A practical path planning and navigation algorithm for an unmanned surface vehicle using the fast marching algorithm

Yuanchang Liu, Rui Song and Richard Bucknall
Department of Mechanical Engineering, University College London
London, U.K, WC1E 7JE
yuanchang.liu.10@ucl.ac.uk, r.song.11@ucl.ac.uk, r.bucknall@ucl.ac.uk

***Abstract*— There has been an increasing interest in the development of unmanned surface vehicles (USVs) in recent decades. As USVs are required to carry out complex missions without any human intervention in various environments, an intelligent path planning algorithm is critical. A path planning algorithm is able to utilise environment data to calculate an optimal trajectory to guarantee safety. To achieve this, in this paper, a novel path planning and navigation (PN) algorithm is proposed. The PN algorithm uses the fast marching method (FMM) as the base algorithm to search for an optimal collision-free trajectory. Then, to facilitate the trajectory tracking of the USV, a new waypoint-generator based on the line-of-sight (LOS) is developed to generate the optimal number of waypoints from the path. The proposed algorithm has been evaluated based on the *Springer* USV, and has been shown that it can be seamlessly integrated with the *Springer's* exiting autopilot to achieve full autonomy.**

Keywords— unmanned surface vehicle (USV); path planning algorithm; USV dynamics; waypoint tracking

I. INTRODUCTION

Robust and efficient path planning algorithm is essential for the navigation of an unmanned surface vehicle (USV). A number of algorithms have been intensively studied for USV path planning in recent years such as the evolutionary algorithm [1], the grid-based algorithm [2], the artificial potential field (APF) algorithm [3] and the fast marching method (FMM) [4]. Among them, the FMM is able to generate the safest and the smoothest path with guaranteed algorithm completeness and fast computation speed [5]. However, as most USV autopilots are tracking waypoints instead of a specific path, it is desired to generate an optimal number of waypoints from the calculated path. During the generation of waypoints, the tracking capability of the USV should be taken into account to make sure that the vehicle can, to the best of its ability, follow the path.

Hence, in this paper, a novel path planning and navigation (PN) algorithm is proposed. First, the PN uses the FMM to search for an optimal path. Then, a waypoints-generator is developed to extract useful waypoints from the path. The total number of waypoints is cautiously determined such that USV can track effectively. Least number of waypoints is maintained along the straight line to improve the tracking stability; whereas waypoints located on arcs are kept with a certain number to avoid a large heading angle change. To validate the PN algorithm, it has been tested on the *Springer* USV platform. Also, to demonstrate that the algorithm can provide better tracking capabilities than other algorithms, results are analysed and compared against the conventional A* algorithm.

II. THE AUTONOMOUS NAVIGATION SYSTEM

As there is no human operator on board, a USV's autonomous navigation system plays a critical role. Fig.1 depicts the system structure of a typical USV navigation system. It consists of three different modules, i.e. the data acquisition module (DAM), the path planning module (PPM) and the advanced control module (ACM). A USV perceives its surrounding environment using the DAM, which acquires navigation information using a range of different sensors such as GPS, Inertial Measurement Unit (IMU), Automatic Identification System (AIS) and marine radar. Using the data obtained, the algorithm in the PPM determines a safe path for the USV. The safe path includes a set of waypoints, which are used by the controller in the ACM as reference points to safely navigate the USV.

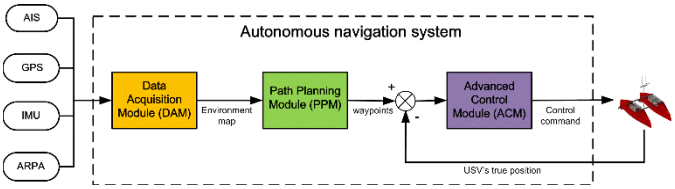


Fig.1. The autonomous navigation system for an unmanned surface vehicle.

