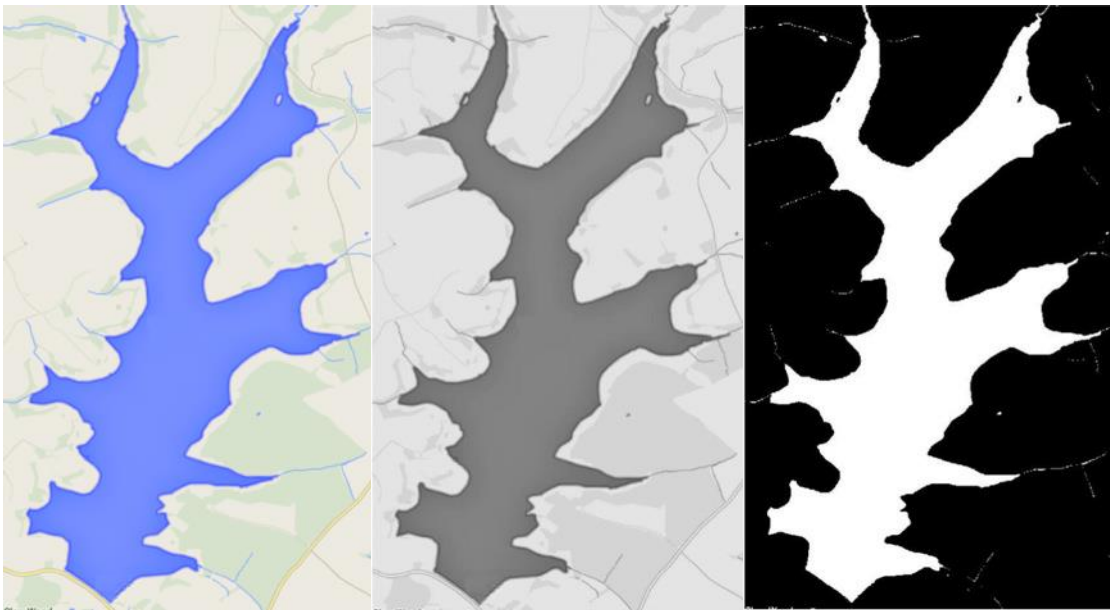
Figure 3. Smoothed A* algorithm procedures. The algorithm consists three procedures, i.e. image processing, A* searching and path smoothing.

3.1. Image processing: The conventional A* algorithm runs on a rasterised map, which is normally referred to as a grid map. However, in practical maritime navigation, the map normally provided is a marine navigation chart, and therefore requires image pre-processing to effectively generate the grid map while maintaining the essential original information. There are two main techniques applied in the image processing step to better inform the algorithm of the surrounding environment. One technique is map transformation. This utilises the Otsu algorithm (Otsu, 1975) and Luminance method, which are mainly used to distinguish the obstacles by transforming a colour map first to a grey scale map and then to a binary grid map. The other technique is the coordinate transformation method, which is applied to convert the navigational data represented in the earth frame to the screen coordinate frame

- **Otsu algorithm & Luminance method**

The Otsu algorithm is a well-known method to generate a binary image from a grey scale image in computer vision by distinguishing the foreground and background pixels. As the geographic information extracted from a marine navigation chart is normally coloured, a luminance method is used to obtain the required grey-scale information. Figure 4. shows an example of converting a coloured image to a binary image.

The final step of the image processing is to obtain the grid map by rasterising the binary image map. In such a case, a configuration searching space (C_{space}) can be obtained in such a way that C_{free} is the place where the grid cell value is equal to 1 (white colour), and C_{obs} has a grid cell value that is equal to 0 (black colour). This makes the risk assessment of collision to be dependent upon the logic value of the space only. Note that, there is a trade-off between the computational time and the resolution of the grid map. The higher the resolution, the longer the computational time required. An acceptable size of each grid cell must be selected for the rasterization. Normally, in robotic studies, when choosing the size of grid cells, it should follow the rule that a robot can move from one grid to its neighbouring grids (Miklic et al., 2012). However, different from robot, USVs are usually non-holonomic systems making their manoeuvrability relatively low. In particular, when taking a turning manoeuvre, a USV's turning capability depends on the turning radius, which is defined by vehicle's dynamic model. When calculating the radius, according to the Standards for Ship Manoeuvrability, the minimum turning radius is 5 times the ship's length for a large vessel and 3 times that for a small sized vessel. Therefore, in this paper, each grid cell size is set at 12 m $\times$12 m, which is around 3 times the overall length of Springer (4.3m). Based on this configuration Springer will be able to make starboard and port side turns of up to 180° within a grid cell.



a) b) c)

Figure 4. Example of grid map generation. Image processing for the map of Roadford Lake. (a) Colour image format. (b) Grey-scale map. (c) Binary image format.

- **Coordinate transformation**

When running the algorithm in real-time, the current USV location on the grid map needs to be provided from the data acquisition subsystem at each time step. The USV location, as determined from the sensors, is represented in the earth frame (values of latitude and longitude). There is therefore a requirement to convert the earth coordinate frame into a screen coordinate frame. By defining the height (h) and width (w) of the screen, in actuality the size of the grid map, the transformation equations and boundary conditions are given by:

$$x_{ref} = \begin{cases} (lon\ \text{-}\ minLon)*3600/scaleX & \text{when } lon \neq minLon\ \&\ lon \neq maxLon \\ 0 & \text{when } lon = minLon \\ w & \text{when } lon = maxLon \end{cases} \quad (3.1)$$

$$y_{ref} = \begin{cases} (lat\ \text{-}\ minLat)*3600/scaleY & \text{when } lat \neq minLat\ \&\ lat \neq maxLat \\ 0 & \text{when } lat = minLat \\ h & \text{when } lat = maxLat \end{cases} \quad (3.2)$$

where x_{ref} and y_{ref} are the referred coordinates on the environment map in the screen coordinate frame. *lon* and *lat* are the fused GPS information. The *scaleX* and *scaleY* are conversion ratios along the *X* and *Y* axes, given by:

$$scaleX = ((maxLon\text{-}minLon)*3600)/w \quad (3.3)$$

$$scaleY = ((maxLat\text{-}minLat)*3600)/h \quad (3.4)$$

where, maxLon, *minLon*, *maxLat* and *minLat* are the geographic coordinates boundary values of the environment map represented in the earth frame. 3600 represents the conversion from the form (degree,degree,degree) to (degree, miniute,seconds). For example, 1 degree*60→miniutes*60→seconds.

To transform pixels into meters, it can be first calculated using latitude and longitude values. The expression is given as:

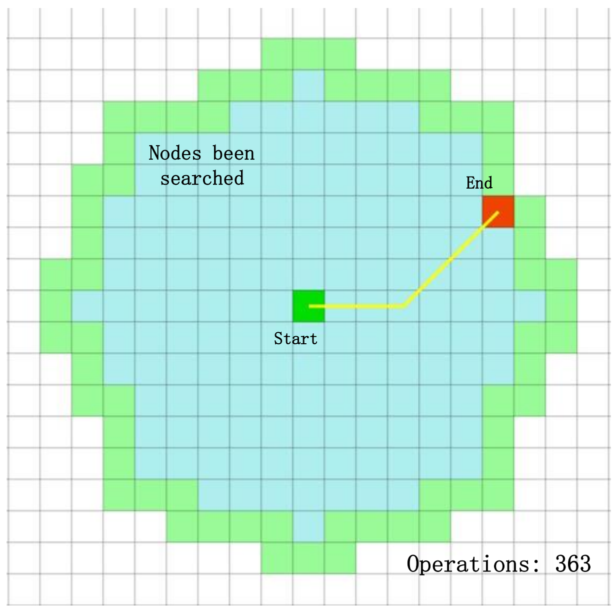
$$\begin{gathered} a = \sin^2(\Delta\phi/2) + \cos\phi_1 * \cos\phi * \sin^2(\Delta\lambda/2) \\ c = 2 * \tan^2(\sqrt{a}/\sqrt{1\text{-}a}) \\ d = R*c \end{gathered} \quad , \quad (3.5)$$

