Neurocomputing 275 (2018) 1550–1566

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficient multi-task allocation and path planning for unmanned surface vehicle in support of ocean operations

Yuanchang Liu*, Richard Bucknall

Department of Mechanical Engineering, University College London, Torrington Place, London WC1E 7JE, UK

ARTICLE INFO

Article history:
Received 28 November 2016
Revised 2 August 2017
Accepted 28 September 2017
Available online 13 October 2017

Communicated by Shen Jianbing Shen

Keywords:
Unmanned surface vehicle (USV)
Task allocation
Path planning
Self-organising map

ABSTRACT

Presently, there is an increasing interest in the deployment of unmanned surface vehicles (USVs) to support complex ocean operations. In order to carry out these missions in a more efficient way, an intelligent hybrid multi-task allocation and path planning algorithm is required and has been proposed in this paper. In terms of the multi-task allocation, a novel algorithm based upon a self-organising map (SOM) has been designed and developed. The main contribution is that an adaptive artificial repulsive force field has been constructed and integrated into the SOM to achieve collision avoidance capability. The new algorithm is able to fast and effectively generate a sequence for executing multiple tasks in a cluttered maritime environment involving numerous obstacles. After generating an optimised task execution sequence, a path planning algorithm based upon fast marching square (FMS) is utilised to calculate the trajectories. Because of the introduction of a safety parameter, the FMS is able to adaptively adjust the dimensional influence of an obstacle and accordingly generate the paths to ensure the safety of the USV. The algorithms have been verified and evaluated through a number of computer based simulations and has been proven to work effectively in both simulated and practical maritime environments.

1. Introduction

Recently, there is an increasing interest in the research of unmanned surface vehicles (USVs). An extensive number of vehicle platforms are being developed and deployed in numerous practical applications, where the military utilisations are dominant. By sending the vehicles to carry out dangerous missions, such as sea patrol and coastal guarding in hazardous environments, the risk to personnel can be significantly reduced as there is minimal involvement of human operators. Note that using the USVs in civilian applications is also possible and promising. One deployment is environmental monitoring. In highly polluted lakes, USVs can be sent out to effectively and efficiently collect water sampling data without exposing humans to harmful elements. Another potential application is the search and rescue missions in post-disaster scenes. In [15], it has been reported that together with a micro aerial vehicle, a prototype USV was used to survey damage in parts of Marco Island after the Hurricane Wilma struck in 2005. It has been demonstrated that the effective utilisation of USVs can dramatically increase the success rate of the rescue mission by reducing the response time.

By comparing and analysing the aforementioned applications, it was found that missions assigned for USVs are normally complex, i.e. a single mission consists of multiple sub-tasks the USV is required to accomplish subject to constraints. An example of such a case has been shown in Fig. 1, which represents a water sample collection mission. Red dots represent the multiple water monitoring stations, where the critical sampling data is stored. Stations are located in the area of interest, each with equal importance and priority, which requires the USV to autonomously visit these locations with the minimal costs such as the minimum traversal distance or the least energy consumption. Multiple stations can be viewed as multiple goal points and the visiting of these points can be regarded as a problem of multi-goal path planning. It is different from the conventional path planning problem, the aim of which is to generate a collision-free path towards a single goal point instead of multiple ones. Due to the complexity of multi-goal path planning, algorithms are normally developed using a hierarchical structure, which includes a *high-level task allocation* scheme and a *low-level path planning* scheme. A sophisticated task allocation algorithm is first used to calculate an optimal task execution sequence, and then the path planning algorithm is used to generate collision free trajectories visiting each goal point by following the sequence.

When performing the task allocation, the problem can be mathematically summarised as the Travelling Salesman Problem (TSP)

* Corresponding author.
E-mail address: yuanchang.liu.10@ucl.ac.uk (Y. Liu).

https://doi.org/10.1016/j.neucom.2017.09.088
0925-2312/

Fig. 1. A USV is conducting the environmental monitoring mission. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

that is *given a list of cities and the distances between each pair of cities, to search for the shortest possible route that visits each city exactly once and returns to the origin city*. Such a problem is NP-hard and can be solved using both exact and heuristic algorithms. The exact algorithms are able to provide the accurate results but with extensive computational time making them only suitable for the problem in low dimension. The heuristic algorithms sacrifices the accuracy of the searching results by providing near optimal solutions. However, the computational speed is much higher than with deterministic method. Recently, the most commonly used heuristic approaches include the genetic algorithms [22], ant colony algorithms [3] and the neural network algorithms [1,10]. In particular, the self-organising map (SOM) based neural network has become one of the most popular approaches to solve the TSP problem due to its intuitive appeal, relative simplicity and promising performance [11,23].

The fundamental idea behind the application of SOM for TSP is that a circular closed ring of neurons is formed and gradually moves towards the cities through competitive training until each city is assigned a neuron. The training session of the SOM primarily includes two procedures, i.e. the selection of the winner neuron and the update of the winning neuron as well as its neighbourhood points. Specific introduction of the SOM will be given in the latter section. A number of studies have been carried out to improve the performance of SOM for TSP from the aspects of reducing the computation complexity and increasing the efficiency of the algorithm. For example, a dynamic ring structure has been proposed in [1], where the neurons on the ring can be adaptively added or deleted based upon the specific training situation. Similarly, in [17], a neuron inhibition scheme was integrated with the SOM to prevent the neuron from winning too much to produce a more balanced result. Cochrane and Beasley[2] proposed a co-adaptive approach, which involves both the competition and co-operation schemes, to improve the solution quality and the computational time of the SOM. Zhang et al. [23] also introduced a combinational approach including the overall and regional competition rules for the SOM to provide better solution quality as well as maintain the fundamental properties of the TSP problem such as the topology preservation and the convex-hull properties.

However, it should be noted that all of the aforementioned studies only consider an environment with no obstacles, and such a problem belongs to the Euclidean Travelling Salesmen Problem (ETSP). When selecting the winner neuron during the training, the Euclidean metric is adopted to calculate the distance between each neuron and the city, and the neuron with the shortest distance will become the winner. However, Euclidean metric is not feasible when obstacles exist between the neuron and the city. During the updating process, neurons should always be outside obstacles. Therefore, to incorporate the collision avoidance capability into the SOM for TSP, a series of work has been carried out in [4–7]. The main improvement made by these authors is to propose a new winner neuron selection rules. Instead of using the Euclidean metric, an approximation of the shortest collision-free path is calculated for each neuron towards the city and the winner neuron is accordingly selected based upon the length of the path. To increase the efficiency of the algorithm, the approximation has

been developed based upon the convex polygon partition of the area of interest, where the path can be found in the visibility graph using the Dijkstra algorithm. The simulation results show that all the neurons can be kept well away from the obstacles when the SOM is being updated with optimal collision-free paths able to be easily found.

However, there are two main problems associated with these studies. First, as noted by the authors, even though the computational time of the proposed approximation is significantly faster than that of the direct calculation of the shortest path, it is still much slower than using the conventional Euclidean distance. Hence, the proposed algorithm may not be feasible for complex missions, where a large number of goal points need to be visited. Second, the calculated collision-free path stays rather close to the obstacles. This is because the approximation adopts the simple visibility graph approach, which does not respect the distance to the obstacles when generating the path. Such an approach may be suitable for the mobile robots applications but not feasible for USVs. In a typical littoral environments, the dimensions of obstacles (such as the shores) vary with the tides, i.e. rising tide brings more water covering the reef and making the obstacle's area shrink; whereas, during the low tide periods, the effective dimension of obstacles can increase. It is desired that when performing the multi-goal path planning, an adaptive approach can be developed to address these issues, especially the second one.

Therefore, in this paper, a novel multi-task allocation and path planning algorithm has been proposed. For the multi-task allocation, an improved SOM with collision avoidance capability is proposed to fast determine a task execution sequence in an environment with obstacles. The new SOM improves the work of Faigl et al. [6] by using a reactive approach to update the SOM neurons. To maintain the computational efficiency of the SOM, the Euclidean metric is used for winner selection; whereas, the collision avoidance capability has been achieved by using a new neurons updating process. A repulsive force field (RFF) has been generated around each obstacle in the environment, and the SOM neurons are updated not only according to the distance to the city but also the repulsive forces that expels the neurons away from the obstacles. For the path planning, the fast marching method (FMM) has been used as it is able to generate a smooth trajectory within sufficient time.