Of the three modules, the PPM plays the most important role in the autonomous navigation system. The aim of path planning is to plan an optimised trajectory connecting mission start and end points without colliding with any obstacle *en route*. It should be noted that collision avoidance, especially for USV navigation, is one of the most important requirements since it ensures the safe operation of the USV. In maritime environment, both static obstacles (buoys and rocks) and dynamic obstacles (moving ships) present potential collision risk for the USV. Any path planning algorithm should be capable of intelligently and autonomously avoiding these obstacles by maintaining a safe distance.

978-1-4799-8736-8/15/$31.00 ©2015 IEEE

III. THE FMM BASED PATH PLANNING ALGORITHM

A. The fast marching method (FMM)

The FMM was first proposed by J. Sethian in 1996 to iteratively solve the eikonal equation to simulate the propagation of interface [6]. The eikonal equation has the form of:

$$\left|\nabla T(x)\right| V(x) = 1 \tag{1}$$

where $T(x)$ is the interface arrival time at point x and $V(x)$ is the interface propagating speed. The specific algorithm is described in Algorithm 1. It adopts the fundamentals of Dijikstra algorithm; however, instead of employing the classical rectilinear distance metric, it uses the Eq. (1) to update the cost function. Therefore, compared with Dijisktra, the results provided by the FMM is more continues.

Algorithm 1 *Fast Marching Method* Algorithm

```
assign all the grid points with cost of Infinity                ▷ Initialisation
T(startPoint) ← 0
Far ← all grid points
Known ← all grid points with know cost
for each adjecent point a of Known point do
    Trial ← a
    T(a) = costUpdate(a)
end for
while Trial is not empty do                                     ▷ Update process
    sort Trial
    p ← point with lowest cost in Trial
    remove p from Trial
    Known ← p
    for each neighbour point a of p do
        T(a) = costUpdate(a)
        if a ∈ Far then
            remove a from Far
            Trial ← a
        end if
    end for
end while
return T
```

B. The FMM based path planning

The FMM based path planning algorithm is described in Algorithm 2. Suppose that the planning space (***M***), where the algorithm is performed on, has a representation of a binary map and is perfectly rasterised. The algorithm first reads in the ***M*** and calculates its speed matrix (***V***). The speed matrix (***V***) has is same size as the ***M*** and defines the interface propagation speed at each point in the ***M***. Based on the ***V***, the FMM is executed to calculate an arrival time matrix ***T***, and upon the time matrix ***T***, the optimal path is finally searched by applying the gradient descent method.

Algorithm 2 *FMM Path_Planning* Algorithm

```
Require: planning space (M), start point (p_start), end point(p_end)
Calculate speed matrix V from M
Arrival time matrix (T) ← FMM(V, p_start, p_end)
path ← gradientDescent(T, p_start, p_end)
return path
```

A more intuitive way to interpret the FMM based path planning is from the potential field perspective. In Fig.2(a), two round obstacles are located near the centre of the map; while the start and end points are at northwest and southeast corners respectively. The map is represented by a binary grid map, where each grid in collision free space has value 1 and grids in obstacle areas have value 0.

The FMM is then applied on such a grid to simulate an interface propagation process. The interface is used to help build up a potential field, whose potential value on each grid point is the local interface arrival time. The interface begins to proceed from the start point on the grid map by taking local grid values to determine propagation speed. When the interface reaches the target point, the potential field (shown in Fig.2(b)) is created. In the field, the potential value at each point represents local arrival time of the interface, which subsequently indicates local distance to the start point if a constant speed matrix is used. Since the interface begins propagating from the start point, the potential of the start point is therefore the lowest and is equal to zero. Potential values at other points increase as the interface advances and reach highest value at the end point. Because the interface is not allowed to transmit inside an obstacle area, obstacles' potentials are infinite. Compared with the potential field generated by the APF, the potential field of the FMM has features of global minimum, which avoids local minima problems and increases the completeness of the algorithm. Based on the potential field obtained, the gradient descent method is then applied to find the shortest collision free path by following the gradient of the potential field. The generated path is represented as the red line shown in Fig. 2(c).

