



Int. J. Nav. Archit. Ocean Eng. (2015) 7:87~99

<http://dx.doi.org/10.1515/ijnaoe-2015-0007>

pISSN: 2092-6782, eISSN: 2092-6790

Path planning on satellite images for unmanned surface vehicles

Joe-Ming Yang¹, Chien-Ming Tseng² and P.S. Tseng¹

¹*Department of Systems and Naval Mechatronic Engineering,
National Cheng Kung University Tainan, Taiwan*

²*Department of Electrical Engineering and Computer Science,
Masdar Institute, Abu Dhabi, UAE.*

ABSTRACT: *In recent years, the development of autonomous surface vehicles has been a field of increasing research interest. There are two major areas in this field: control theory and path planning. This study focuses on path planning, and two objectives are discussed: path planning for Unmanned Surface Vehicles (USVs) and implementation of path planning in a real map. In this paper, satellite thermal images are converted into binary images which are used as the maps for the Finite Angle A* algorithm (FAA*), an advanced A* algorithm that is used to determine safer and sub-optimal paths for USVs. To plan a collision-free path, the algorithm proposed in this article considers the dimensions of surface vehicles. Furthermore, the turning ability of a surface vehicle is also considered, and a constraint condition is introduced to improve the quality of the path planning algorithm, which makes the traveled path smoother. This study also shows a path planning experiment performed on a real satellite thermal image, and the path planning results can be used by an USV.*

KEY WORDS: A* algorithm; Collision avoidance; Image analysis.

INTRODUCTION

Unmanned vehicles are able to complete a variety of tasks based on their equipped modules. However, before these vehicles begin, guiding them through cluttered environments is the first challenge. Therefore, planning a collision-free path and designing a navigation system for USVs are our goals for this study.

Path planning is integral and crucial in various fields including robotics and video game designing (Choset, 2005; Murphy, 2000; Patel, 2000). The classic problem of determining the shortest path in a cluttered environment has been one of the main objectives for many research efforts over the years. Dijkstra (Dijkstra, 1959) proposed an algorithm to address the shortest path problem. The Dijkstra algorithm can ultimately calculate the optimal path, but it searches for nodes in every direction, which makes this algorithm inefficient and impractical to use. Hart (Hart et al., 1968) utilized the heuristic function to estimate the cost of the path from the initial vertex to the goal vertex. The Dijkstra algorithm is combined with this heuristic function to form a new node-searching strategy known as the A* algorithm. The heuristic function can increase the efficiency of the Dijkstra algorithm by pruning the search space in maps.

Choset demonstrated that A* algorithm is optimal with respect to the chosen branching factor in a gridded map (Choset,

Corresponding author: Chien-Ming Tseng, e-mail: ctseng@masdar.ac.ae

This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

2005). The A* algorithm utilized in gridded maps uses a branching factor of 4 or 8, thus its potential headings are artificially constrained to multiples of 45°. As a result, the paths found by A* are not truly the shortest. Yap (Yap, 2002) proposed the concept to decompose terrains into a set of hexagonal grids. His research indicated that a better result was acquired, yet his method is rarely used because most extant digital maps are composed of square grids. Some researchers introduced post-procedures to address the constraints of potential headings. Botea (Botea et al., 2004) adopted a post-smoothed method called A* Post-Smoothed (A*PS) to eliminate redundant turns in the path found by A* algorithm. Despite how the potential headings of resulting paths found by A*PS are not limited to the branching factor, the resulting paths are still not the true shortest paths (Daniel et al., 2010). Daniel (Daniel et al., 2010) proposed that the Theta* algorithm would conquer the limitations of potential headings. The main concept of Theta* and A* is the same, but an additional procedure is executed in Theta*: It considers whether there exists a line-of-sight between the vertex and the parent of the vertex's parent. If this relationship exists, the parent of the vertex will be replaced by the grandparent of the vertex. The procedure can make potential headings of these paths unconstrained. However, Theta* is also not guaranteed to calculate the true shortest path, especially when obstacles are too close to one another.

USVs have been developed for many years (Manley, 2008). In 2006, Woods Hole Oceanographic Institution utilized USVs to map the terrain of the seafloor (Denny et al., 2006). Furthermore, USVs were used to inspect littoral structures in the aftermath of Hurricanes Wilma (2005, Florida) and Ike (2008, Texas) (Steimle et al., 2009). In order to navigate these USVs autonomously, control theories are applied on them. Some developed USVs are autonomously guided by PID controls. Caccia (Caccia et al., 2008) applied PID controls on a USV named Charlie, and their PID controls could perform basic control tasks such as auto-heading, auto-speeding and straight line following. The dynamic control of Charlie was improved by using a nonlinear Lyapunov-based control law in 2009 (Bibuli et al., 2009). Naeem (Naeem, 2011) also utilized PID controls to make USVs adhere to the generated seaway.

In this study, the objectives are focused on the path planning algorithm, which will be applied to the USV in the future.

METHODOLOGY

Gridded map

Continuous terrain has to be discretized before executing a path planning algorithm. In this investigation, color satellite images and thermal satellite images are utilized as the gridded maps of FAA*. These satellite images including thermal type and color type are bought from National Space Organization (NSO) in Taiwan. The resolution is 2 meters per pixel length. In simulation part, the gridded map is like an array, which stores only two values, namely 1 and 0. The grid which has zero value means that the grid is an obstacle. In the RESULT part, the thermal satellite image will be used instead of the simulation maps, in fact, the thermal map is also an array, and each cell in the array stores the value of heat. The detail of utilizing thermal map will be described in the RESULT part.

A* (A-star) algorithm

A* algorithm is widely used in technological industries such as robotics and video game designing (Choset, 2005; Murphy, 2000; Patel, 2000). This algorithm is optimized with respect to the chosen branching factor within a gridded map. Fig. 1 portrays a scheme that used a branching factor of 8.

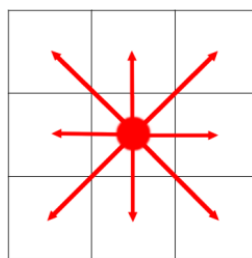


Fig. 1 Branching factor 8 scheme.