It should be noted that the utilisation of vector field to assist with SOM has been adopted in some other literature. For example, in [21] and [20], vector field consisting of the attractive and repulsive fields has been incorporated within SOM to achieve an improved performance on multi-model optimisation problem. However, there are two aspects distinguishing the algorithm proposed in this paper and the ones in those literature. First, the aim of using the vector field is different and the way of constructing the vector field is not the same. In [21], because the vector field is used to assist with the updating process of the SOM, the fields have been created around neurons instead of obstacles making the algorithm lack of collision avoidance capability. However, as the aim of this paper lies in the improvement of the navigation safety, the repulsive force field (RFF) has been specifically generated around obstacles such that the collision avoidance requirement can be satisfied. Also, differing from the conventional way of generating the field, i.e. calculating the force at each point in the domain, in this paper, the new way of generating the RFF based upon the FMM [18] has been implemented. By using such an approach, the RFF can be fast created covering the whole area of interest in a single step.

Second, in order to successfully guarantee the safety, instead of being a uniform field as presented in [21] and [20], the field proposed in this paper has an adaptive dimension. The area of RFF can be controlled and varied according to specific requirements by introducing a new controlling parameter called *propagation scale limit*. For example, when a strict demand on the navigation safety presents, the parameter is able to accordingly restrict the propagation process of the FMM and adaptively adjust the dimension of RFF to make sure that sufficient repulsive force can be generated to avoid any collision. The detailed explanation of such a process will be given in Section 2.2.

The rest of the paper is organised as follows. Section 2 specifically introduces the procedures to construct an adaptive RFF. In Section 3, the detailed algorithm structure for multi-task allocation and path planning is described, which includes the improved SOM algorithm and the FMM based path planning algorithm. The proposed algorithms and methods are verified by simulations in Section 4. Section 5 concludes the paper and discusses the future work.

2. Adaptive repulsive force field

Artificial potential field method has been commonly used in robotics motion planning with the primary concept being to construct two different fields that are repulsive and attractive ones to weigh up the influences from obstacles and goal points respectively. The fields are normally generated according to the distance to obstacles or the goal points depending on the type of field to be constructed and the total potential field is the sum of these two fields. Note that the main drawback associated with the potential field method is the local minima problem. To overcome this, in this paper, a new potential field construction approach based on the fast marching method (FMM) has been proposed. As stated in [8], the FMM constructs the field by simulating an electromagnetic wave propagation process, where the wave emanates from the start point and continues to propagate until reaching the end point. The potential value of the field represents the local distance to the start point, and the farther away from the start point, the higher the potential value. Because of this, the generated field will only have one global minima point which is located at the start point with the potential value being 0 and no other local minima points exist in other locations within the area of interest.

2.1. Fast marching method

The FMM was first proposed by Sethian in 1996 to iteratively solve the Eikonal equation to simulate the propagation of an interface [16]. The Eikonal equation has the form as:

$$|\nabla(T(x,y))|V(x,y)=1 \tag{1}$$

where $T(x, y)$ is the interface arrival time at point (x, y) and $V(x, y)$ is the interface propagation speed. Eq. (1) belongs to the partial differential equation (PDE) and its numerical solution can be obtained via the upwind differential method. In Fig. 2, suppose (x, y) is the point at which $T(x, y)$ needs to be solved. The neighbour of (x, y) is a point set containing four elements $(x+\Delta x, y)$, $(x-\Delta x, y)$, $(x, y+\Delta y)$, $(x, y-\Delta y)$. $T(x, y)$ can be obtained as:

$$T_1 = min\left(T_{(x-\Delta x,y)}, T_{(x+\Delta x,y)}\right) \tag{2}$$

$$T_2 = min\left(T_{(x,y-\Delta y)}, T_{(x,y+\Delta y)}\right) \tag{3}$$

$$|\nabla T_{(x,y)}| = \sqrt{\left(\frac{T_{(x,y)}-T_1}{\Delta x}\right)^2 + \left(\frac{T_{(x,y)}-T_2}{\Delta y}\right)^2} \tag{4}$$

$$\left(\frac{T_{(x,y)}-T_1}{\Delta x}\right)^2 + \left(\frac{T_{(x,y)}-T_2}{\Delta y}\right)^2 = \frac{1}{(V_{(x,y)})^2} \tag{5}$$

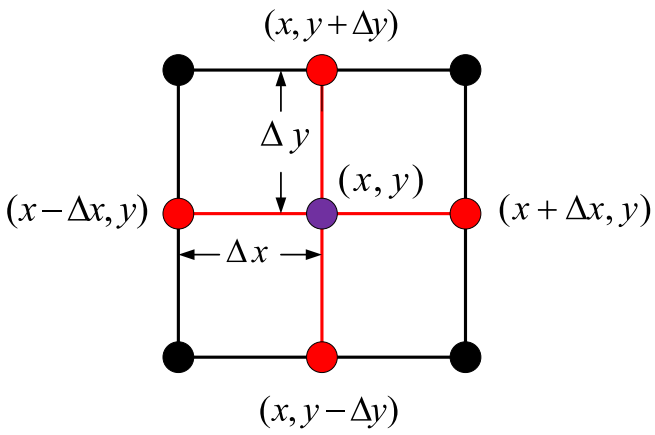

Fig. 2. Grid point (x,y) and its neighbours. The grid map has four connectivity, the grid point therefore has four neighbour points. The grid size is Δx and Δy in the x and y direction respectively.

where Δx and Δy are the grid spacing in the x and y directions. The solution of Eq. (5) is given by

$$T_{(x,y)} = \begin{cases} T_1 + \frac{1}{V_{(x,y)}} & \text{if } T_2 \geq T \geq T_1, \\ T_2 + \frac{1}{V_{(x,y)}} & \text{if } T_1 \geq T \geq T_2, \\ \text{quadratic solution of Eq. (5)} \end{cases} \quad (6)$$

The pseudo-code of the FMM is shown in Algorithm 1. It adopts

Algorithm 1 Fast Marching Method Algorithm.

Require: configuration space (χ), start point (p_{start})
assign all the grid points in χ with the cost of Infinity ▷ Initialisation
$T(p_{start}) \leftarrow 0$
Far $\leftarrow$ all grid points in χ
Known $\leftarrow$ all grid points with known cost
for each adjecent point a of *Known* point **do**
  Trial $\leftarrow a \cup$ *Trial*
  $T(a) = costUpdate(a)$ ▷ Using Eq. (6)
end for
while *Trial* is not empty **do** ▷ Update process
  $p \leftarrow$ point with the lowest cost in *Trial*
  remove p from *Trial*
  Known $\leftarrow p \cup$ *Known*
  for each neighbour point a of p **do**
    $\tilde{T}(a) = costUpdate(a)$ ▷ Using Eq. (6)
    if $\tilde{T}(a) < T(a)$ **then**
      $T(a) \leftarrow \tilde{T}(a)$
    end if
    if $a \in$ *Far* **then**
      remove a from *Far*
      Trial $\leftarrow a \cup$ *Trial*
    end if
  end for
end while
return T

the fundamentals of Dijikstra algorithm by grouping the grid points into three different categories, i.e. the *Far, Known* and *Trial* point sets:

- *Far*: contains grid points with undecided arrival time value (T). In the first time step when running the FMM, all grid points except the start points belong to *Far*.
- *Known*: contains grid points with decided arrival time values (T). Such values will not be changed when the algorithm is executed.
- *Trial*: contains grid points with calculated arrival time values (T); however, values may be changed when the algorithm is running.