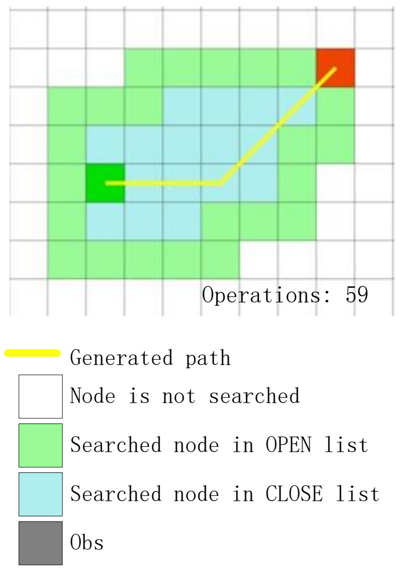
where *d* is the length in meters, ϕ is latitude, λ is longitude, *R* is earth's radius equalling 6,371 km, (note that angles are represented in radians). To further transform between pixels and meters: 1 pixel = *d*/*w* in the horizontal axis, and 1 pixel = *d*/*h* in the vertical axis.

3.2. 4-GEOMETRY A* ALGORITHM: The conventional A* algorithm has been implemented to calculate the least distance path which consists of a finite ordered sequence of waypoints from the mission start point to the mission end point. It plans a path from a start node $\boldsymbol{X}_{start}(s)$ to an end node $\boldsymbol{X}_{end}(s)$, where s is the set of states including the USV's heading angle θ and coordinates (x,y). To calculate the path, it stores an estimated distance cost from $\boldsymbol{X}_{start}(s)$ to each node $\boldsymbol{X}(s)$ denoted as $f(\boldsymbol{X}(s))$ with the initial value for all the nodes on the grid defined as $f\,(\boldsymbol{X}(s)) = \infty$. The algorithm begins the search by setting the cost of $\boldsymbol{X}_{start}(s)$ to be 0, then places $\boldsymbol{X}_{start}(s)$ into a priority queue known as the OPEN_list. Each element $(\boldsymbol{X}(s))$ in this queue is ordered according to the sum of its current distance cost from the start point, denoted as $\mathrm{g}(\boldsymbol{X}(s))$, and a heuristic estimation of the distance cost to the end point $(\boldsymbol{X}_{end}(s))$, denoted as $h(\boldsymbol{X}(s))$. Such a sum represents the local distance cost function, expressed as:

$$f(\boldsymbol{X}(s))=g(\boldsymbol{X}(s))+h(\boldsymbol{X}(s)). \tag{3.6}$$

The node with the minimum value of $f(\boldsymbol{X}(s))$, denoted as $\boldsymbol{X}_n(s)$, is placed at the front of the OPEN_list. The heuristic $h(\boldsymbol{X}(s))$ evaluates the cost of the optimal path from $\boldsymbol{X}(s)$ to $\boldsymbol{X}_{end}(s)$ and is used to guide the search towards to $\boldsymbol{X}_{end}(s)$. Figure 5 shows that using the heuristic cost function, the efficiency, in terms of the computational time and the memory space is improved.





(a) (b)

Figure 5. Efficiency comparisons when considering and without considering the heuristic cost. Search from the same start point (green square) to end point (red square). (a) Without heuristic guidance, the number of iteration is 363. (b) With heuristic guidance, the iteration number reduced to 59. The searched points stored in the memory space are shown in light green and blue cells.

After selecting $\boldsymbol{X}_n(s)$ from the queue, the algorithm then updates the cost of all neighbouring nodes that can be reached directly from $\boldsymbol{X}_n(s)$. The cost is calculated by adding the current cost of the node $\boldsymbol{X}_n(s)$ ($g(\boldsymbol{X}_n(s))$) and the distance between $\boldsymbol{X}_n(s)$ and its neighbouring node $\boldsymbol{X}_i(s)$, denoted as $d(\boldsymbol{X}_n(s), \boldsymbol{X}_i(s))$. This cost, namely $g(\boldsymbol{X}_n(s))+ d(\boldsymbol{X}_n(s), \boldsymbol{X}_i(s))$, will be used as the updated cost of $\boldsymbol{X}_i(s)$ if it is less than the node's current cost. If the cost of a neighbouring node $\boldsymbol{X}_i(s)$ changes, $\boldsymbol{X}_i(s)$ will be placed in the OPEN_list. It should be noted that the algorithm continues selecting the node with the lowest value from the OPEN_list and places it into a CLOSED_list. During these iterations, if the heuristic is appropriate and admissible, i.e. guaranteed to not overestimate the cost between two nodes, then the path cost from $\boldsymbol{X}_{start}(s)$ to $\boldsymbol{X}_{end}(s)$ is guaranteed to be optimal. Once $\boldsymbol{X}_{end}(s)$ has been selected and moved to the

CLOSED_list, the cost updating process within the algorithm terminates.

In order to find the optimal path, which consists of a sequence of waypoints, each node keeps a record of its parent node ($\boldsymbol{X}_{i.parent}$) or predecessor during the iterations. The final path will be searched from $\boldsymbol{X}_{end}(s)$ by iteratively following each node's predecessor until $\boldsymbol{X}_{start}(s)$ is reached. The corresponding pseudocode is described in Algorithm 1.

The time complexity of the conventional A* algorithm depends on the number of nodes (N_{open}) that have been extracted from the OPEN_ list. The time complexities for expanding the successor nodes is $\mathcal{O}(N_{open})$, and for resorting is $\mathcal{O}(N_{open}\log(N_{open}))$, if a fast sorting method is applied. Therefore, the time complexity for the complete A* algorithm is $\mathcal{O}(N^2_{open}\log(N_{open}))$.

Algorithm1 A* algorithm

```
Input: start node $X_{start}$, end node $X_{end}$
1.          OPEN_list := $X_{start}$, where $f(X_{start})= h(X_{start})$
2.          CLOSE_list := { }
3. while    OPEN_list is not empty do
4.          current node $X_n$ := the node in the OPEN_list with the lowest $f(X)$
5.          if   $X_n = X_{end}$   break
6.          Remove $X_n$ from OPEN_list and add it to CLOSE_list
7.          for   each adjacent node, $X_i$ of $X_n$ do
8.                if $f(X_i)=0$ || $X_i \in$ CLOSE_list   continue
9.                if $X_i \notin$ OPEN_list
10.                     add $X_i$ into OPEN_list
11.                     the parent node of $X_i$, $X_{i.parent} = X_n$
12.                     calculate $f(X_i)$, $g(X_i)$ and $h(X_i)$
13.               if $X_i \in$ OPEN_list
14.                     $g(X_i)'=g(X_i)+d(X_n,X_i)$
15.                     if $g(X_i)' \geq g(X_i)$   continue
16.                     else
```

```
17.                 $\boldsymbol{X}_{i\cdot parent} = \boldsymbol{X}_n$
18.                 $g(\boldsymbol{X}_i)=g(\boldsymbol{X}_i)'$
19.                 $f(\boldsymbol{X}_i) = g(\boldsymbol{X}_i)+ h(\boldsymbol{X}_i)$
20.         Resort and keep OPEN_list sorted by $f$ values
21. $\boldsymbol{X}_p = \boldsymbol{X}_{end}$
22. Path_list := $\boldsymbol{X}_p$
23. while  $\boldsymbol{X}_p \neq \boldsymbol{X}_{start}$ do
24.         $\boldsymbol{X}_p = \boldsymbol{X}_{p\cdot parent}$
25.         Path_list = {Path_list, $\boldsymbol{X}_p$}
Return: Path_list
```

- **Cell connection**

Generally, 2-geometry cell connection is used to represent changes of direction along the path. As shown in Figure 6(a), the turning degree of 2-cell geometry (4 neighbours) is 90°. However, it is impractical to fix and limit any and all heading changes to 90° for path change manoeuvres. To overcome this limitation a 4-cell geometry (8 neighbours) connection is used to improve the path property throughout this work. As shown in Figure 6(b), along with the 90° turn, a 45° turn is also available as an option of manoeuvre. In this way, the efficiency can be improved since the number of points to be evaluated has increased from 4 to 8 during each iteration.