Once the start and goal vertices have been determined, A* algorithm generates visible neighbors of the start vertex based on the branching factor. Three variables listed below are calculated and stored in every vertex that is generated by A*. The procedure of A* algorithm is briefly described below where the details of A* algorithm can be acquired from the references (Choset, 2005; Daniel et al., 2010):

- 1) The G value $g(s)$ is the length of optimal path from the start vertex to vertex s , and every G value of the vertices may be updated before A* terminates.
- 2) The H value $h(s)$ is the heuristic value determined by the heuristic function. This value is an estimate of the distance from vertex s to the goal vertex. In this study, the straight line distance from vertex s to the goal vertex is used.
- 3) The Parent value $parent(s)$ is utilized to track the start vertex from the goal vertex after A* is completed. Every vertex has only one parent.

There are two global data structures used in A* algorithm, namely the open list and the closed list.

The open list stores the vertices that A* considers for expansion. The G value of the vertex in the open list is updated when the new G value is less than the old one. The detail procedure is described in later section.

The closed list is used to store vertices which have been expanded, and it ensures that every vertex has been expanded at most once.

The vertex chosen to undergo expansion is based on the criterion that the vertex has the minimal F value in the open list. The evaluation of the F value in A* is depicted in the following equation:

$$F = G + H \quad (1)$$

A* on grids, A* with post-smoothed method and finite angle A*

There are many versions of A* algorithm, different versions are developed to cope with different problems. For example, in video games like the Pacman, the Pacman can only move in 4 directions (up, right, left and down), so the A* algorithm should only plan a path within these 4 direction. In higher level game, the characters in the game can move 8 directions (branching factor 8) shown in Fig. 1, therefore, an 8 directions of A* algorithm is developed.

The basic A* algorithm is called A* on Grids, this is usually used in video games or gridded map. The A* on Grids algorithm is relatively easy to use than that of other versions. The branching factor of A* on Grids is 8 (shown in Fig. 1), which means that the potential headings of A* on Grids are constrained to multiples of 45 degrees. To be more specific, the A* on Grids algorithm can only plan a path that the path is composed of turning angle of multiples of 45 degrees (Patrick 2005). Consequently, there are unnecessary headings in the desired path.

The Post-Smoothed Method is proposed to smoothen the path found by A* on Grids. This method smoothen the redundant headings by examining the line-of-sight. Fig. 2 displays the results of A*PS where A*PS did shorten the path found by A* on Grids. However, A*PS is also not guaranteed to determine the true shortest path because it uses the resulting path from A* on Grids. It is possible that the A*PS can also finds the optimal path in Fig. 2, however, in the later section, we will prove that the FAA* is better than A*PS by averaging the results of 500 random maps in different randomness condition (10%, 20%, 30% random obstacles).

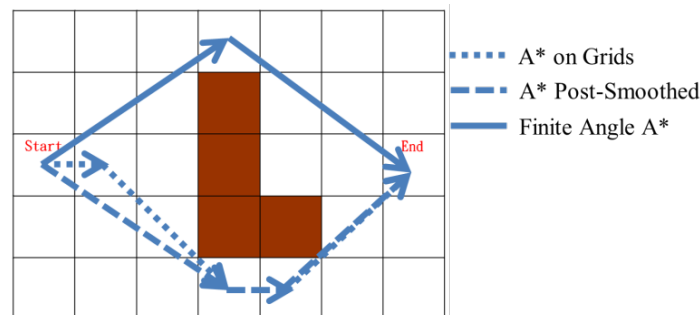


Fig. 2 The resulting paths of A* on Grids, A*PS and FAA*.

Finite-Angle A* algorithm is an advanced A* algorithm proposed to help plan a safer and shorter path. In FAA*, the branching factor can be decided by users and depends on the complexity of the maps. Finite Angle A* (FAA*) algorithm is introduced to tackle the shortcomings of A* on Grids and A*PS. FAA* can improve the resulting path of A* on Grids by increasing the value of the branching factor shown in Fig. 4, but increasing this factor will result in high computation time. Hence, a list of branching factors is created to eliminate redundant expansions. This list ensures that every potential heading can only correspond to just one path length when expanding the vertices within an acceptable time.

Branching factor list

A list of branching factors is generated to increase the efficiency of FAA* by eliminating redundant expansions. The potential headings and their corresponding paths are stored in this list. Fig. 3 displays the scheme of branching factor 16. The potential headings in the first quadrant are symmetrical with respect to the origin, x-axis or y-axis. Therefore, only the potential headings in the first quadrant are stored. Table 1 shows the stored lengths which correspond to these potential headings. To conserve some computation time, the squares of the lengths of the corresponding paths are used.

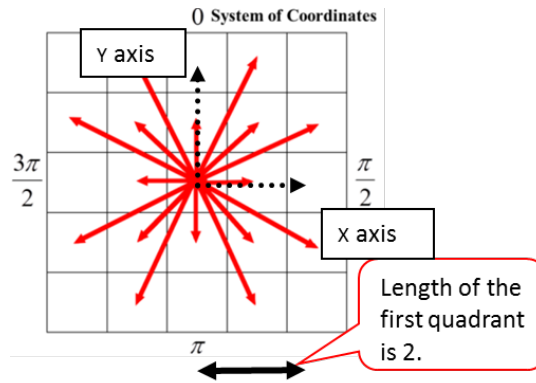


Fig. 3 Grids with branching factor 16.

Table 1 The list of corresponding lengths for branching factor 16.

Length	1	5	2	5	1
Angle	0	$\pi/8$	$\pi/4$	$3\pi/8$	$\pi/2$

In Fig. 3, the matrix of the branching factor has a size of 5×5 . The expanding lengths in the up, down, left and right directions are 2 (length from origin to the boundary is 2, shown in Fig. 3) in the figure, and thus the matrix is expressed as *ExpandSize(2)* in this study. *ExpandSize* can be used to express the potential headings in grids without much difficulty. This expression is utilized in the following procedure instead of the branching factor values. The procedure of algorithm FAA* is shown in Appendix.