However, instead of employing the classical rectilinear distance metric, it uses Eq. (6) to update the cost function, which measures the distance in Euclidean metric. The potential field generated using the FMM can thus be viewed as a distance potential field, where the potential value represents local distance to the propagation start point. A more explicit illustration of such a field has been displayed in Fig. 3. In Fig. 3a, two circular 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 the value 1 and grids in obstacle areas have the value 0. FMM is then applied to 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. The evolution process of interface is shown in Fig. 3b, where the brighter the colour is, the longer the arrival time. When the interface reaches the target point (iteration = 40,000), the final potential field is created.

2.2. The repulsive force field construction process

The example given in Fig. 3b has a single wave propagation start point; it is however also possible to run the FMM from multiple start points. Since the potential field created by the FMM is able to indicate distance information, if multiple waves are emitted from the obstacles, a new potential field reflecting how close a local point is to obstacles can thus be generated, based on which a repulsive force related to the obstacles can be further obtained.

The process of creating the repulsive force field consists of two steps: 1) using the FMM to generate a repulsive potential field implicitly reflecting the risk of obstacles and 2) calculating the gradients of the potential field to get the corresponding force vector field. In the first step, the FMM is run to obtain the repulsive potential field, denoted as D_{rep}, where the potential values represent the distance to obstacles and the longer the distance, the higher the potential. D_{rep} can be calculated using the expression:

$$D_{rep} = FMM(p_{obs}, \alpha) \quad (7)$$

where p_{obs} is the obstacle's location. $FMM(\bullet, \alpha)$ represents the FMM process but with a new parameter denoted as the *propagation scale limit* (α) added. α has been mainly introduced to control the influence area of obstacles in such a way that after running the FMM, any potential value higher than α will be reset to α to reduce the dimension of the potential field. Fig. 4 represents the process of generating a repulsive potential field in a constrained environment. A simulated environment representing a typical maritime environment with two small islands and a channel has been displayed in Fig. 4a. The generated D_{rep} with $\alpha = 0.8$ is represented in Fig. 4b with the contour plot clearly showing the dimension and the distribution of the potential field. When the value of α is reduced to 0.3, as shown in Fig. 4c, the area of the repulsive potential field has been significantly reduced, which demonstrates that the dimension of the D_{rep} can be controlled by adjusting α. Note that the value of α should be adaptively calculated according to a number of the factors such as the safety and the overall distance requirements, which will not be discussed in this paper.

After generating α, the next step is to vectorise the field to generate the repulsive vector field. D_{rep} is vectorised by calculating the

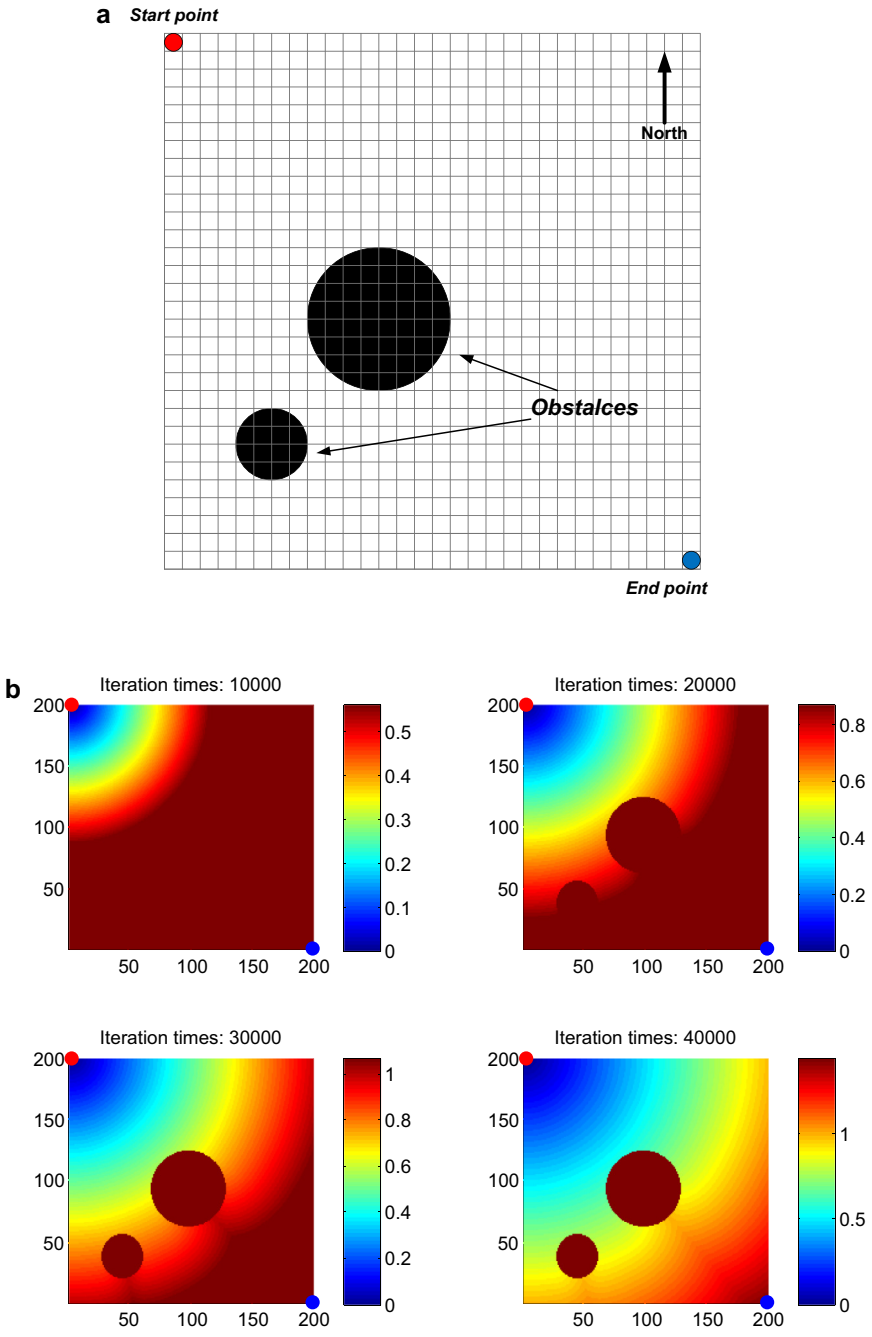

Fig. 3. The illustration of the fast marching method (FMM). (a) Grid map. (b) Simulating interface propagation process by using FM method. Interface starts to emit from (0, 200) and ends at (200, 0). Process are recorded at iteration times 10,000, 20,000, 30,000, 40,000, respectively.

gradients as:

$$\boldsymbol{F} = \nabla(D_{rep}) = \left(\frac{\partial D_{rep}}{\partial x}, \frac{\partial D_{rep}}{\partial y}\right) \tag{8}$$

and the generated $\boldsymbol{F}$ is further normalised such that a unit vector field can be obtained. However, it should be noted that in practice, the influence of the repulsive force should vary according to the distance to obstacles, i.e. the closer to the obstacle, the larger the force. Therefore, $\boldsymbol{F}$ has been re-scaled by referring to the local distance to obstacles as:

$$\boldsymbol{F}_{Rep} = \boldsymbol{F} * (1 - D_{rep}) \tag{9}$$

The obtained $\boldsymbol{F}_{Rep}$ is shown in Fig. 4d, where repulsive force has been clearly shown.