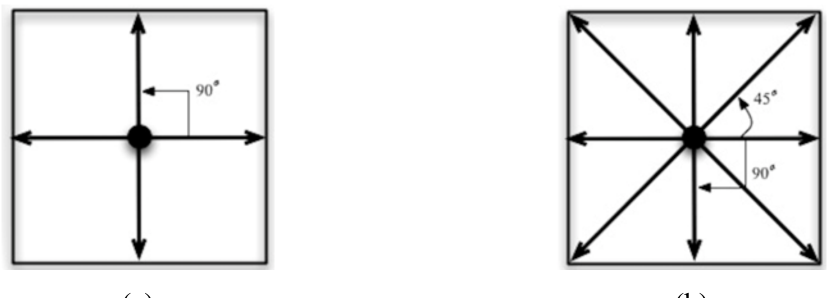


Figure 6. Grid cell connection styles. (a) 2-geometry cell connection scheme, searches 4 neighbours in each expansion. (b) 4-geometry cell connection scheme, searches 8 neighbours in each iteration.

3.3. PATH SMOOTHING: The trajectory provided by the conventional A* algorithm usually consists of a number of short line segments connecting a series of waypoints. As shown in Figure 7, there are two major shortcomings associated with the generated path. First, the main focus of the A* algorithm is to minimise distance cost in such a way that the jags generated are minimised. It sacrifices turning cost in order to obtain the shortest path. Second, the path obtained is not navigationally continuous, which may not be practicable for the USV's autopilot subsystem to track. Therefore, to overcome these disadvantages novel and advanced path smoothing algorithms have been developed. As the path smoothing process is applied after path searching, the time complexity depends on the number of waypoints existed along the route after each process.

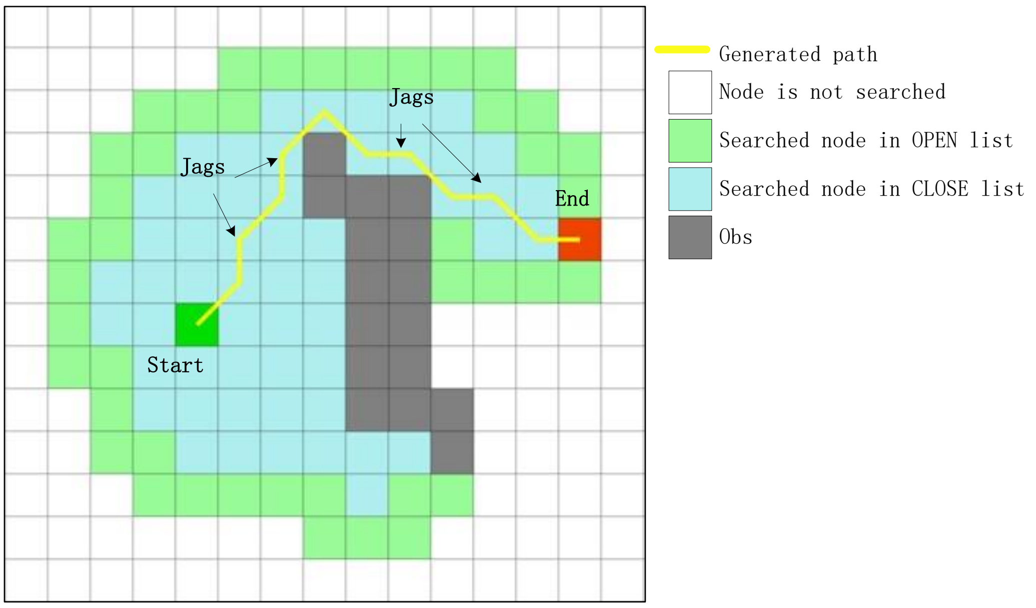


Figure 7. Path generated by the conventional A* algorithm with jags. To avoid obstacles (Obs, grey nodes), a generating path with jags is inevitable.

To increase the practicability of the path, there are three main objectives that need to be achieved: 1) To achieve soft acceleration: without smoothing， may lead to unstable motion and over-actuation particularly at sharp turning points. 2) To save energy less turning reduces the energy consumption for manoeuvring. 3) To track the trajectory more robustly a continuous path is easier for the autopilot to track than discrete waypoints which may require the USV turn too sharply.

Figure 8 shows the flowchart of the path smoothing algorithm with three individual path smoothers (PS): The line-of-sight path smoother (LOPS), the waypoint refining path smoother (WRPS) and the interpolation based path smoother (IPS).

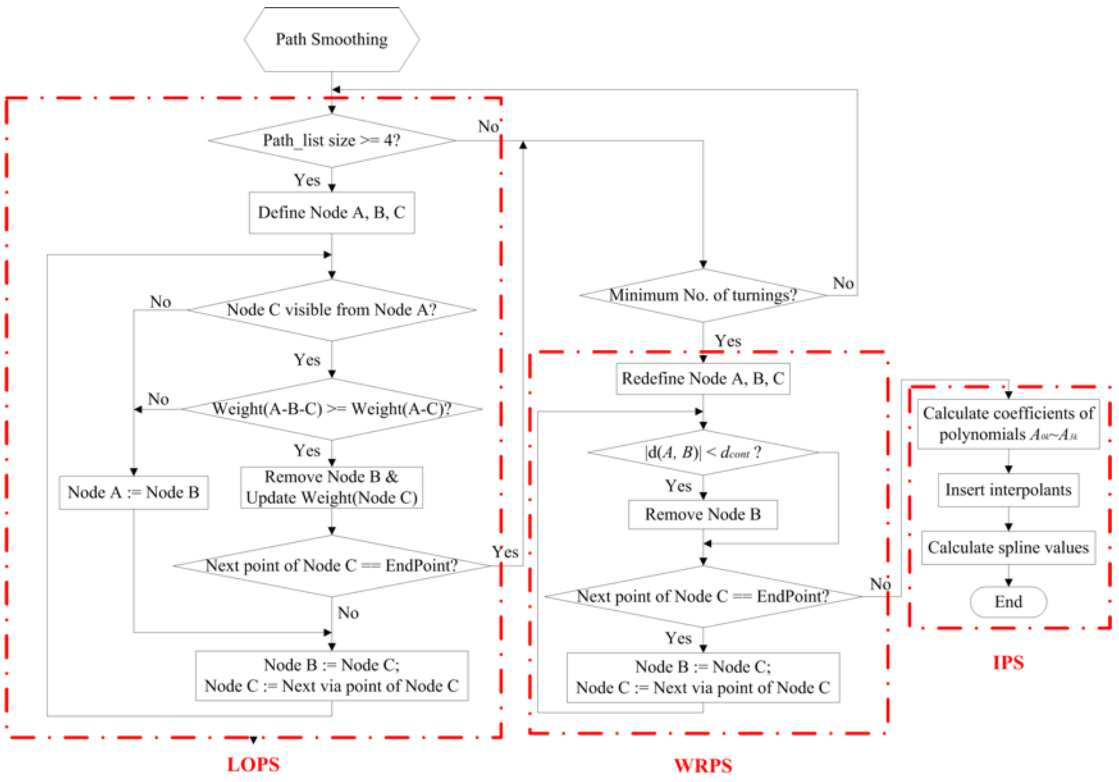


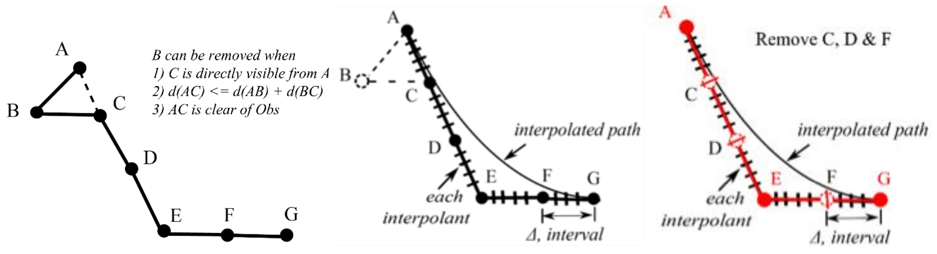
Figure 8. Flow chart outlining the path smoothing algorithm.