Operation of finite angle A* (FAA*)

Before describing the FAA*, a concept of safe distance is introduced at first. The safe distance is the distance from the path (s to s') to the obstacle, where the center of the obstacle is used as the grid center in Fig. 4. In this section, the values of the safe distance should be greater than $\sqrt{2}/2$ units in length. Considering the condition that is shown in Fig. 4, if the values of the safe distance are not greater than $\sqrt{2}/2$ units in length, it will result in a valid path between vertex s and vertex s' , and this condition is impractical in the real environment. Hence, the lengths of the safe distance utilized in this section are greater than $\sqrt{2}/2$ grid units. Figs. 5~7 display examples of the operations of FAA* with *ExpandSize(2)*. The G and H values are labeled on the vertices that were generated. The arrows point towards to the parent as the safe distance is greater than $\sqrt{2}/2$ units in length. Detailed recordings of the operations are described below.

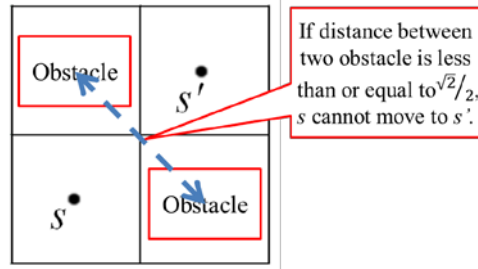


Fig. 4 s should not move to s' , so if the values of the safe distance are greater than $\sqrt{2}/2$ units.

Expansion 1: First, FAA* expands vertex *Start* (B1) based on the list of branching factors shown in Fig. 5. In this expansion, vertices A1, A2, A3, B2, C1, C2 and C3 are generated. However, vertex C2 is eliminated later because it is an obstacle. In addition, vertex C3 is also eliminated because the line-of-sight is invalid between vertex B1 and vertex C3 as proven in Fig. 6. The line-of-sight (shot-dot line in Fig. 6) between vertex B1 and vertex C3 does not exist because the distance between the center of grid C2 and the straight line B1C2 is less than $\sqrt{2}/2$ units in length. The vertices A1, A2, A3, B2 and C1 are successfully generated and are all added to the open list (full lines that point to B1 in Fig. 6). Once the expansion of vertex *Start* is complete, vertex *Start* moves from the open list to the closed list. Afterwards, the vertex with the minimal F value is chosen to be expanded later, that is, vertex B2 is selected to be the next one that undergoes expansion.

	1	2	3	4
A	G=1.00 H=3.61	G=1.41 H=2.83	G=2.24 H=2.24	
B	Start	G=1.00 H=2.24		
C	G=1.00 H=3.00			Goal

Fig. 5 Vertex expansion of FAA* in Step 1.

	1	2	3	4
A	G=1.00 H=3.61	G=1.41 H=2.83	G=2.24 H=2.24	
B	Start	G=1.00 H=2.24		
C	G=1.00 H=3.00			Goal

..... Blocked Path

Fig. 6 Line of Sight (B1,C3) does not exist.

Expansion 2: FAA* expands vertex B2 based on the list of branching factors. Vertices A4, B3 and C4 are visited for first time, so they are added to the open list, and their parent is vertex B2 displayed in Figure 7. The vertices A1, A2 and A3 are also generated and examined in this expansion. Furthermore, it is not required to update the G values as well as the parents of the vertices A1, A2 and A3 because the new G values of these generated vertices are all greater than the original G values. For example, the vertex A3 has a new G value of 2.41 when B2 is expanded. The resulting new G value is greater than the old G value of 2.24, and hence, the G value of vertex A3 does not need to be updated. The line-of-sights (short-dot line in Fig. 7) do not exist between vertex B2 and vertex C3 or between vertex B2 and vertex C1. After the expansion of vertex B2 is completed, vertex C4 is selected to be expanded next and is added to the closed list.

	1	2	3	4
A	G=1.00 H=3.61	G=1.41 H=2.83	G=2.24 H=2.24	G=3.24 H=2.00
B	Start	G=1.00 H=2.24	G=2.00 H=1.41	
C	G=1.00 H=3.00			G=3.24 H=0.00

..... Blocked Path

Fig. 7 Results of the second expansion.

Expansion 3: Since the goal vertex C4 is added to the closed list, FAA* terminates. The optimal path is then obtained by extracting the parent from vertex C4. The resulting path is $C4 \rightarrow B2 \rightarrow B1$.

FAA*, A*PS and A* on grids

Details of these three path planning algorithms are discussed in reference to Tseng's research (Tseng et al., 2012), Table 2 shows the path length and run time of FAA*, A*PS and A* on Grids in 100×100 gridded maps with different, random obstacles. The path length and run time are averaged over 500 maps. In these tables, the experimental results show that FAA* can determine a shorter path than both A* on Grids and A*PS, though in the higher *ExpandSize* it cost more time to calculate the paths. However, FAA* with high *ExpandSize* can be used to determine a much more shorter path before the USV begins its mission, higher *ExpandSize* can be utilized as an off-line method. On the other hand, lower *ExpandSize* can be used as a real time path planning algorithm (on-line method). In fact the A* algorithm can determine near optimal solution and that is why different versions of A* algorithm did not improve much further (Daniel et al., 2010). Different versions of A* algorithm are widely used in the autonomous cars in the famous DARPA challenge in USA. Many universities including Cornell university, University of Michigan and Carnegie Mellon University etc... utilize different versions of A* algorithm which are particularly developed for the cars (Dmitri et al., 2010). Like these universities did, the FAA* is developed for unmanned surface vehicle in this study. A*PS uses the result of A* on Grids, so constraint condition (will be described in later section) cannot be applied, since the branching factor is only 8 in A* on Grids. FAA* is more flexible to be implemented because *ExpandSize* is also flexible. In this study, FAA* with *ExpandSize*(10) is used as an off-line path planning for the following experiments.