C. The fast marching square (FMS)method

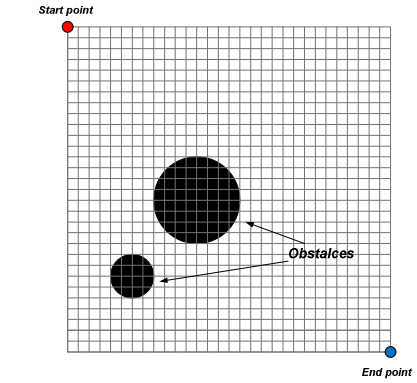
One of the problems associated with path planning by directly using the FMM method is the generated path is too close to obstacles. Such a drawback is especially impractical for USVs, because near distance areas around obstacles (mainly islands and coastlines) are usually shallow water, which is not suitable for marine vehicles to navigate. Hence, it is important to keep the planned path a safe distance away from obstacles.

To tackle this problem, [5] has proposed a new algorithm named the fast marching square (FMS) method. The basic concept behind the FMS is to apply the conventional FMM algorithm twice but with different purposes. The FMS is represented in Algorithm 3. It first generates a safety potential map (***Ms***) by applying the FMM to propagate interfaces from all the points in obstacle area. Based on the ***Ms***, the FMM is executed again from the start point to generate the final path. By using the same previous planning space, the path generated by the FMS is shown as the green trajectory in Fig.2(c) with increased safety.

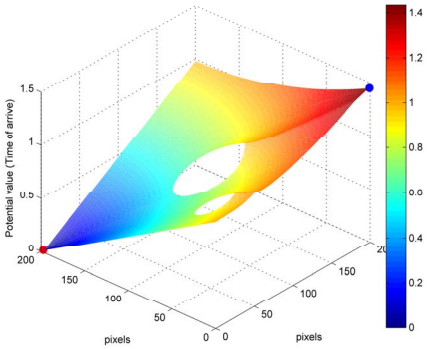
Algorithm 3 *Fast_Marching_Square* Algorithm

Require: planning space (M), start point (p_{start}), end point(p_{end})

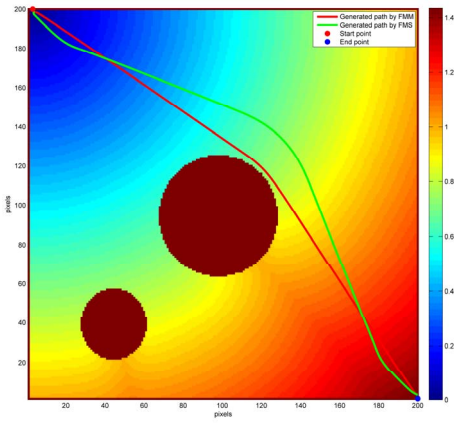
1: **for** each point a in obstacle **do**
2: $obstaclePoints \leftarrow a$
3: **end for**
4: $M_s \leftarrow FMM(obstaclePoints)$
5: $T \leftarrow FMM(M_s, p_{start}, p_{end})$
6: $path \leftarrow gradientDescent(T, p_{start}, p_{end})$
7: **return** $path$



(a)



(b)



(c)

Fig.2. (a) The grid map. (b) Potential field generated by using the FMM. (c) Path generated by using the FMM and the FMS.

IV. THE WAYPOINTS GENERATOR

Trajectory provided by the FMS algorithm has good characteristics of continuity and smoothness. If the autopilot system of a USV is capable of tracking a continuous path, there is no needs to amend it. However, as in this paper the algorithm is specifically designed for the *Springer* USV, which employs an autopilot system tracking instead of a continuous path but a series of waypoints by using the line-of-sight (LOS) theory, it is required to extract informative waypoints from the path to be used as reference points for the tracking operation.