3. Multi-task allocation and path planning based on the improved SOM

The structure of the proposed multi-task allocation and path planning algorithm has been illustrated in Fig. 5. Environmental information, including the dimension of the area of interest as well as obstacles locations, will be used as the primary system input together with specific mission information such as the total number and locations of the tasks. This information will form a synthetic mission map and be fed into the multi-task allocation module, where the task execution sequence is calculated. The mission sequence is then transferred to the path planning algorithm to generate the trajectory for the USV. The path planning is carried out via two different procedures: 1) the off-line path planning

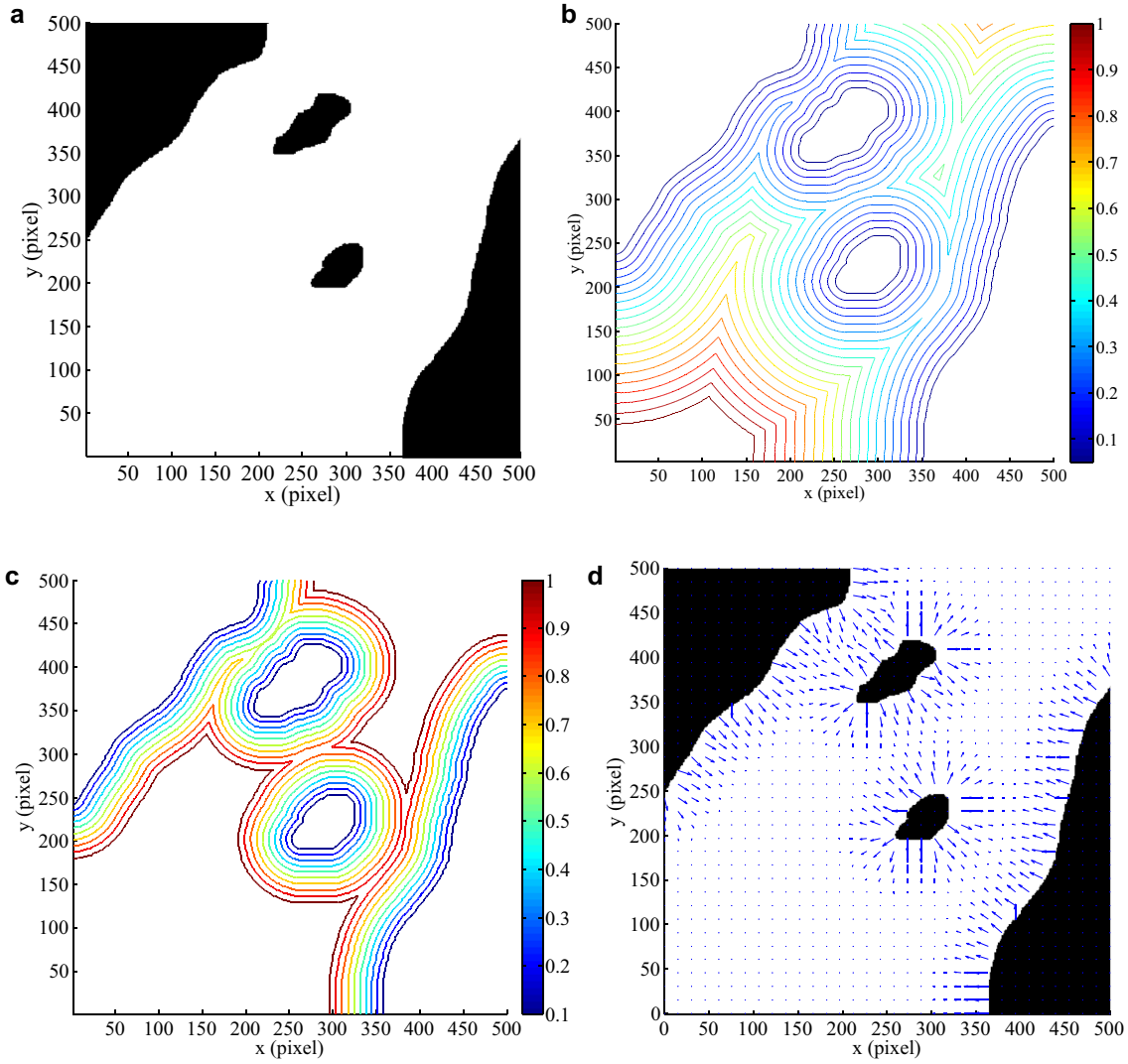

Fig. 4. The construction of the adaptive repulsive force field. (a) The simulated environment represents a typical maritime environment including two small islands and a channel. (b) The generated repulsive potential field (D_{rep}) with $\alpha = 0.8$. In the potential field, the further away the location from the obstacles, the higher the potential value. (c) The generated repulsive potential field (D_{rep}) with $\alpha = 0.3$. (d) The obtained repulsive force vector field ($\boldsymbol{F}_{Rep}$) with $\alpha = 0.3$, where the repulsive force has been clearly shown. The closer the distance to the obstacles, the larger the repulsive force.

produces a reference trajectory that the USV is able to follow for most of the operation. 2) the on-line path planning works alongside the collision risk assessment scheme to modify the path to avoid any unexpected static or dynamic obstacles. In the following sections, algorithms for solving the multi-task allocation[1] and path planning will be described individually.

3.1. Improved self-organising map (SOM) for TSP

3.1.1. SOM based algorithm for TSP

The self-organising map (SOM) proposed by Kohonen [12] is a type of artificial neural network using unsupervised learning to produce low dimensional representation of an input space. The basic structure of SOM is a two-layered network including the output layer and the input layer as represented in Fig. 6a. Neurons that need to be trained are contained in the output layer, and the training is achieved via the connections between output and input layers. In general, the nodes on the input layer are randomly located; while, in the output layer, a two dimensional regular topology such as the rectangular or the hexagonal grid is used and such a topology is largely maintained throughout the training process.

When applying the SOM to the TSP, each node C_i in the input layer represents a city (or a task in task-allocation problem) with a Cartesian coordinate (c_x, c_y) representing its location. The weights associated with the output layer neurons are in the same dimension and indicates the locations of neurons. Since the fundamental requirement of the TSP is to visit all the cities and eventually return to the start point, a circular topology (shown in Fig. 6b) is thus used in the output layer. After forming such a topology, neurons then compete to become the winner throughout the training process, which adopts an unsupervised strategy consisting of two different procedures:

- *Winner selection*: a city (C_i) is first selected from the input space. The Euclidean distance between this city and all the neurons in the output layer is then calculated by:

$$d_{CW} = |C_i - W_j| = \sqrt{(c_{xi} - w_{xj})^2 + (c_{yi} - w_{yj})^2}, \tag{10}$$

where W_j represents the location of the jth neuron, and the winner neuron is selected as the one having the minimum dis-

[1] multi-task allocation and travelling salesman problem (TSP) mathematically share the same meaning in this paper.

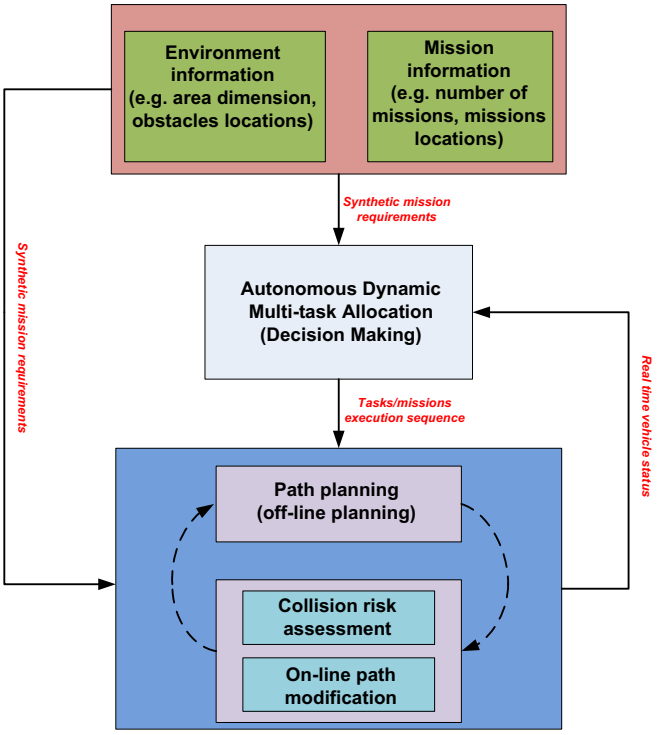

Fig. 5. The structure of the proposed of multi-task allocation and path planning algorithm for USVs.

tance value:

$$W_{win} = \text{argmin}_W (d_{CW}) \tag{11}$$

- *Neighbourhood updating*: after the determination of the winner neuron, the neighbourhood updating process will move the winner neuron as well as its neighbour points towards new positions according to:

$$W_j = W_j + \mu * f(d, G) * (C_i - W_j) \tag{12}$$

where μ is the learning rate, which determines the computation time. $f(d, G)$ is the neighbourhood function identifying the neighbourhood size of the winner neuron. The function can be viewed as a smoothing kernel which has the central influence on the winner neuron, and various forms can be used to define such a function with the most commonly used being:

$$f(d, G) = \exp(-d^2/G^2) \tag{13}$$

where d is the lattice distance on the topology and G is the gain parameter, which defines the width of the kernel (or the size of the neighbourhood). It should be noted that G should be a monotonically decreasing function of time, This is because during the initial stages, neurons are relatively far away from the cities, a large G is preferred to make the neurons can fast move towards its corresponding city; whereas, at the latter stages, a stable situation has been formed and neurons become less competitive, therefore a small value of G should be used.