Table 2 Runtime and path length of three different algorithms in different random obstacles environment.

Random grid map	Method	Run time (second)	Path length (unit grid length)	Path length increased (%)
10%	A*	0.033	120.809	5.28
	A*PS	0.034	115.921	1.02
	exp3	0.031	115.220	0.41
	exp5	0.090	115.131	0.33
	exp10	0.434	114.753	0.00
20%	A*	0.048	128.352	9.51
	A*PS	0.049	122.031	4.12
	exp3	0.047	119.301	1.79
	exp5	0.120	117.813	0.52
	exp10	0.620	117.203	0.00
30%	A*	0.057	137.836	14.77
	A*PS	0.058	131.462	9.05
	exp3	0.057	124.329	2.64
	exp5	0.134	122.633	1.12
	exp10	0.800	121.387	0.00

Line-of-sight

To make FAA* more practical, a modified definition of line-of-sight is proposed to let FAA* determine the safer paths. Many studies concentrate on solving the true shortest path problems. The true shortest path is composed with a set of lines tangent to the obstacles on the map (Daniel et al., 2010; Lozano-Perez and Wesley, 2000), which means that this path is not collision-free and is impractical to use. To steer USVs autonomously on the planning path will require navigation systems, and

errors in these navigation systems cannot be avoided completely. In addition, the dimensions of the vehicles have to be considered as well as the accuracy of the civilian GPS system used in our USV. Thus, collision-free paths are much more useful than the true shortest paths.

Line-of-sight is defined as a straight line that does not encounter any obstacles from vertex s to vertex s' , and it can be written otherwise as $LineofSight(s,s')$. In the previous section, it is mentioned that A* algorithm selects a node to expand and then generates visible neighbors for that selected node. These visible neighbors are determined by examining the line-of-sight between themselves and that selected node. For instance, if $LineofSight(s,s')$ exists and is identified, vertex s' is then designated as a visible neighbor of vertex s .

In this investigation, a modified definition of line-of-sight is proposed so that FAA* is able calculate a safer path. The new definition of line-of-sight requires that none of the obstacles appear in the safety area. This safety area is identified as a rectangle shape illustrated in Fig. 8 (dotted line). This rectangular region is determined by a safe distance D and safe distance L . Distance D usually denotes half of the ship's width plus a safe margin, and distance L represents half of the ship length along with a safe margin. We define that $LineofSight(s,s')$ exists if there are no obstacles in the safety area.

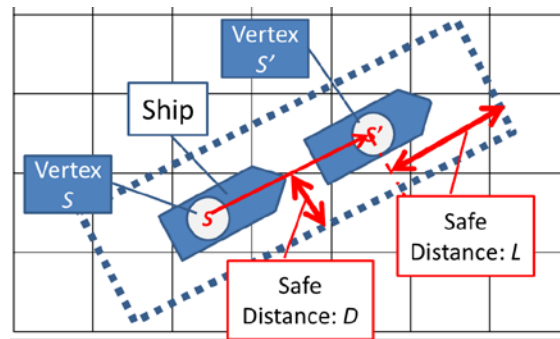


Fig. 8 The definition of safety area.

Fig. 9 shows how the safe margin and ship dimensions are related. In this case, a ship with a size of 8 meters long and 4 meters wide is used. Fig. 9(a) displays the result of FAA*, which does not consider the dimensions of the ship. For this scenario, FAA* did plan an optimal path, but it is not useful in real world, because the ship and the obstacle will collide. Fig. 9(b) illustrates the path that considered a ship's dimensions, but it is still not pragmatic, because a potential error in the ship's navigation system will still result in the ship colliding with the obstacle. Fig. 9(c) shows a safer path, where a width margin of 6 meters is used. A safer path is planned not only by considering a ship's dimensions but also a safe margin, and that is how our line-of-sight works. The results show that our definition of line-of-sight is useful for collision-free path planning. The map in Fig. 9 is 100×100 in grids, where the resolution is $1\text{ m} \times 1\text{ m}$.

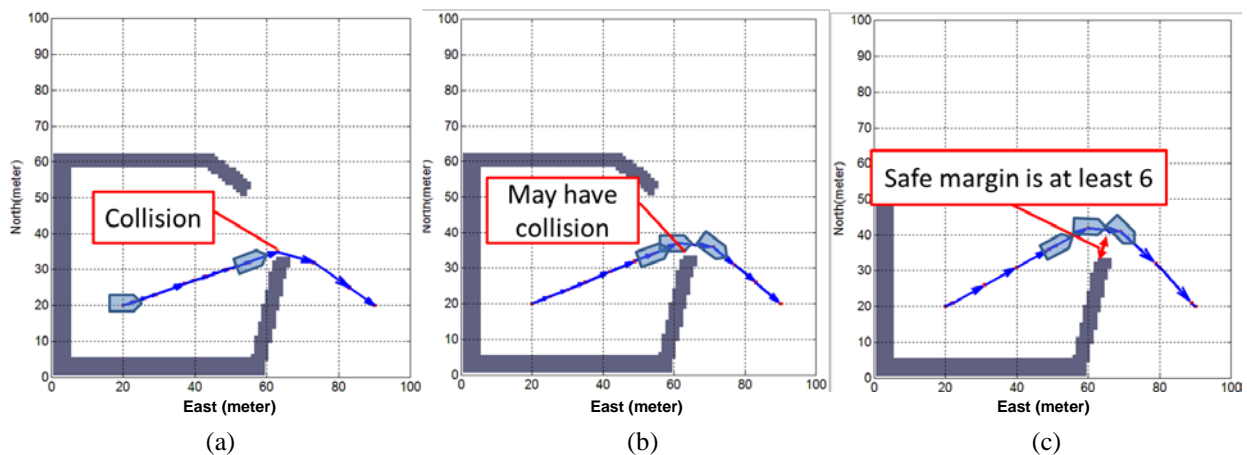


Fig. 9 (a) Path without considering the ship's dimensions; (b) Path considering the ship's dimensions but ignoring the safe margin; (c) Path considering the ship's dimensions and safe margin.