- **Line-of-sight path smoother (LOPS)**

The LOPS is applied first to reduce the number of turning points after the conventional A* searching procedure. It should be noted that the precondition to use such a smoother is that there must be at least four points in the path list. Therefore, the first action is a check of the number of path points generated from A* algorithm. An iteration is then carried out from the first point to the last. For each iteration, the LOPS checks all nodes in consecutive groups of three, (1) the current checking point, denoted as point A; (2) A's next via point in the path, denoted as point B; and (3) the next via point of B, denoted as point C. Figures 9(a) and 9(b) illustrate how this method works. Assuming the original sequence of points in the path list is A-B-C-D-E-F-G, and it satisfies two conditions:

1) Point C is directly visible to A as shown in Figure 9(a);
2) Distance between A and C is less than or equal to the sum of the distances between A and B and the distance between B and C.

If the above conditions are satisfied point B will be removed from the path. The new path becomes A-C-D-E-F-G as shown in Figure 9(b). In such a case, the number of points are reduced after each loop. The LOPS will be terminated until the number of turnings has been reduced to the lowest number feasible through application of the LOPS. However, the path still remains as a series of waypoints connected by rigid straight lines. The IPS is therefore applied to make the path more continuous.



(a) (b) (c)

Figure 9. Path smoothing process. (a) The original path, where point B is unnecessary. Due to the nature of grid search limitation, such 'jag' on point B is inevitable in conventional A*

algorithm, as shown in Figure 7. (b) LOPS, where point B has been removed. (c) WRPS, where points C, D and F have been removed; and the smoothed curve path is generated by IPS.

- **Waypoint refining path smoother (WRPS)**

Before applying the IPS, redundant waypoints may exist along a straight path segment such as points C and D shown in Figure 9(b). From previous empirical experiments, it has been found that these waypoints could be allied to adverse vehicle tracking. Closely placed waypoints will generate extra control points, which could lead to degraded tracking performance. To overcome this problem, a waypoint refining path smoother (WRPS) has been specifically designed based on the condition given as:

$$|d(\boldsymbol{X}_{n+1})\text{-}d(\boldsymbol{X}_n)|<d_{cont} \tag{3.7}$$

If Equation 3.7 is satisfied, i.e. the distance between two adjacent waypoints, $\boldsymbol{X}_n$ and $\boldsymbol{X}_{n+1}$, is smaller than the optimal control distance, denoted as d_{cont}, the redundant waypoint $\boldsymbol{X}_{n+1}$ is removed. From previous experiments, it was found that although the size of grid cell is well defined, during the navigation that a large heading change is required, compromised tracking results may be generated especially between two adjacent waypoints that are relatively close to each other. This is mainly due to the limitation of vehicle's tuning capability. Therefore, to compensate such a tracking error and achieve better performance, the path planning algorithm that can provide optimal number of waypoints is required and the optimal control distance d_{cont} is introduced. The value of d_{cont} is determined from the performance of pre-done experiments in identifying Springer's dynamic model. In addition, the value of d_{cont} is experience based and related to the operating environment (may be influenced by the hydrodynamic forces).

- **Interpolation path smoother (IPS)**

With the aim to further improve the continuity of the path, a cubic spline interpolation is used. From among a number of interpolation methods, the cubic spline is employed because its feature of least deviation from the original path (Fossen, 2002). It is a process of approximation to depict a parametric curve from the discontinuous path, as shown in Figure 9(c).

The main rationale of the cubic spline method is the use of a spline consisting of third order polynomials to connect a set of control points (or waypoints on a path). The interval between two control points has the form of polynomial function, which is defined by:

$$f_k(\mathrm{x}) = A_{3k}(\mathrm{x}-\mathrm{x}_k)^3+A_{2k}(\mathrm{x}-\mathrm{x}_k)^2+A_{1k}(\mathrm{x}-\mathrm{x}_k)+A_{0k} \tag{3.8}$$

where k is the index of the control points. By giving the coordinate values of each control point, the polynomials' coefficients (A_{0k}, A_{1k}, A_{2k} and A_{3k}) can be calculated using the following constraints:

1) Cubic polynomials match the values of the equation at both ends of the interval (between two control points) $[\boldsymbol{X}_k,\ \boldsymbol{X}_{k+1}]$, namely: $f_k(\mathrm{x}_k)=y_k$ and $f_k(\mathrm{x}_{k+1})=y_{k+1}$.

2) The first (velocity) and second (acceleration) derivatives of the cubic spline function at each interval control point have the same value making the generated path continuous, namely $f'_{k-1}(\mathrm{x}_k)=f'_k(\mathrm{x}_k)$ and $f''_{k-1}(\mathrm{x}_k)=f''_k(\mathrm{x}_k)$.

3) The first derivatives at the start and end points are equal to the initial and final velocities; whereas, the second derivatives at the start and end points are zero.

4. *SPRINGER* DYNAMIC MODEL: Figure 10 shows the block diagram of the *Springer* USV control model. The model has two inputs that represent the rotational speeds of each propeller, denoted as n_1 and n_2 in revolutions per minute (rpm). A single output represents the USV heading. For tracking along a straight line, the vehicle requires both propellers to rotate at the same rpm but while turning motion the two propellers will have a rotational speed differential (Annamalai et al., 2013). Two modes of thruster velocity can therefore be defined as the common mode (n_c) and differential mode (n_d), and be expressed respectively by:

$$\begin{cases} n_c = \frac{n_1 + n_2}{2} \\ n_d = \frac{n_1 - n_2}{2} \end{cases} \tag{3.9}$$

To obtain the dynamic model for the *Springer* USV, a system identification (SI) technique has been applied noting that the dynamic model for the USV changes with the environment (Naeem et al., 2006). The MATLAB model for *Springer* USV is obtained using the SI toolbox analysing the data obtained from several pre-done field tests. For data acquisition purposes, several inputs including a pseudorandom binary sequence were applied to the thrusters and the heading response was recorded. The input to the SI is the differential mode thruster velocity n_d, which causes the vehicle to manoeuvre as required (n_c is maintained to conserve the operating regime). The acquired data were processed and filtered at 1 1 Hz since this frequency was deemed to be adequate for *Springer's* controller design noting that the dynamic model for the USV changes with the environment (Naeem et al., 2006).

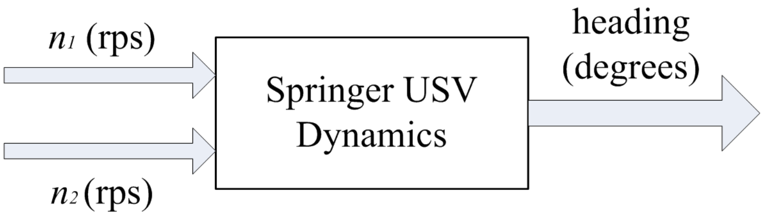


Figure 10. Block diagram of the USV model with two inputs and one output.