During the generation of waypoints, there is a trade-off between the number of waypoints and USV tracking performance. On one hand, to maintain the continuity feature of the path, it is required to place as many waypoints as possible along the path. On the other hand, intensive waypoints will create a large number of control points, which could possible generate a series of unwanted control outputs further affecting USV tracking performance. However, if there are not enough waypoints placed on the path, especially on the turning arcs (See Fig.3(a)), a large heading angle change will occur during the transition from straight line to the circle arc, and a jump in the desired yaw rate will be experienced, which is difficult for autopilot to cope with.

Hence, in this paper, a novel waypoint-generator algorithm is proposed to obtain useful waypoints from the calculated trajectory. It consists of two main procedures: 1) generate a series of consecutive waypoints retaining the characteristics of the trajectory and 2) a 'waypoints-trimmer' is used to reduce the number of waypoints on straight path while maintaining waypoints located on arcs. For the first step, total of number of desired waypoints can be determined by:

$$n = \frac{L_{trajectory}}{d_{\text{int}erval}} \tag{2}$$

where $L_{trajectory}$ represents the total length of the calculated trajectory and $d_{\text{int}erval}$ is the distance between every two adjacent waypoints. $d_{\text{int}erval}$ can be selected according to the specific dynamics of the USV, i.e. for a USV with high manoeuvrability, $d_{\text{int}erval}$ can be assigned with a relative small value as the USV is able to adjust its autopilot efficiently. However, it should be noted that a larger number of waypoints n is preferred as the larger the number is, the better the trajectory characteristics can be retained especially for non-straight sections.

After the determination of the total number of waypoints, the 'waypoint-trimmer' is employed to refine these waypoints by following the LOS theory. The details of the 'waypoint-trimmer' is presented in Algorithm 4. From the start point, every three waypoints (point A, point B, point C) are selected

in sequence to calculate the turning angle $\Delta\phi$. If point C is not visible to point A, which is expressed as:

$$\Delta\phi < \theta_{\min} \tag{3}$$

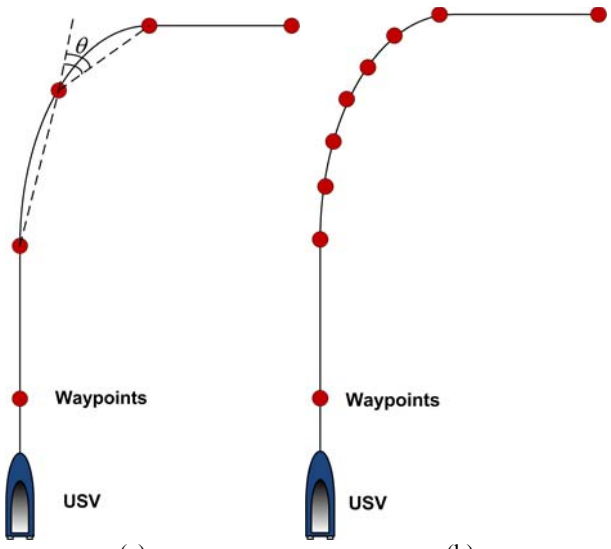
where $\theta_{\min}$ is a predefined parameter having a relatively small value, a 'straight' line can be formed between A and C thereby point B being removed. If point C is visible to point A, an arc is possibly existing between A and C; therefore, the point B needs to be kept to maintain the shape of the trajectory.

Algorithm 4 *Waypoint trimmer* Algorithm

Require: waypoints(Wp), angleMin(θ_{min})

$q_A \leftarrow Wp_{start}$
$q_B \leftarrow q_A.next$
$q_C \leftarrow q_B.next$
while q_C != Wp_{goal} **do**
  $\Delta\phi = \phi_{AB} - \phi_{BC}$
  if $\Delta\phi <= \theta_{min}$ **then**
    $Wp.remove(q_B)$
  else
    $q_A \leftarrow q_B$
  end if
  $q_B \leftarrow q_C$
  $q_C \leftarrow q_C.next$
end while
return Wp

By following the proposed waypoionts-generator, final waypoints can be extracted from the original path shown in Fig.3(b). It can be observed least waypoints are located on the straight line; whereas circle arc has more number of waypoints to facilitate autopilot to track.