Another important issue associated with the SOM for TSP is the determination of the number of neurons. Too few neurons make it difficult for the SOM to find the exact solution of the TSP; however, too large a number of neurons will be a waste of computation resource. After a series of experiments, it has been recommended in [23] that $2N$ or $3N$ of neurons is sufficient, where N is the number of cities.

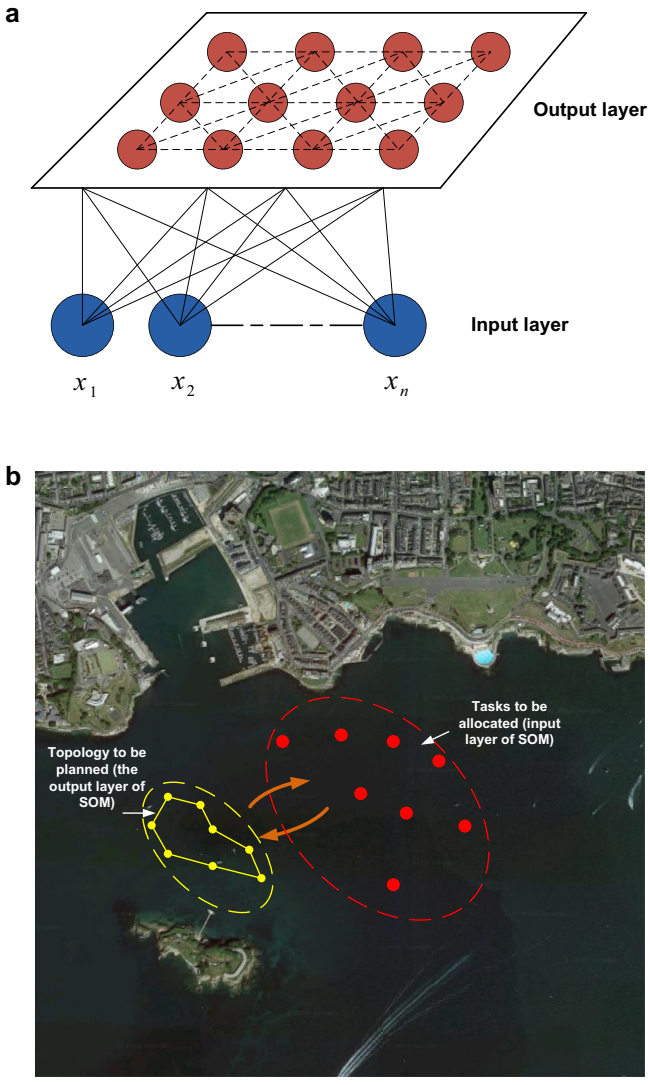

Fig. 6. Illustration of the self-organising map (SOM). (a) Basic structure of the SOM. A two-layered neural network including input layer and output layer is used. (b) In the application of the SOM for TSP, a circular topology for the output layer is used.

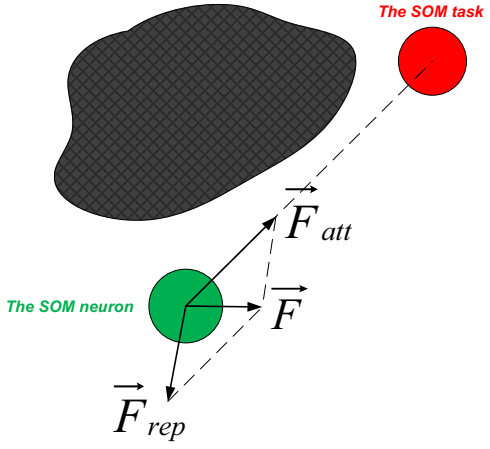

Fig. 7. The improved neuron updating process with the integration of the repulsive vector field. Neurons will now not only be updated based on the attractive force from the city but the repulsive force from obstacles.

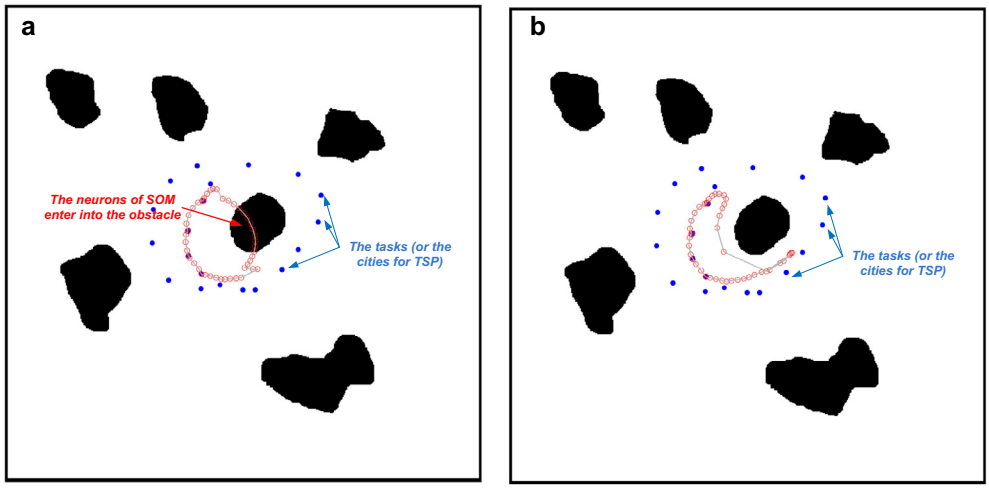

Fig. 8. The comparison of the updating processes. (a) The updating process of the conventional SOM. (b) The updating process of the improved SOM with collision avoidance capability.

3.1.2. Improved SOM for TSP with obstacle avoidance capability

The Euclidean metric based neuron update function can be explained from the potential field perspective. The second term on the right hand side of Eq. (12) is acting as an attractive force, which attracts the neurons to move towards the selected city (C_i). By following the Euclidean distance, a shortest updating route can be clearly formed. However, it is also obvious that without any emphasis on the obstacle avoidance, in a constraint environment the neuron may enter an obstacle area, which eventually provides a compromised solution of the problem.

Given the repulsive force field created in the previous section, to incorporate obstacle avoidance capability, an improved SOM algorithm has been proposed in this paper with the algorithm's pseudo-code described in Algorithm 2. The main modification made to Eq. (12) is to have a new neuron updating rule. A repulsive force is added and the neurons updating process will follow the equation as:

Algorithm 2 Improved Self-organising Map Algorithm.