Constraint condition of initial status of ships

When a ship's dimensions are taken into account, the ship's initial heading also has to be considered. If the ship's initial heading is ignored, the planned path cannot be pragmatic. In Fig. 10(a), the FAA* calculates an optimal path without considering the ship's initial heading; therefore, the resulting path is impossible for many ships to follow. Furthermore, sharp angles in the path, which are angles between two consecutive line segments, are also removed to make the path smoother, and the constraint equation is depicted in Eq. (2). Because the ship is static at the beginning, the turning angle of the first movement should also be constrained. In this study, the constraint angle is set to 45 degrees, and the initial admissible angle is set to 18 degrees. Given an expanded vertex $N_i = (nx_i, ny_i)$, a parent of the vertex $P_i(N_i) = (px_i, py_i)$ and a successor of the vertex $S_i(N_i) = (sx_i, sy_i)$, two quantities are defined: (i) $\Delta P_i N_i = N_i - P_i$, the velocity vector from a vertex's parent to the vertex; (ii) $\Delta N_i S_i = S_i - N_i$, the velocity vector from a vertex to its successors, where $i \in \text{open list}$. With the constraint condition, successors are determined by line-of-sights as well as turning angles, and the turning angle ϕ_i is defined as:

$$\phi_i = \cos^{-1} \frac{\Delta P_i N_i^T \Delta N_i S_i}{|\Delta P_i N_i| |\Delta N_i S_i|}, \quad \forall S_i, \phi_i < \pi/4 \quad (2)$$

Figs. 10(b) and (c) display the resulting path of different initial ship headings. In this case, the vehicle length is 4 meters with the margin length of 0.5 meters, and the vehicle width is 2 meters with the margin width of 0.5 meters. Hence, distance D is 1.5 meters, and distance L is 2.5 meters. The vehicle's original position is (6,4), and target is at (20,25). In addition, the map size is 30×30 in grids, where the resolution of map is $1 \text{ m} \times 1 \text{ m}$. The resulting path is determined by FAA* with *ExpandSize* (10). Blue arrows in Fig. 10 denote tangential velocities of the waypoints.

FAA* with the constraint condition can plan a more pragmatic and smoother path, however, the constraint condition cannot be implemented in A* on Grids and A*PS. Because A* on Grids can only move in eight directions, it means that the potential headings of A* on Grids are artificially constrained to multiples of 45 degrees, and all these multiples are eliminated due to our constraint condition in Eq. (2). Furthermore, the post-smoothed method cannot be used since the examination of the line-of-sight between one node to another will eliminate the contribution of the constraint condition.

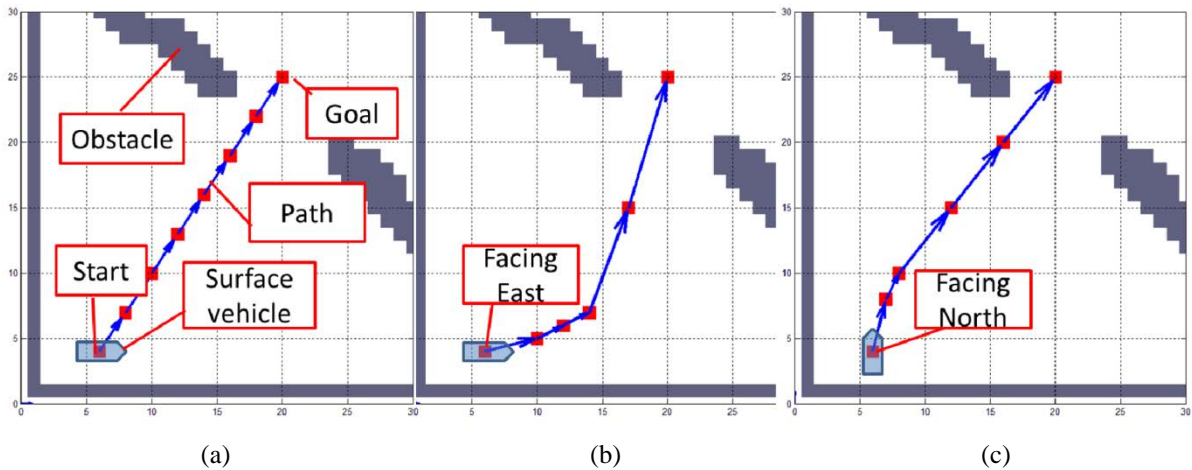


Fig. 10 (a): The resulting path without constraint condition; (b): The resulting path with constraint condition when USV faces east; (c): The resulting path with constraint condition when USV faces north.

RESULT

In this section, all method proposed in previous sections are combined to achieve an effective path planning algorithm for the unmanned surface vehicle.

FAA* with constraint condition and ship dimensions

In the previous section, Finite Angle A* with *ExpandSize(10)* is effective enough to calculate the path. Therefore, FAA* with *ExpandSize(10)* is used for this experiment. All implementations are calculated with MAC book pro mid 2010 with Core 2 Duo 2.4G and 4 GB memory. For these path plans, the dimensions of the surface vehicle are as follows: The length is 10 meters with a margin of 1 meter, and the width is 6 meters with a margin of 5 meters. The map is 100×100 in grids, where the resolution is $1\text{ m} \times 1\text{ m}$. Fig. 11 displays the resulting path when the surface vehicle faces east. The starting point is (16,30), and target is (60,90) and (90,10) for Figs. 11(a) and (b) respectively. In Fig. 12(a) and (b), the resulting paths with the surface vehicle facing north are also displayed. Path lengths and computation times are shown in Table 3.