(a) (b)

Fig.3. (a) Generated waypoints with less on circle arc. (b) Generated waypoints with more on circle arc.

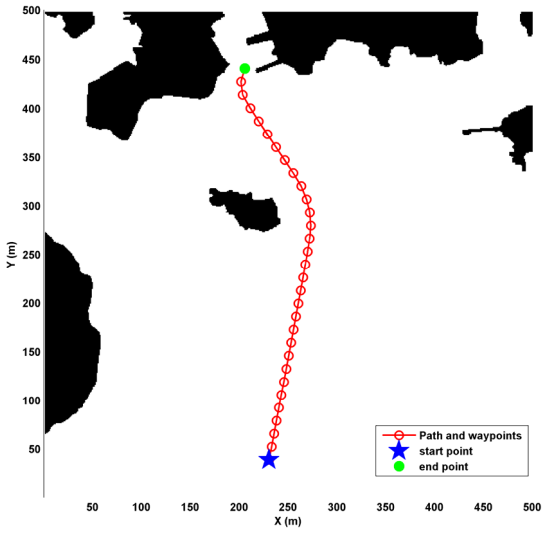
V. Simulations and Results

To test the proposed path planning algorithm, two simulations have been carried out. In the first simulation, the waypoins-generator has been verified in a simulated sea area. Then, the second simulation has been undertaken by using the *Springer* USV model to test the proposed PN algorithm. The algorithm has been coded in Matlab and simulations are run on the computer with a Pentium i7 3.4 GHz processor and 4GB of RAM.

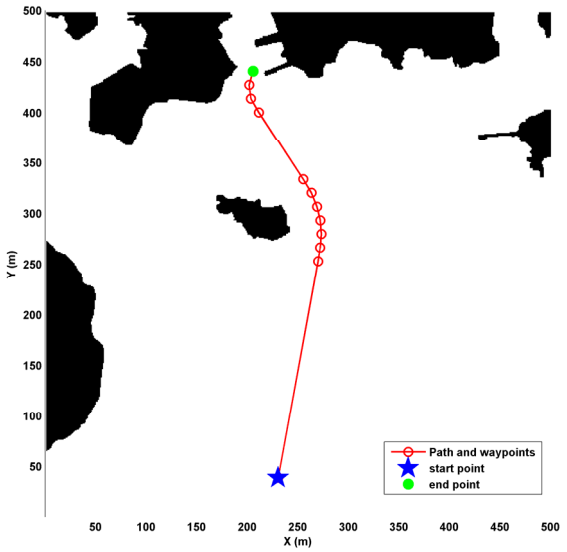
A. Simulation 1

The first simulation validates the proposed waypoints-generator and uses a practical water area, the Plymouth harbour area, as the testing environment. The area has a dimension of 2.5 km * 2.5 km and has been transformed to a 500 pixels * 500 pixels binary map, which could be directly used by the algorithm. The path start and end points have the coordinates of (225,48) and (200,448).

Results are represented in Fig.4. In Fig.4(a). the result after running the first step of the waypoints-generator is displayed. It can be observed that a total number of 31 waypoints are generated and equally placed along the trajectory with the interval distance about 5 pixels, which is equivalent 25 m. Then, the 'waypoint-trimmer' is used to generate optimal number of waypints with the results shown in Fig.4(b). It is now clear that number of waypoints has been reduced to 12 with most of them located in the arc sections to preserve the continuity of the path.



(a)



(b)

Fig.4. (a) Path with equally generated waypoints. (b) Path with optimal number of waypoints.