Prior to testing the performance of *Springer*'s NGC system with the smoothed A* algorithm integrated, several model identification trials were carried out. *Springer* was put through some predetermined manoeuvres, during which data such as vehicle heading was recorded using a digital compass. During these trials, a forward speed of 3 knots was maintained, i.e. n_c = 900 rpm, while n_d was varied to steer the vessel. All the recorded data were analysed by the built-in MATLAB SI function using the steering model expressed as:

$$\begin{cases} x(k+1) = \mathbf{A}x(k)+\mathbf{B}u(k) \\ y(k) = \mathbf{C}x(k)+\mathbf{D}u(k) \end{cases} \tag{3.10}$$

in which the sampling time is 1 s, $u(k)$ represents the differential thrust input in rpm and $y(k)$ represents the heading angle in radians. The calculated values of the coefficients of matrices **A**, **B**, **C** and **D** are:

$$\mathbf{A} = \begin{bmatrix} 1.1130 & 0.3519 & -0.4221 & -0.04596 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0.004301 & -0.003881 & -0.001648 & 0.001247 & 1 \end{bmatrix}$$

$$\mathbf{B} = [1 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$\mathbf{C} = [0 \quad 0 \quad 0 \quad 0 \quad 1]$$

$$\mathbf{D} = 0$$

This model, with the specific coefficients matrices, is then applied for the simulations and experiments to test the performance of using the smoothed A* algorithm.

5. RESULTS: This section presents the results from both the actual trials using *Springer* USV and the computer based simulations. To best verify the effectiveness of the algorithm under simulated conditions, the practical environment extracted from the Google map has been utilised. The practical field trials have been undertaken on Roadford Lake which offered the opportunity of testing under a range of weather conditions. The specific configurations for simulations and experiments are as follows:

1) Simulations of off-line path planning using a practical environment to verify the path smoothing performance of the algorithm (Section 5.1).
2) Simulations of off-line path planning when integrating with *Springer's* NGC system in a practical environment (Section 5.2).
3) Experiment of off-line path planning when integrating with *Springer's* NGC system in a practical environment (Section 5.3).

The smoothed A* algorithm has been coded in JAVA and MATLAB. Simulations were run on a computer with a Pentium i7 3.4 GHz processor and 8 GB of RAM. In the simulations, it was assumed that the water current is 1 m/s flowing south to north. To simplify the implementation of the algorithms, some assumptions are predefined:

- The USV is considered as a mass point., in such case the hydrodynamic forces can be neglected.

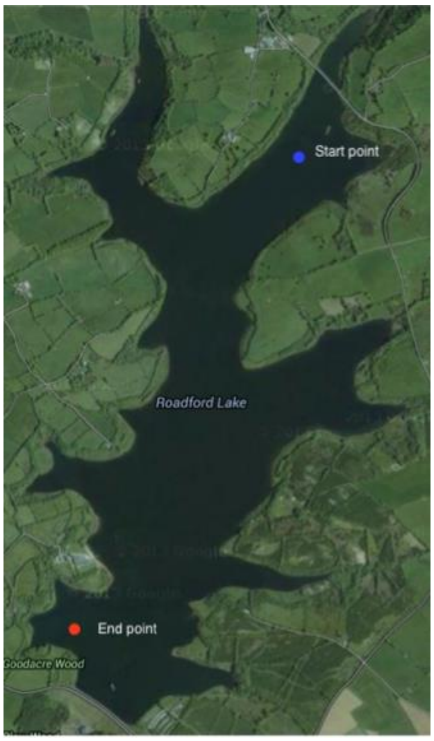
- The velocity of the vehicle at the start and end points are assumed to be zero, which satisfies the constraints of the IPS introduced in Section 3.3.
- When tracking the trajectory, the velocity of the USV is set at 1.5 m/s when the vehicle is more than 20 m away from a waypoint and is reduced to 1 m/s when the vehicle is within 20m of a waypoint.
- The distance value is calculated and measured in Euclidean space.
- The environment for both the simulations and experiment are assumed to be the same. Therefore, the dynamic model of *Springer* USV obtained in Section 4 can be applied for both verifications.

5.1. SIMULATION IN A PRACTICAL ENVIRONMENT: Results of running the proposed algorithm in the representation of a practical environment are now presented to show the performances of different path smoothers. The selected environment is Roadford Lake, shown in Figure 11(a) with the start and end points marked as blue and red dots respectively. Using the image processing illustrated in Section 3.1, the extracted environment map is first transformed into a binary map, which is further rasterised into a grid map with dimensions of 100×350 grid cells as shown in Figure 11(b).

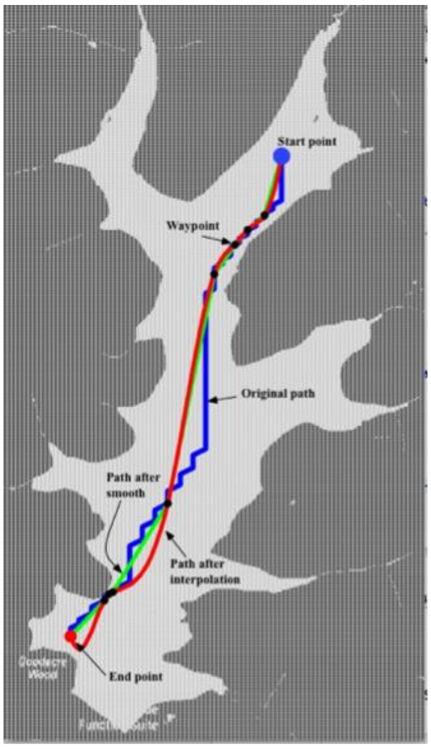
Figure 11(b) shows the effect of the application of the path smoothers to the original path generated by the conventional A* algorithm. The line in blue is the path generated by the conventional A* algorithm. It can be seen that a number of zig zags are formed, making such a path unsuitable for practical navigation. The line in green is the trajectory obtained by applying the LOPS and WRPS. Note that within the WRPS, the optimal control distance (d_{cont}) is defined as 50 m to align with the dynamics of *Springer*. It can be observed that the jags have been markedly reduced, producing a path with a reduced number of turns. The red line shows the more continuous path obtained after application of the IPS to the path generated after LOPS and WRPS application. These results show that the smoothed A* path planning algorithm can achieve a smooth and continuous path successfully for a practical environment and the next step would be integration into *Springer* for field trials.

Table I shows the computational data results of the simulation. It can be seen that the computational time for the conventional A* algorithm is less than that for the smoothed A* algorithm (100 ms for conventional A* “compared to 485 ms for smoothed A*). However, although the smoothing process increases computational cost it reduces distance cost. The distance of the path generated by the conventional A* algorithm is 2380 m compared to 2301 m provided by the smoothed A* algorithm. In addition, the number of course changes of the path has be reduced significantly from 39 to 9 by using the smoothed A* algorithm.

To confirm the reliability and consistency of the verification process additional simulations with different start and end points were carried out. In line with the evaluation of the initial simulation results, the total distance cost and computational time of the two algorithms have been compared with the results shown in Table II. It is evident that the smoothed A* algorithm consistently provides a path with lower distance cost. Although the intrinsic property of the conventional A* algorithm is to find the shortest path, the total distance cost can be reduced further by using the smoothed A* algorithm as the new path seeks to avoid unnecessary turnings. It can be seen that for each of the simulations, the computational time for the smoothed A* algorithm is longer than that for the conventional A* algorithm, as the path smoothing procedure consumes more time to generate a smooth path. However, Table II indicates that the modified path planning algorithm is still fast enough to satisfy the sampling time constraint (less than 1 second) indicating that the algorithm should be suited to real-time applications.