Require: set of input cities (χ) in two dimensional space, parameters of the improved SOM ($d, G, \mu, \alpha_{speed}, d_{min}$), the maximum number of iterations ($iter_{max}$)

Initialise N neurons of SOM with a ring topoloty as $W = (W_1, W_2, \ldots, W_N)$. ▷ Initialisation
$iter \leftarrow 0$
while $iter \neq iter_{max}$ **do**
  Randomly select a city (C_i) from (χ)
  Find the winning neuron (W_i) of the city (C_i) from W using Eqs. (10) and (11)
  for each neighbourhood point (W_j) of the winning neuron (W_i) **do**
    if W_j is in the obstacle area **then**
      $W_j = W_j + \mu * f(d, G) * (\frac{(C_i - W_j)}{\|(C_i - W_j)\|} + \overrightarrow{f_{rep}})$
    else
      $W_j = W_j + \alpha_{speed} * \mu * f(d, G) * \frac{(C_i - W_j)}{\|(C_i - W_j)\|}$
    end if
  end for
  $iter \leftarrow iter + 1$
end while
return W

$$W_j = \begin{cases} W_j + \mu * f(d,G) * (\frac{(C_i - W_j)}{\|(C_i - W_j)\|} + \overrightarrow{f_{rep}}) & \text{if } d_{obs}(W_j) <= d_{min}, \\ W_j + \alpha_{speed} * \mu * f(d,G) * \frac{(C_i - W_j)}{\|(C_i - W_j)\|} & \text{otherwise.} \end{cases} \quad (14)$$

and $\overrightarrow{f_{rep}}$ is calculated from:

$$\overrightarrow{f_{rep}} = \boldsymbol{F}_{rep}(w_{xj}, w_{yj}) \quad (15)$$

where $\boldsymbol{F}_{rep}(\bullet)$ is the function returning the local repulsive force at W_j's position ((w_{xj}, w_{yj})) by referring to the repulsive force vector field $\boldsymbol{F}_{rep}$. $d_{obs}(W_j)$ calculates the minimum distance to the obstacles from the position of W_j. d_{min} is a predefined minimum distance. Eq. (14) reveals that when $d_{obs}(W_j) <= d_{min}$, neurons are close to obstacles and the obstacle avoidance is thus triggered by updating neurons based upon both attractive and repulsive forces (shown in Fig. 7); whereas, if neurons stay in the safe areas, the conventional updating is applied but a new parameter ($\alpha_{speed} > 1$) introduced to achieve a fast convergence speed.

To demonstrate the obstacle avoidance capability, a comparison between the conventional SOM and the improved SOM is represented in Fig. 8 using an environment containing multiple obstacles and 20 tasks to be visited. Fig. 8a shows the updating process of the conventional SOM. It can be observed that some of the neurons have already entered into the area of the obstacle; whereas in Fig. 8b, using the improved SOM, neurons can be kept well away from the obstacle by following the guidance of the repulsive force. Also, the neurons having no collision risks with obstacles (mainly the neurons on the left section of the ring), almost the same topology is maintained, which demonstrates that the improved SOM can largely preserve the feature of the conventional SOM. This is an important characteristic, especially when analysing the optimality of the final solution. As mentioned before, the conventional SOM updates the neurons by maintaining the neighbourhood topology. Within the generated tour, from each city the nearest location is always connected; therefore, the shortest sub-tour will be achieved making the overall tour near-optimal [13]. Hence, because the neighbourhood preserving feature can be largely maintained in the improved SOM, the optimality of solution is also retained.

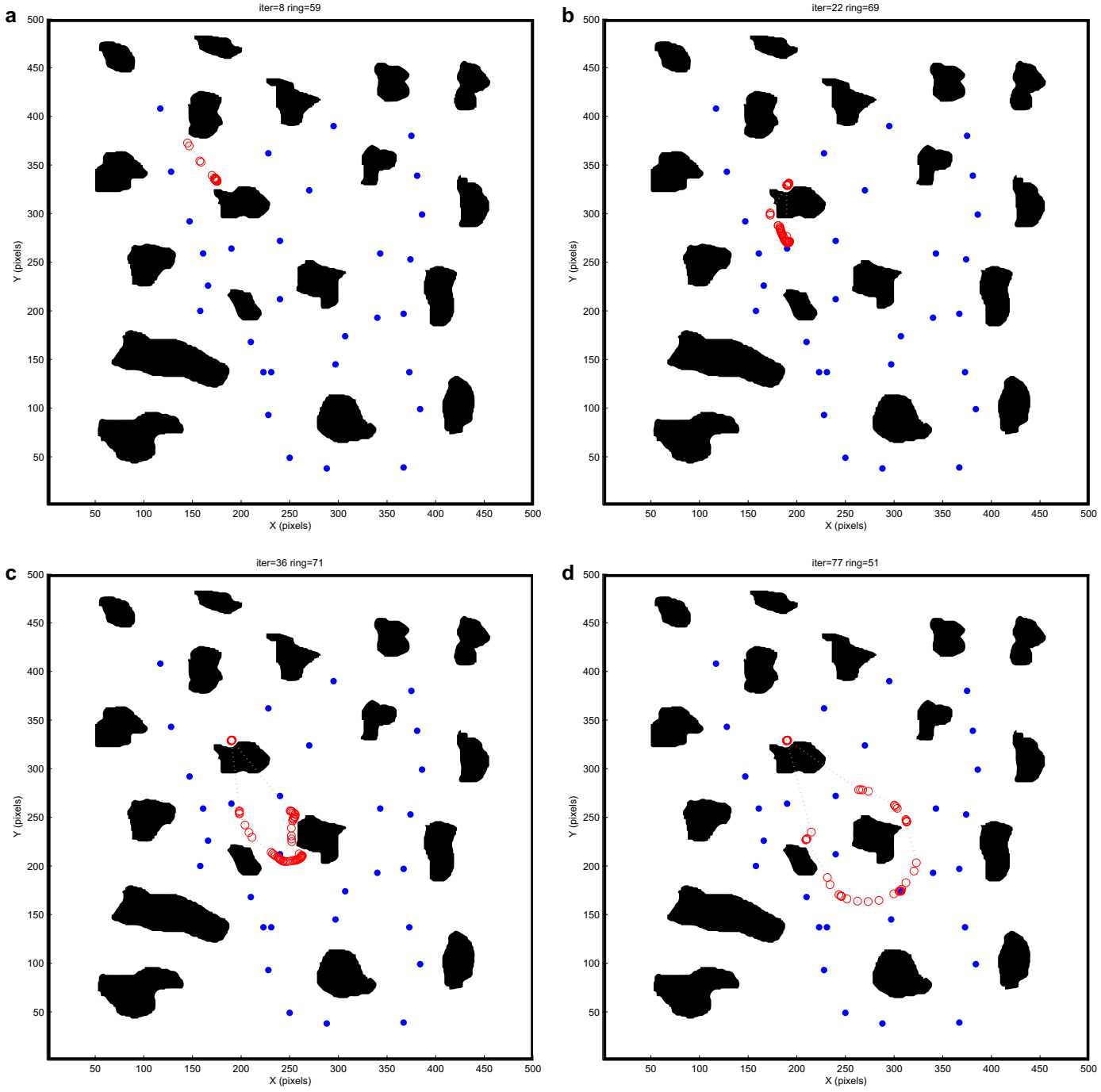

Fig. 9. The updating processes of the improved SOM for multi-task allocation in cluttered environment. A total number of 30 tasks are located in the area of interest, which includes 18 randomly generated obstacles. While the SOM is being updated, all neurons can be kept well away from obstacles. (a) Step 13. (b) Step 22. (c) Step 36. (d) Step 77. (e) Step 160. (f) Step 288. (g) Step 372. (h) Step 462.

3.2. Multi-goal path planning for USV

Using the improved SOM, an optimised task execution sequence can be accurately determined. According to Fig. 5, the path planning algorithm is then called to generate trajectories visiting each mission point. In this paper, the trajectory is calculated on the basis that minimum distance cost should be achieved, and intuitively, the straight line connecting each pair of mission points should be considered as the optimal path. Results provided by the SOM can be used as the initial solution for path planning, however further path improvements are required to address two critical issues, i.e. redundant nodes occurring between two missions and the connection between two missions may pass through an obstacle.