B. Simulation 2

1) The Springer USV

The *Springer* USV is a catamaran structured vessel developed by University of Plymouth (shown in Fig.5). A watertight twin hull forms the main body of the *Springer* with critical parts such as batteries stored inside. Above two hulls, two peli cases containing the Navigation, Guidance and Control (NGC) system are placed. To obtain navigation information, devices such as GPS are installed on the mast of the *Springer* together with the wireless communication antenna to permit exchange of information. Detailed information of the *Springer* can be found in [7] and [8].

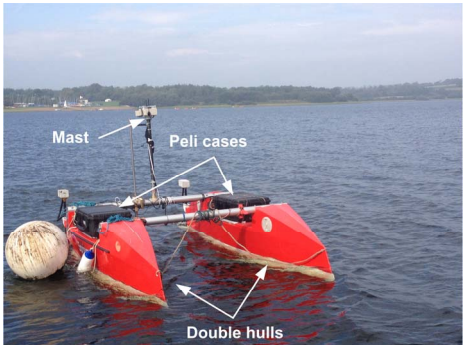


Fig.5. The *Springer* USV.

The *Springer* is steered by a differential mechanism by having different Revolutions Per Minute (RPM) values of two propellers. Therefore, the heading of the vehicle can be controlled by:

$$n_d = \frac{n_1 - n_2}{2} \tag{4}$$

where n_1 and n_2 are two propeller thrusts, and n_d is the differential mode thrust having the maximum value of 1000 rpms. The dynamic model of the *Springer* was calculated from system identification (SI) process using the data obtained from the real trails conducted at Roadford reservoir, Devon, UK. [7] The system dynamics can be expressed in state space form as:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{5}$$

where

$$A = \begin{bmatrix} 1.1130 & 0.3519 & -0.4221 & -0.046 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.0043 & -0.0039 & -0.0016 & 0.0012 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{6}$$

Since the aim of this paper is about path planning algorithm design, the PID controller, which is one of most common and widely used controllers, is implemented for simplicity. The PID controller is designed as:

$$n_d = K_p \tilde{\theta} + K_d \tilde{r} + K_i \int_0^t \tilde{\theta}(\tau) d\tau \tag{7}$$

where $\tilde{\theta}$ is the heading error between current heading and reference heading. $\tilde{r}$ is the yaw rate error. K_p, K_d and K_i are proportional, integral and derivative gain respectively.

2) Simulation results

Simulations have been carried out to test the tracking performance of the *Springer* by following the waypoints extracted from the PN algorithm. To validate and demonstrate the proposed PN algorithm, the A* algorithm, which is the most commonly used approach solving path planning problems, has been used as the benchmarking algorithm in the same environment. Results obtained from two algorithms are compared and analysed.

Simulation environment is represented in Fig.6(a). It is an area with the dimension of 500m * 500m. Two static obstacles are located in the middle together with landmass obstacles at top and bottom parts. The start and end points have the coordinates of (286,184) and (235,402). The speed of the

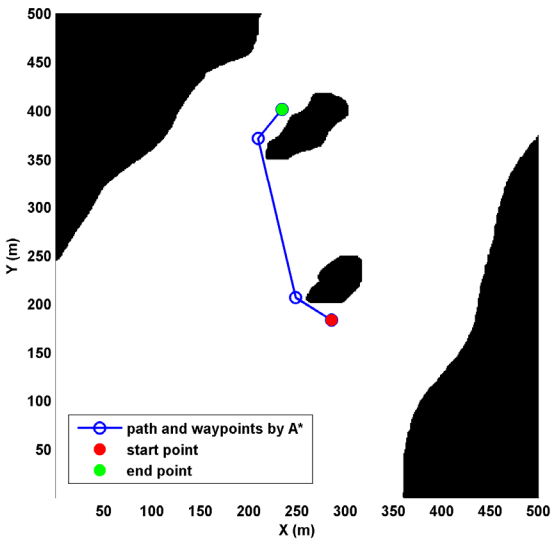
Springer is set to be 1.5m/s subjected to a random small velocity perturbation within the range of [0, 0.1]. In addition, to have a better representation of ocean environment, current with the constant speed of 0.2m/s and constant direction of 90° is added.