(a)



(b)

Figure 11. Off-line path planning simulation result. (a) The satellite map (2.13×3.93 km) of Roadford Lake with marked start and end points. (b) The binary grid map (100×350 grid cells, 1 grid cell column length = 21.3 m and row length = 11.2m) with trajectories generated using different PSs of the smoothed A* algorithm, i.e. original trajectory without path smoothing, namely the conventional A* algorithm result (blue line), trajectory after applying the LOPS and WRPS, and smoothed trajectory after using the IPS (red line)

Table I. Properties comparisons of conventional and smoothed A*

Performance	Conventional A*	Smoothed A*
Time (ms)	100	485
Distance Cost (m)	2380	2301
No. of turns	39	9

Table II. Simulation results with different start and end points

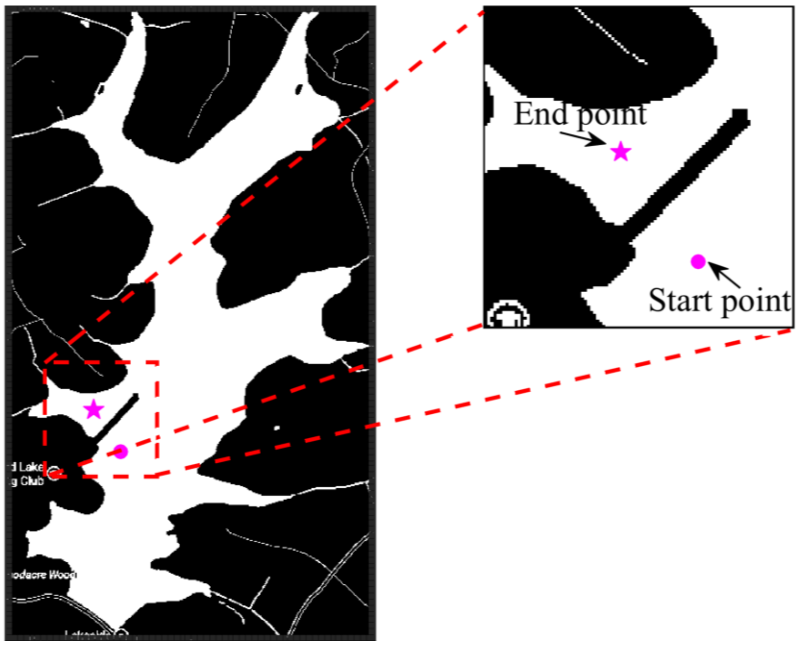
Test	Start point	End point	Total distance cost (m)		Computation time (ms)	
	(m, m)	(m, m)	A*	Smoothed A*	A*	Smoothed A*
1	(639, 896)	(1491, 2016)	257.00	254.48	56	236
2	(319, 2464)	(1491, 2016)	155.40	152.31	26	137
3	(1278, 1120)	(1278, 2464)	184.50	182.99	19	150
4	(639, 2800)	(1278, 1120)	349.95	345.95	77	392

5.2. SIMULATIONS USING THE DYNAMICS OF A PRACTICAL USV: To further test the algorithm simulations have been undertaken to demonstrate the seamless integration of the smoothed A* algorithm with the NGC system of *Springer* USV. The dynamic model and the turning radius of the vehicle have been specifically applied to test the degree of accuracy with which the vehicle is able to follow the generated path.

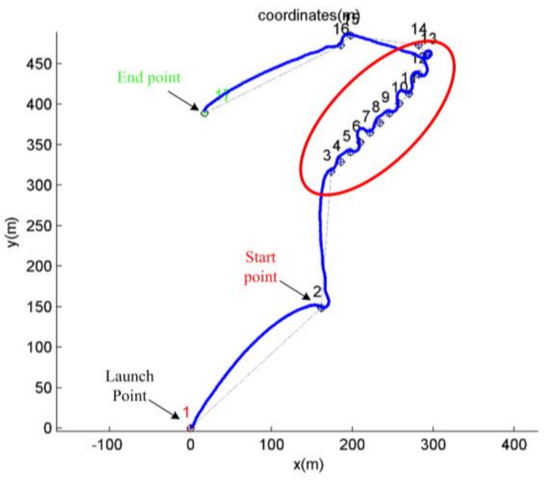
In the simulations, the PID controller in the autopilot is used to track the waypoints. In addition, to increase the complexity of the simulation, a virtual obstacle has been added to the environment (shown in Figure 12(a)) such that the collision avoidance capability can be tested. The USV launch point is (-4.2368°, 50.6954°) in longitude and latitude; whereas, the start and end points transmitted to the path planning algorithm are (-4.2344°, 50.6967°) and (-4.2364°, 50.6988°). The reason for such a configuration is that when conducting the experiments, the USV is normally launched from the pier and starts the mission in the middle of lake. Therefore, the USV launch point and mission start point are not the same. Another simulation condition is that as the adopted PID controller is designed to follow a series of waypoints instead of a continuous trajectory, the IPS is therefore abandoned with only the LOPS and WRPS being used as path smoothers.

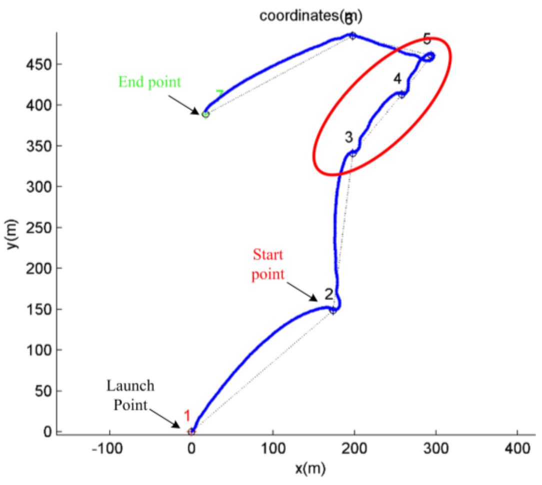
To compare the performances with different path smoothers, simulations have been carried out by applying the smoothed A* algorithm with and without WRPS. The results are presented in Figure 12(b) (without WRPS) and 12(c) (with WRPS) showing the generated trajectories as well as the associated tracking performance of *Springer*. The generated waypoints are depicted as crosses in circles and connected by black dashed lines while the blue lines are the trajectories taken by the vehicle when following these waypoints.

Comparing the results of the two tests, it can be observed that the blue trajectory in Figure 12(c) is much smoother than the one in Figure 12(b), especially in the section denoted by the red boundary. This is because, through the use of the WRPS, redundant waypoints can be removed producing a path that the vehicle's autopilot is better able to follow. It should be noted that such behaviour is especially important in terms of maintaining safety. Although the planned path is guaranteed to be obstacle free, when the USV is navigating, any deviation from the path may still put the vehicle at risk. These simulation results of the two tracking trajectories are used as the benchmark for the experiments presented in the next subsection.