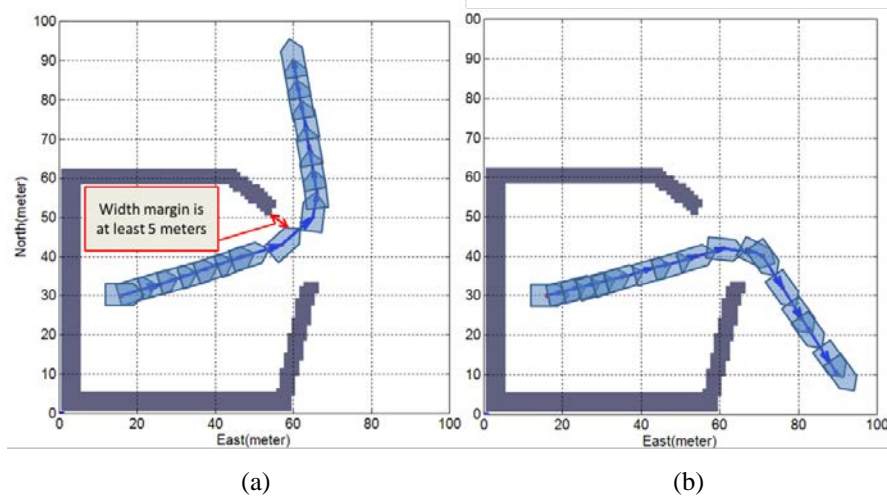


Fig. 11 Using FAA* to determine the path for the USV, where the vehicle faces east.

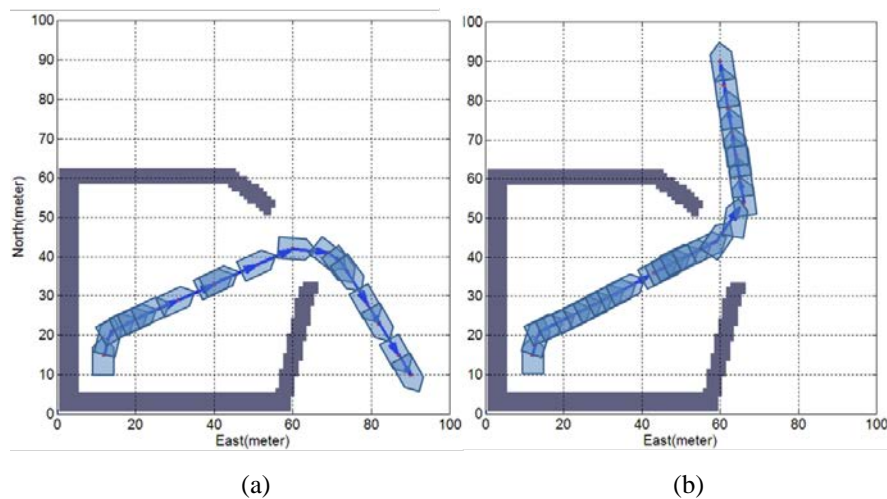


Fig. 12 Using FAA* to determine the path for the USV, where the vehicle faces north.

Table 3 Computation time and path length of four cases.

	Computation time (second)	Path length (meter)
Fig. 12(a)	1.6063	94.2586
Fig. 12(b)	1.8222	92.3903
Fig. 13(a)	1.8823	103.7313
Fig. 13(b)	2.0336	105.6072

Implementation of FAA* on satellite images

In this chapter, the procedure of applying FAA* on satellite thermal images is discussed. There are two beneficial aspects of using satellite images as the maps for USVs: First, there is a complete digital map database of Taiwan, so this can reduce the cost of creating maps. Second, if there are any sudden environmental changes, new paths can be re-planned instantly because satellite images are generally updated every day.

The satellite images used in this study are taken by the satellite named FORMOSAT-2 which is Taiwan's satellite. Generally, the satellite equips with many sensors, like thermal sensor, gravitation sensor and color camera etc.... The satellite images we used are called orthophoto which is an aerial photograph geometrically corrected ("orthorectified") such that the scale is uniform.

These satellite images including thermal type and color type are bought from National Space Organization in Taiwan. The resolution is 2 meters per pixel length. The google map is not suitable for path planning, because the satellite images from google map are deformed. On the other hand, the maps bought from National Space Organization are well calibrated and aligned. The scales of these satellite images are uniform. In this study gridded maps are used in all path planning algorithms. The grids are equal in size, and their centers are used as the vertices. In addition, the gridded maps are not further decomposed since we think that the resolution of 2 meters is quite enough comparing to the error of GPS system. The study area is called Anping harbor, and the approximate GPS location for the area are that latitude: 22.988911° N and longitude: 120.151080°E. The unit is dd.dddd. The date of the image is March 2013.

The satellite image we used is built based on the Taiwan Datum 1997 (TWD97) system which is kind of 2-degree transverse Mercator projection. In order to match the coordinates between the map and the navigation system of the USV, coordinates of the resulting paths under the TWD97 system are converted into World Geodetic System 1984 (WGS84) format (Steve, 2012; Snyder, 1987).

When the coordinates of the satellite thermal image is converted from TWD97 into WGS84, the USV can use the regular GPS system to position itself and then navigate itself autonomously.

The thresholds of satellite thermal images

The objective in this section is to create a map instantly for the USV, so the satellite technology involved is introduced here. The color satellite image is not effective in our study, because the color of oyster rafts is very similar to that of sea water, which is illustrated in Fig. 13(a). Hence, the satellite thermal image, in Fig. 13(b), is utilized to separate oyster rafts from the water region. The water region is used to plan the path for the USV. Two thermal thresholds are used to binarize the satellite thermal images. One threshold is utilized to treat sea water in the harbor, where the sea water in the harbor is slightly higher in temperature than offshore sea water, which usually has the same thermal value as offshore oyster rafts. Another threshold is used to treat the offshore regions. Furthermore, in Fig. 13(b), the land region can be easily distinguished by using the satellite thermal image.

In the satellite thermal image, it shows that the value of oyster rafts is higher than sea water. That is because oyster rafts absorb sun light and release more heat than the sea water, which makes their temperature relatively high and therefore can be detected by a satellite. The binary image of the satellite thermal image is displayed in Fig. 13(c).