The A* algorithm is first applied to generate trajectory. It searches the path based on the least distance cost principle and the generated path is plotted in Fig.6(a) with blue line. It can be observed that the path now consists of four waypoints including start and end points. However, it is evident that the path is not smooth and has large heading angle changes at two internal waypoints.

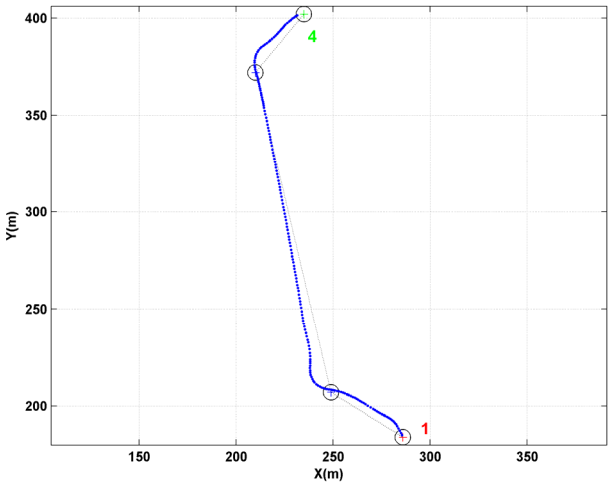
In Fig.6(b), the actual route taken by the *Springer* is represented with an obvious effect brought by the large heading changes. Since the change of the heading exceeds the USV's turning constraint, the *Springer* can only gradually make the turning, which eventually forms a distance offset away from the desired trajectory. In Fig.6(c), the distance offset values for whole operation is recorded and the largest offset is 8m at time step 58.

In comparison, results obtained by using the proposed PN algorithm are shown in Fig.7. In Fig.7(a), the path is smoother than the A*'s and consists of 17 waypoints with most of them located at arc sections to avoid large heading changes.

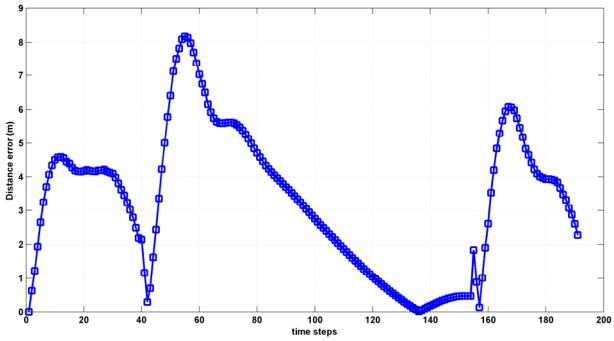
Fig.7(b) shows the tracking results. It is clear that the *Springer* can now well following each waypoint while staying close to the planned trajectory. Fig.7(c) makes a record of the offset value and shows that the largest offset now is only 4m, which is a 50% increase in terms of the safety compared with the A*. It should be noted that such a performance is vital for USV navigation, especially in complex environment with multiple obstacles, where any deviation from the route may cause further collisions.



(a)

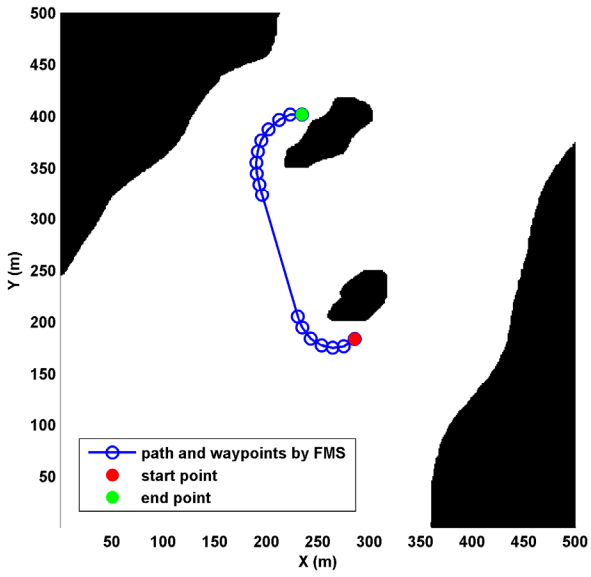


(b)