(a)

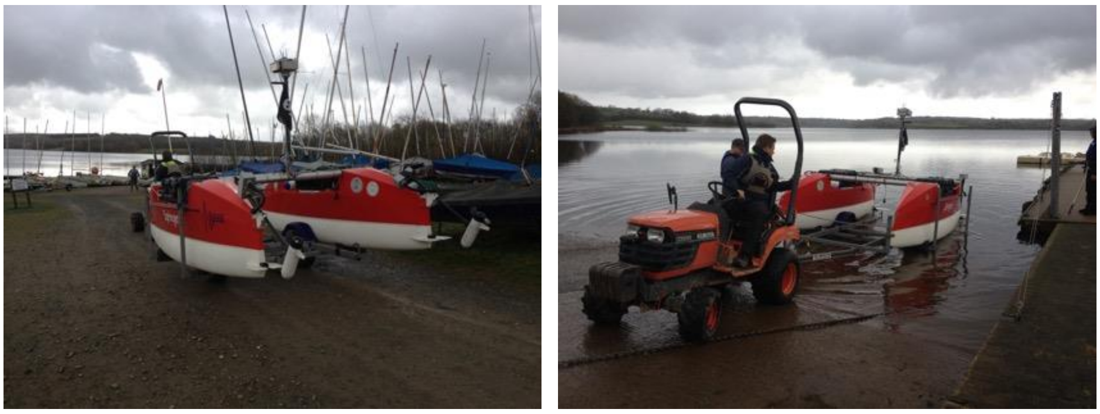




(b) (c)

Figure 12. *Springer* off-line simulated path tracking result using PID controller. (a) The configuration searching space with one barrier virtual obstacle. The grids (100×350 grid cells, 1 grid cell column length = 21.3 m and row length = 11.2m) are represented in grey colour. (b) Generated waypoints and simulated tracking trajectory result without applying the WRPS. (c) Generated waypoints and simulated tracking trajectory result considering USV dynamic model by applying the WRPS. The simulated tracking trajectories are represented in blue lines. The planned paths are shown in black lines.

5.3. EXPERIMENTS ON A PRACTICAL USV: The aim of these field trials is to validate the practical performance of the smoothed A* algorithm and the implementation of a complete NGC system of *Springer*. The experiment was conducted off-line, in that the trajectory was generated before the vehicle started the mission. The locations of *Springer*'s launch point and path planning start and end points are the same as in Section 5.2. Hence, for the off-line path planning, the waypoints generated before launching the USV are the same as in Section 5.2. During the field trials, two GPS and three compasses are used in the Navigation Data Acquisition subsystem to obtain the navigational information. To improve the accuracy of the navigational data, Kalman filtering technology is applied to fuse the recorded data. This implementation work was done by (Motwani et al., 2016). *Springer* used the PID controller to adjust the heading such that each waypoint can be targeted. Figure 13 shows *Springer* on land and being launched from the jetty.



(a) (b)

Figure 13. Different status of *Springer* during the experiment. (a) *Springer* on land. (b) *Springer* is launching.

The tracking performance of *Springer* is shown in Figure 14. The red and blue lines represent the tracking results of the waypoints generated with and without considering WRPS, respectively. By examining the two trajectories, it can be seen that *Springer* was able to reach the end point without colliding with any obstacles. Tracking performance that mirrored that of the simulations was realised from the field trials. The frequency and severity of course changes are reduced along the red line when compared with the blue one, which clearly indicates that the inclusion of the WRPS in the smoothed A* algorithm can better optimise the number of waypoints producing a path that is easier for the vehicle to follow. In addition, the success of *Springer* autonomously reaching the end point without any human intervention demonstrates that the developed path planning algorithm can be integrated into *Springer*'s NGC system, which improves the autonomy level of the vehicle.

Table IV provides further comparison of the results from the simulations and field tests detailed in Figures 12(b), 12(c) and 14 from three aspects: the maximum and the minimum offsets between the simulated and experimental tracking trajectories and the tracking time from launching *Springer* to the USV reaching the end point of the transit. From these comparisons, it can be seen that the effect of WRPS, even though “subject to the vehicle's dynamic constraints, reduces the maximum offset between the simulated and experimental results from 44 m to 27 m. In addition, using the smoothed A* algorithm with WRPS the tracking time taken by the USV is shortened from 93 s to 35 s. Reduced transit time can also indicate that less energy should be consumed by the vehicle, which could be another feature of the smoothed A* algorithm. These results prove that the smoothed A* algorithm can be applied to practical applications.

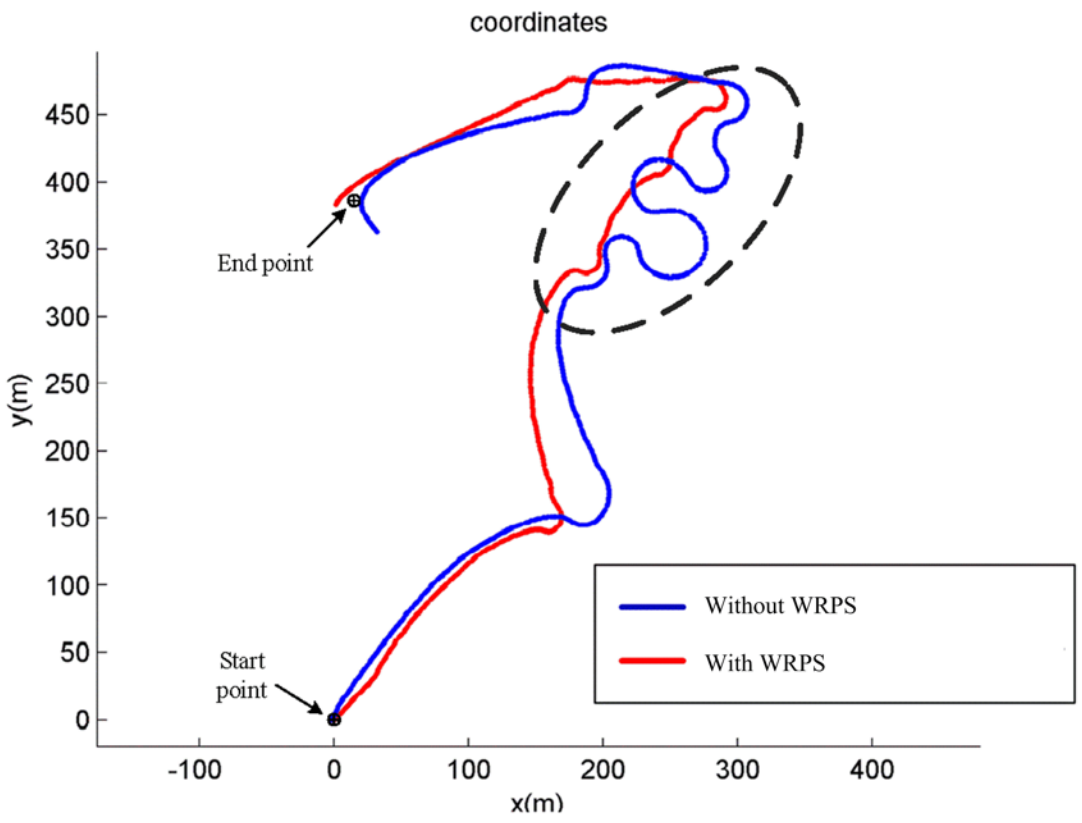


Figure 14. Off-line experiments of tracking the planned path. The blue line is the tracking result of the path generated without using the WRPS. The red line is tracking result of the path generated using the WRPS. The configuration searching space, start and end points are the same as in Section 5.2.

Table IV. Comparisons between simulation and experiment results

	Smoothed A* without WRPS	Smoothed A* with WRPS
max_Offset (m)	44	27
min_Offset (m)	22	15
Time cost (s)	93	35

6. CONCLUSION: A smoothed A* path planning algorithm for autonomous USV NGC system has been developed, verified, and validated by both simulation and field trials. The algorithm is designed with the aim of enhancing the feasibility of path planning with collision avoidance in a real environment. The structure of the algorithm can be divided into three processes. These are, searching space construction, collision free path generation and path smoothing. An image processing method was applied to build the grid map of the searching environment and distinguish the obstacle and free spaces. To improve the confidence of the results and computational time, a 4-cell geometry connection A* algorithm was implemented as the main algorithm. Due to the nature of grid searching limitations, three path smoothers were designed to improve the path continuity. Results showed that the smoothed A* algorithm outperforms the conventional A* algorithm in terms of turning and distance cost.