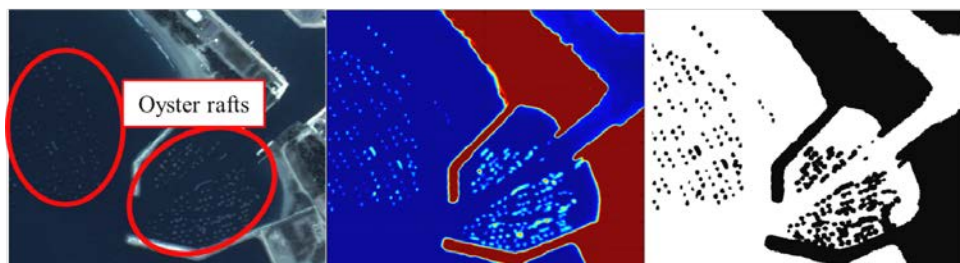


Fig. 13 (a) Color satellite image; (b): Satellite thermal image; (c): Binary image of the satellite thermal image.

A binary image can then be used as a map for the USV. Binary images are composed of only two values, namely "1" which is considered as the water region, and "0" which is considered as the land region (the obstacles). The procedure of converting a satellite thermal image into a binary one is displayed in Fig. 14.

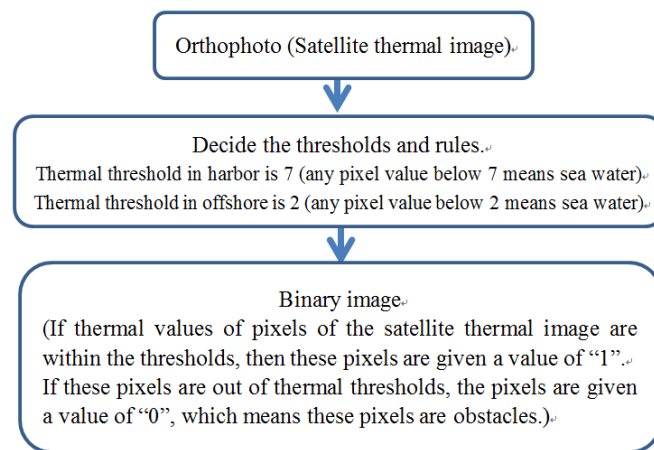


Fig. 14 The procedure of converting satellite thermal images into binary images.

In Fig. 15, an implementation of FAA* on the satellite thermal image is carried out. The FAA* uses *ExpandSize(10)*, and the surface vehicle is 16 meters long and 10 meters width. In addition, the length margin is 1 meter and the width margin is 15 meters, where a high width margin can prevent a collision. Distance D in this case is half of the ship width plus a width safe margin, and hence D is 20 meters. The satellite thermal image in Fig. 15 is in the size of 500×600 pixels, where the resolution is $2\text{ m} \times 2\text{ m/pixel}$. The satellite thermal image in Fig. 16 has a size of 600×900 pixels, where the resolution is $2\text{ m} \times 2\text{ m/pixel}$. In Fig. 16, the satellite thermal image is used, but the resulting paths are printed on a color satellite image for easy interpretation. Run-times and path lengths of all cases are depicted in Table 4.

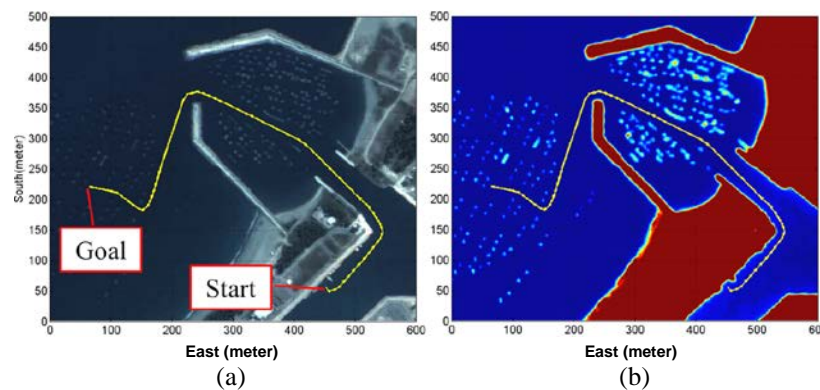


Fig. 15 (a) Resulting path printed on a color satellite image;
(b) Resulting path printed on a satellite thermal image.

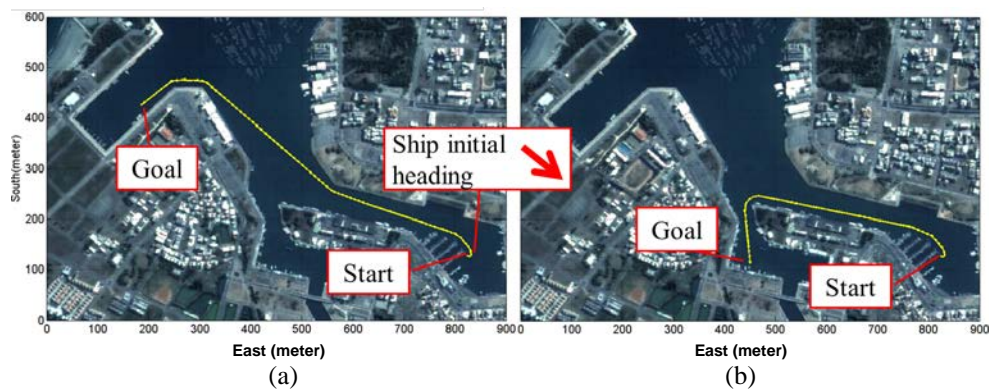


Fig. 16 (a) Resulting path printed on a color satellite image;
(b) Resulting path printed on a color satellite image.

Table 4 Computation time and path length of FAA* on a satellite thermal image.

	Computation time (second)	Path length (meter)
Fig. 16	65.46	1677.90
Fig. 17(a)	78.53	1582.05
Fig. 17(b)	35.71	1112.45

CONCLUSION

In this study, two main objectives are discussed in detail: path planning for unmanned surface vehicles and the methods which improve the performances of the path planning algorithm. The proposed definition of line-of-sight can assist FAA*, A*PS or A* on Grids in calculating a relatively shorter but safer path instead of a path that has the shortest distance and is not collision-free. In addition, the constraint condition is useful to calculate a path when the ship's initial heading is different. According to the constraint condition and the new line-of-sight definition proposed in this study, resulting paths are more pragmatic and can be utilized by a USV directly.