- *Node creation and deletion*: After running the SOM algorithm proposed previously, each mission point can be assigned a neuron and a closed tour visiting each mission is formed with all the neurons being kept outside obstacles areas. However, redundant neurons will be possibly located between some of the mission points making the connection of two points unfeasible to be used as the optimal path. This is caused by the SOM's

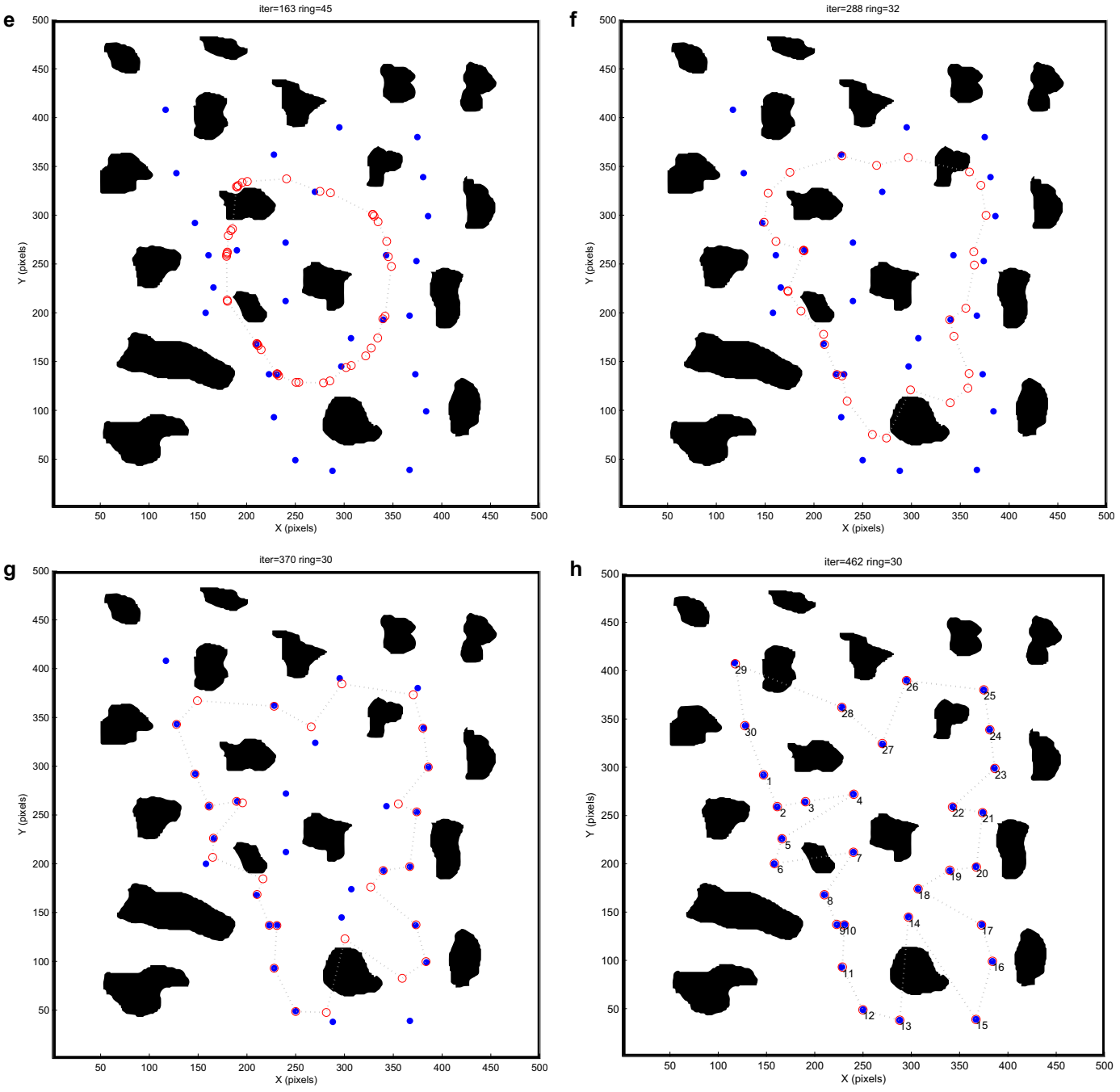

Fig. 9. Continued

configuration making the number of neurons more than that of the cities. Therefore, to address such an issue, the on-line neuron duplication and deletion scheme proposed in [1] has been used in this paper to update the SOM with the optimal number of neurons. The main concept herein is to start the SOM updating process with a single neuron, and a neuron is duplicated if it is the winning neuron for two different cities in any iteration and deleted if it has not been a winner after three iterations.

- *Path refinement*: Without redundant neuron nodes, it can be assured that each mission will be connected with the straight line, which is the optimal path for minimum distance. However, the connection between two missions may still pass through an obstacle as only the neurons are considered to be collision free when updating the SOM. One of the possible ways to address this problem would be to incorporate the path planning problem into the SOM in such a way that during each iteration any connection of two neurons that pass through an obstacle will be re-planned to provide a collision free path. Such a solution is theoretically feasible but not practically applicable. This is because any path planning algorithm is time consuming, and for the case when a considerable number of missions are included in a complex environment, the execution of path planning together with the SOM will be very slow, which is not a benefit for practical applications. In addition, during the initial steps of the SOM, neurons are in a rather unstable state and will change their position dramatically. Making the use of a path planning algorithm at this stage is therefore unfeasible. Hence, in this paper, instead of the direct integration, path planning algorithm

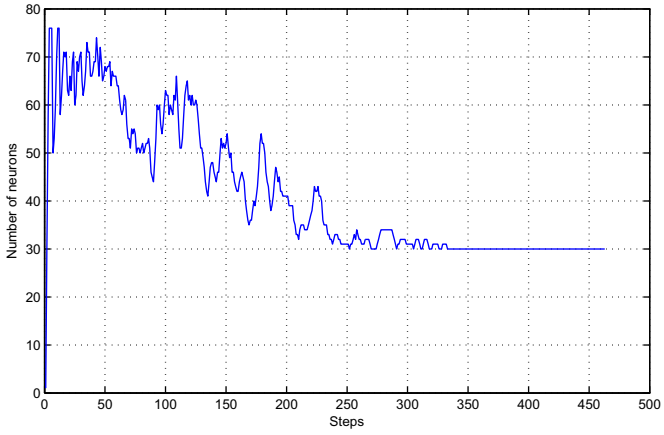

Fig. 10. The number of generated neurons during the SOM updating process. Addition and deletion processes have been constantly used to ensure optimal number of neurons will be generated.

has been applied after running the SOM and node deletion algorithms.

The adopted algorithm is the fast marching square (FMS) method proposed in [9] and [14]. It is an improved version of the FMM, which maintains the features of the FMM such as the path's continuity, smoothness and provides additional features such as improved safety. The FMS is represented in Algorithm 3.

Algorithm 3 Fast_*Marching_Square* Algorithm.

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

1: **for** each point a in obstacle area in M **do**
2: $obstaclePoints \leftarrow obstaclePoints + a$
3: **end for**
4: $M_s \leftarrow FMM(M, obstaclePoints)$
5: $T_{FMS} \leftarrow FMM(M_s, p_{start})$
6: $T_{FMS} \leftarrow fieldRescale(T_{FMS}, \beta)$
7: $path \leftarrow gradientDescent(T_{FMS}, p_{start}, p_{end})$
8: **return** $path$

It first generates a safety potential map ($\mathbf{M}_s$) by applying the FMM to propagate interfaces from all the points in obstacle areas. Based on $\mathbf{M}_s$, the FMM is executed again from the start point to generate the potential field $\mathbf{T}_{FMS}$. A safety parameter (β) is now introduced to rescale $\mathbf{T}_{FMS}$ by following the same procedure described in generating the adaptive repulsive force field. The dimension of $\mathbf{T}_{FMS}$ can thus be adjusted adaptively to assist with providing the safest trajectory. Finally, based upon the $\mathbf{T}_{FMS}$ the gradient descent method will be used to search for the optimal trajectory.

4. Algorithm validation

To validate the proposed algorithm, two different sets of computer based simulations have been carried out in this section. The primary aim of the first simulation set is to verify the effectiveness of the improved SOM algorithm dealing with multi-task allocation in a 'self-constructed' environment; whereas, the second test has considered both the multi-task allocation and path planning requirements and uses a practical maritime environment for simulations. A number of different simulation configurations have been set in both tests so as to demonstrate the capability of the proposed algorithm dealing with various practical requirements, and the results have been analysed comparatively to show the performances. The algorithm has been coded in Matlab and simulations are run on the computer with a Pentium i7 3.4 GHz processor and 4 GB of RAM.