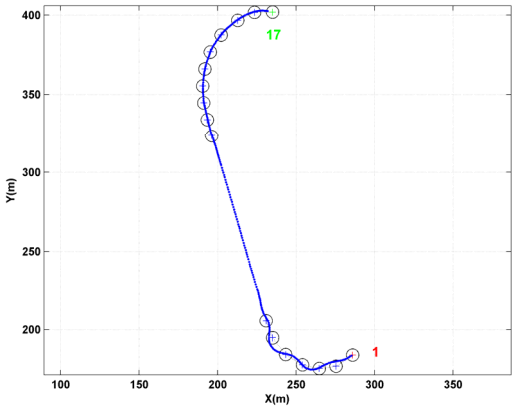


(c)

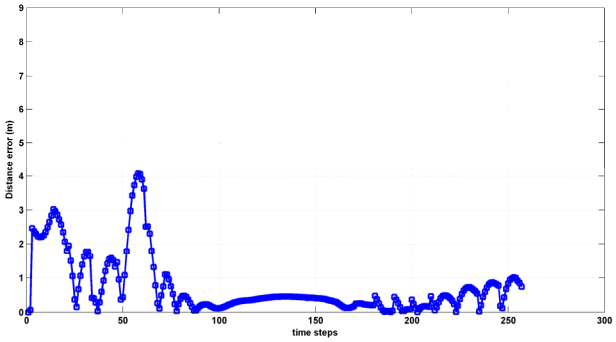
Fig.6. Results generated by using the A*.



(a)



(b)



(c)

Fig.7. Results generated by using the proposed PN algorithm.

VI. CONCLUSIONS

This paper introduced and discussed a novel path planning and navigation algorithm for USV application. The algorithm is able to generate a trajectory with improved smoothness and safety. To integrate with USV's autopilot, a novel waypoints-generator has been proposed. The most efficient number of waypoints can be extracted from the path, which is used as the reference points for tracking. The proposed algorithm has been validated using a real USV platform, the *Springer* USV, in a static environment with multiple obstacles. The results show that the path generated by the algorithm is more convenient for the *Springer* to track as compared with the conventional A*.

ACKNOWLEDGMENT

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- 10-1-0652,the group consists of universities and industry partners conducting small ships related researches. The authors would also to thank Prof Robert Sutton for providing the parameters of the *Springer* USV.

REFERENCES

[1] Szlapczynski R, Szlapczynska J. On evolutionary computing in multi-ship trajectory planning[J]. Applied Intelligence, 2012, 37(2): 155-174.

[2] Naeem W, Irwin G W, Yang A. COLREGs-based collision avoidance strategies for unmanned surface vehicles[J]. Mechatronics, 2012, 22(6): 669-678.

[3] Xue Y, Lee B S, Han D. Automatic collision avoidance of ships[J]. Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment, 2009, 223(1): 33-46.

[4] Liu Y, Bucknall R. Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment[J]. Ocean Engineering, 2015, 97: 126-144.

[5] Gómez J V, Lumbier A, Garrido S, et al. Planning robot formations with fast marching square including uncertainty conditions[J]. Robotics and Autonomous Systems, 2013, 61(2): 137-152.

[6] Sethian J A. A fast marching level set method for monotonically advancing fronts[J]. Proceedings of the National Academy of Sciences, 1996, 93(4): 1591-1595.

[7] Naeem W, Xu T, Sutton R, et al. The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring[J]. Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment, 2008, 222(2): 67-79.

[8] Sharma S K, Naeem W, Sutton R. An autopilot based on a local control network design for an unmanned surface vehicle[J]. Journal of Navigation, 2012, 65(02): 281-301..