FAA* can be used on binary satellite images. These binary images were converted from satellite thermal images by using thermal thresholds to process the pictures. Oyster rafts can be easily distinguished by using thermal images compared to utilizing color images.

FAA* with the constraint condition and the new line-of-sight definition is therefore a useful method to calculate the sub-optimal path for the surface vehicle.

The FAA* can be used as an on board real time path planning algorithm, the average running time of FAA* with *ExpandSize* 3 is about 0.05 seconds, which means the update rate of path planning algorithm is 20 Hz theoretically. Some field tests are under implemented, to implement the path planning and autonomous navigation, the control theory is also essential.

REFERENCE

- Bibuli, M., Bruzzone, G., Caccia, M. and Lapiere, L., 2009. Path-following algorithms and experiments for an unmanned surface vehicle. *Journal of Field Robotics*, 26, pp.669-688.
- Botea, A., Muller, M. and Schaeffer, J., 2004. Near optimal hierarchical path-finding. *Journal of Game Development*, 1, pp.1-22.
- Caccia, M., Bibuli, M., Bono, R. and Bruzzone, G., 2008. Basic navigation, guidance and control of an unmanned surface vehicle. *Autonomous Robots*, 25, pp.349-365.
- Choset, H., 2005. *Principles of robot motion: theory, algorithms, and implementations*. Massachusetts: MIT Press.
- Daniel, K., Nash, A., Koenig, S., Daniel, K. and Felner, A., 2010. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39, pp.533-579.
- Denny, J.F., O'Brien, T.F., Bergeron, E., Twichell, D.C., Worley, C.R., Danforth, W.W., Andrews, B.A. and Irwin, B.J., 2006. Advances in shallow-water, high-resolution sea-floor mapping; integrating an autonomous surface vessel (ASV) into near shore geophysical studies. *American Geophysical Union Fall Meeting (EOS Trans. AGU)*, San Francisco, Calif, 11-15 December 2006, 87(52).
- Dijkstra, E.W., 1959. A note on two problems in connection with graphs. *Numerische Mathematik*, 1, pp.269-271.
- Dmitri, D., Sebastian, T., Michael, M. and James, D., 2010. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5), pp.485-501.
- Hart, P., Nilsson, N. and Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4, pp.100-107.
- Kevin, M.P., 1998. *Fuzzy control*. Boston: Addison Wesley Longman, Inc.
- Lozano-Perez, T. and Wesley, M., 2000. An algorithm for planning collision-free paths among polyhedral obstacles. *Communication of the ACM*, 22, pp.560-570.

- Murphy, R., 2000. *Introduction to AI robotics*. Massachusetts: MIT Press.
- Manley, J.E., 2008. Unmanned surface vehicles, 15 years of development. *In Proceedings of MTS/IEEE Oceans'08*, Quebec City, Canada, 15 - 18 September 2008, pp.1-4.
- Naeem, W., Irwin, G.W. and Yang, A., 2011. COLREGs-based collision avoidance strategies for unmanned surface vehicle. *Mechatronics*, 22, pp.669-678.
- Patel, A., 2000. *Amit's Game Programming Information*. [online] Available at <<http://theory.stanford.edu/?amitp/GameProgramming/MapRepresentations.html>> [Accessed 3 August 2014].
- Patrick Lester, 2005. *A* path finding for beginners*. [online] Available at <<http://theory.stanford.edu/?amitp/GameProgramming/MapRepresentations.html>> [Accessed 3 August 2014]
- Snyder, J.P., 1987. *Map projections - a working manual*. Available online: Geological Survey (U.S.). [online] Available at <http://pubs.er.usgs.gov/djvu/PP/PP_1395.pdf> [Accessed 3 August 2014]
- Steimle, E., Murphy, R., Lindemuth, M. and Hall, M., 2009. Unmanned marine vehicle use at hurricanes wilma and ike. *In Proceedings of MTS/IEEE Oceans'09*, Biloxi, MS, U.S.A., 6 - 29 October 2009, pp.1-6
- Steve, D., 2012. *Converting UTM to latitude and longitude (or vice versa)*. [online] Available at <http://www.policyalmanac.org/games/aStarTutorial.htm> [Accessed 3 August 2014].
- Tseng, C.M., Fan, C.C. and Yang, J.M., 2012. Collision-free path planning for unmanned surface vehicle by using advanced A* algorithm. *Proceedings of the 26th Asian-Pacific Technical Exchange and Advisory Meeting on Marine Structure*, Fukuoka, Japan, 3 - 6 September 2012, pp.251-256.
- Yap, P., 2002. Grid-based path-finding. *Proceedings of the Canadian Conference on Artificial Intelligence*, 27 - 29 May 2002, pp.44-55.

APPENDIX

```

1  Main()
2      Creat branching factor list (BFList);
3      Set start as current vertex;
4      InsertOpen(parent(start),g(start),h(start));
5      while Open  $\neq$  0
6          Foreach  $\left( \begin{array}{l} \text{angle}(s,s') \in \text{BFList} \\ \text{length}(s,s') \in \text{BFList} \\ \text{LineOfSight}(s,s') \end{array} \right)$  do
7              if  $s' \in \text{Open}$  then
8                  CheckGValue(s,s');
9              else
10                 InsertOpen(parent(s),g(s),h(s))
11              $s = \text{MinimumF}(\text{Open list});$ 
12             if  $s = \text{goal}$  then
13                 return 'Algorithm completed';
14             return 'No path existed';
15         end
16     Void CheckGValue(s,s')
17         if  $(g(s') > g(s) + \text{LineOfSight}(s,s'))$  then
18              $g(s') = g(s) + \text{LineOfSight}(s,s');$ 
19              $\text{Parent}(s') = s;$ 
20     end

```

Pseudocode of finite angle A*