The main contribution of this paper is the determination that the smoothed A* algorithm can be successfully integrated with the NGC system of a practical USV. The algorithm imported the navigational data from on-board sensors to determine the position of the USV. The waypoints calculated by the algorithm were sent to the autopilot to control the USV's cruise. Both the simulation and experimental results showed that by considering the USV’s dynamic model, the smoothed A* algorithm can be integrated and applied for real applications with better performance compared with the conventional method.

As regards future work, three main issues have not yet been investigated in this paper.

First, the quality and reliability of the navigational data as well as the effect of a complete loss of navigational data. This may cause the path planning to generate trajectories that have increased collision risk with obstacles while the USV is tracking. Second, the battery monitoring will be added as the constraints of path planning for better energy management. Third, the hydrodynamic forces from environmental influences, such as waves, currents and tides can affect the tracking performance especially in ocean environment. Therefore, to further improve the reliability of generated route, the hydrodynamic forces should be considered from path planning level.

ACKNOWLEDGEMENTS

This work is supported by the ACCeSS group. The Atlantic Centre for the innovative design and Control of Small Ships (ACCeSS) is an ONR-NNRNE programme with Grant no. N0014-03-0160, the group consists of universities and industry partners conducting small ships related researches. The authors are also grateful for Autonomous Marine Systems Research Group, Plymouth University and Professor Robert Sutton for providing the dynamic model of *Springer* USV and helping undertake the experiments. The authors are also indebted to Mr. Konrad Yearwood for his valuable critique of this paper.

REFERENCE

Annamalai, A. S., Motwani, A., Sutton, R., Yang, C., Sharma, S., & Culverhouse, P. (2013). Integrated navigation and control system for an uninhabited surface vehicle based on interval Kalman filtering and model predictive control. *In Control and Automation 2013: Uniting Problems and Solutions, IET Conference*, 1-6.

Boh´acs, G., Gyimesi, A., and R´ozsa, Z. Development of an intelligent path planning method for materials handling machinery at construction sites. *Periodica Polytechnica. Transportation Engineering*, 44(1):13, 2016.

Casalino, G., Turetta, A., & Simetti, E. (2009, May). A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields. In *Oceans 2009-Europe* (pp. 1-8). IEEE.

Chiang, C., Chiang, P., Fei, J., and Liu, J. A comparative study of implementing fast marching method and a* search for mobile robot path planning in grid environment: Effect of map resolution. In *2007 IEEE Workshop on Advanced Robotics and Its Social Impacts*, pages 1–6. IEEE, 2007.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010), 'Path Planning for Autonomous Vehicles in Unknown Semi- structured Environments', *The International Journal of Robotics Research,* 29 (5), 485-501.

Fossen, T. I. *Marine control systems: guidance, navigation and control of ships, rigs and underwater vehicles*. 2002.

Goldberg, A. and Harrelson, C. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165. Society for Industrial and Applied Mathematics, 2005.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1972), 'Corrections to "A formal basis for the heuristic determination of minimum cost paths"', *ACM SIGART Bulletin,* (37), 28-29.

Kala, R., Shukla, A., and Tiwari, R. Fusion of probabilistic a* algorithm and fuzzy inference system for robotic path planning. *Artificial Intelligence Review*, 33(4): 307–327, 2010.

Kim, H., Park, B., & Myung, H. (2013). Curvature path planning with high resolution graph for unmanned surface vehicle. In *Robot Intelligence Technology and Applications 2012* (pp. 147-154). Springer, Berlin, Heidelberg.

Liu W., Liu Y., Song R. and Bucknall R. (2015). Towards the development of an autonomous navigation system for unmanned vessels. *Proceedings of International Navigation Conference.* February 24th-26th 2015, Manchester Conference Centre/UK

Miklic, D., Bogdan, S., Fierro, R., & Song, Y. (2012). A grid-based approach to formation reconfiguration for a class of robots with non-holonomic constraints. *European journal of control*, *18*(2), 162-181.

Motwani, A., Sharma, S. K., Sutton, R., & Culverhouse, P. (2013). Interval Kalman filtering in navigation system design for an uninhabited surface vehicle. *Journal of Navigation*, 66(05), 639-652.

Motwani, A. (2015). Interval Kalman Filtering Techniques for Unmanned Surface Vehicle Navigation. *PhD thesis*, Plymouth University.

Motwani, A., Liu, W., Sharma, S., Sutton, R., & Bucknall, R. (2016). An interval

Kalman filter–based fuzzy multi-sensor fusion approach for fault-tolerant heading estimation of an autonomous surface vehicle. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, *230*(3), 491-507.

Naeem, W., Sutton, R., & Chudley, J. (2006, June). Soft computing design of a linear quadratic Gaussian controller for an unmanned surface vehicle. In *Control and Automation, 2006. MED'06. 14th Mediterranean Conference* on (pp. 1-6). IEEE.

Naeem, W., Sutton, R., & Chudley, J. (2006). Modelling and control of an unmanned surface vehicle for environmental monitoring. In *UKACC International Control Conference*, August, Glasgow, Scotland.

Naeem, W., Irwin, G. W., & Yang, A. (2012). COLREGs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics*, 22(6), 669-678.

Naeem, W., Sutton, R., & Xu, T. (2012). An integrated multi-sensor data fusion algorithm and autopilot implementation in an uninhabited surface craft. *Ocean Engineering*, 39, 43-52.

Oriolo, G., Vendittelli, M., and Ulivi, G. On-line map building and navigation for autonomous mobile robots. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 3, pages 2900–2906. IEEE, 1995.

Otsu, N. (1975). A threshold selection method from gray-level histograms.Automatica, 11(285-296), 23-27.

Sharma, S. K., and Sutton, R. (2013). A genetic algorithm based nonlinear guidance and control system for an uninhabited surface vehicle. *Journal of Marine Engineering & Technology*, 12(2), 29-40.

Smierzchalski, R. (1999). Evolutionary trajectory planning of ships in navigation traffic areas. *Journal of marine science and technology*, 4(1), 1-6.

Tam, C., & Bucknall, R. (2010). Path-planning algorithm for ships in close-range encounters. *Journal of marine science and technology*, 15(4), 395-407.

Tam, C., & Bucknall, R. (2013). Cooperative path planning algorithm for marine surface vessels. *Ocean Engineering*, 57, 25-33.

Tsou, M. C., & Hsueh, C. K. (2010). The study of ship collision avoidance route planning by ant colony algorithm. *Journal of Marine Science and Technology*, 18(5), 746-756.

Wang, Y., Yu, X., & Liang, X. (2018). Design and implementation of global path planning system for unmanned surface vehicle among multiple task points. arXiv preprint arXiv:1807.08106.

Xu, T., Chudley, J., & Sutton, R. (2006). A fuzzy logic based multi-sensor navigation system for an unmanned surface vehicle. In *Proceedings of the UKACC International Control Conference*, Glasgow, UK.

Xu, T. (2007). An intelligent navigation system for an unmanned surface vehicle. *PhD thesis*, Plymouth University, UK

Xue, Y., Clelland, D., Lee, B. S., & Han, D. (2011). Automatic simulation of ship navigation. *Ocean Engineering*, 38(17), 2290-2305.

Yang, J. M., Tseng, C. M., & Tseng, P. S. (2015). Path planning on satellite images for unmanned surface vehicles. *International Journal of Naval Architecture and Ocean Engineering*, 7(1), 87-99.

Zhu, M., Wang, Y., & Wen, Y. (2013). A Global Path Planning Algorithm of Unmanned Vessel in Inland Waterway. *In ICTIS 2013: Improving Multimodal Transportation Systems-Information, Safety, and Integration*, 2106-2113.