4.1. Simulations of multi-task allocation using the improved SOM algorithm

4.1.1. Validation of collision avoidance capability

The aim of this test is to validate the effectiveness of the proposed improved SOM dealing with multi-task allocation in a cluttered environment. A 500 pixels × 500 pixels area has been constructed, in which 18 irregular shaped obstacles are tightly located, to represent a typical constrained maritime environment (shown in Fig. 9a). A total number of 30 tasks are also included in the area with their positions randomly and sparsely located in the free space (white area). In addition, in order to prevent the neurons colliding with obstacles during the updating process, parameter α in Eq. (7) has been configured to be 0.1 such that the algorithm is able to find the sequence for task execution with the optimal distance and ensured safety.

Simulation results have been shown in Fig. 9, which presents the updating processes at different time steps. A single neuron is generated at initial time step with its coordinates located at (120,420), the algorithm then uses the on-line neuron creating strategy to increase the number of neurons so as to appropriately assign each task with a neuron. For example, at time step 8 (shown in Fig. 9a), 59 neurons have been generated to form the ring of the SOM. It also can be observed that because of the attraction forces generated by the target points at the bottom right of the area, the SOM ring moves towards the centre of the operational space to ensure that tasks in 'remote' areas can also be assigned with a neuron on the ring.

The collision avoidance capability of the improved SOM algorithm can also be demonstrated from the simulation results. As the updating process is being iterated, the SOM ring encounters a number of obstacles and is able to avoid violating these obstacles by following the guidance of the generated repulsive force field. In Fig. 9b, instead of entering the obstacle, the ring has been split to form two different parts. Nodes staying above the obstacle will be primarily used to map the tasks on the upper section of the area; whereas the rest nodes will continue to move towards the centre part of the area but being well kept away from the obstacles (shown in Fig. 9c).

Once it has reached the centre of the area, the ring will then be expanded so as to better cover each task. Such a phenomenon has been clearly represented in Fig. 9d–f. It also should be noted that the number of the neurons is decreasing during these time steps. This is because that during the initial steps neurons have been created constantly to make sure that enough nodes can be generated to cover all tasks in the area of interest. However, once a neuron has reached the threshold, which has been set as the 2.5 times of the number of the tasks, the deletion process will become dominant to eliminate those neurons that have won after 3 times in an iteration. The number of neurons on the SOM ring generated at each time step has been recorded and shown in Fig. 10, it can be seen that from time step 333 the number of neurons has been stable and remains at 30, which is the same as the number of tasks. At time step 462, all tasks have been assigned with a neuron, which subsequently generates a task execution sequence shown in the Fig. 9h. Note that when executing the tasks between tasks 6–7, 13–14 and 28–29, the straight line cannot be used as the optimal route as it will violate the obstacle area, it is therefore required to use the path planning algorithm to generate a new route in these places, the result of which will be represented in later simulations.

To further demonstrate the performance of algorithm, a comparison of computational time has been presented in Table 1. The proposed algorithm is tested in the same complex environment as

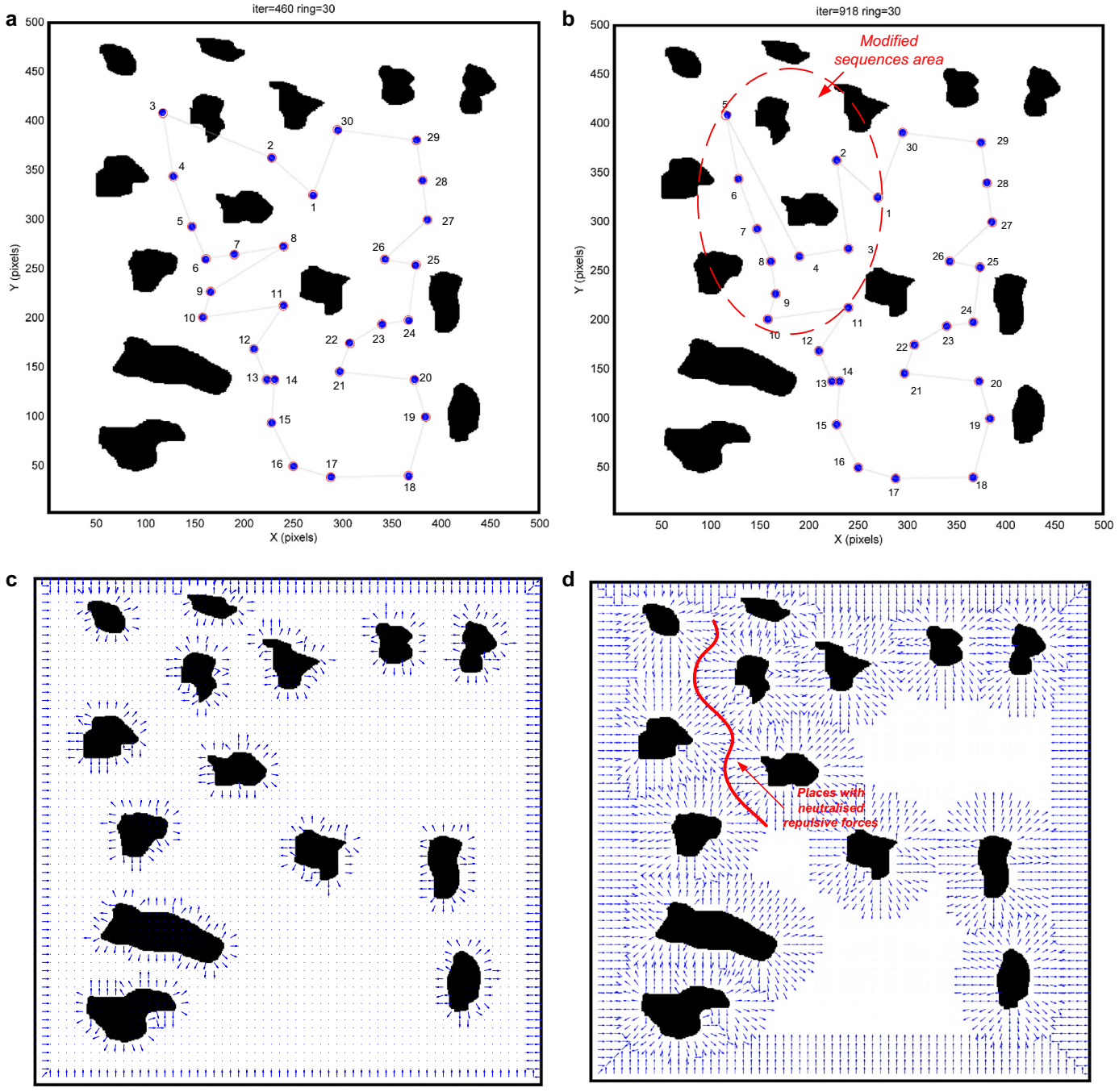

Fig. 11. Simulation results for different safety requirements. (a) Results when $\alpha = 0.1$. (b) Results when $\alpha = 0.5$. (c) The generated repulsive force field when $\alpha = 0.1$. (d) The generated repulsive force field when $\alpha = 0.5$. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article).

Table 1
Comparison of computational time for different environments with different obstacles.

No. of tasks	Time (s)	Propagation scale limit
10	17	0.1
20	25	0.1
30	26	0.1
40	38	0.1
50	50	0.1

shown in Fig. 9a but with different number of tasks to be executed. It can be observed that the computational time, as one would expect, grows with the number of tasks with the longest time taking about 50 s, which is fast enough for the USV to make an off-line decision. However, when the algorithm is implemented for the on-line decision making, improvement is still required to reduce the computational time down to seconds. One of the possible approaches would be to port the current algorithm from Matlab to more computational efficient languages such as C++. Also, current algorithm is coded in single-threaded execution structure, some subroutines could be parallelised so that the multi-core processor can be used to improve the computational efficiency.

4.1.2. Validation of safety settings

In this section, the capability of the algorithm dealing with different safety requirements has been verified. A similar simulation environment to the previous test has been used, which is shown

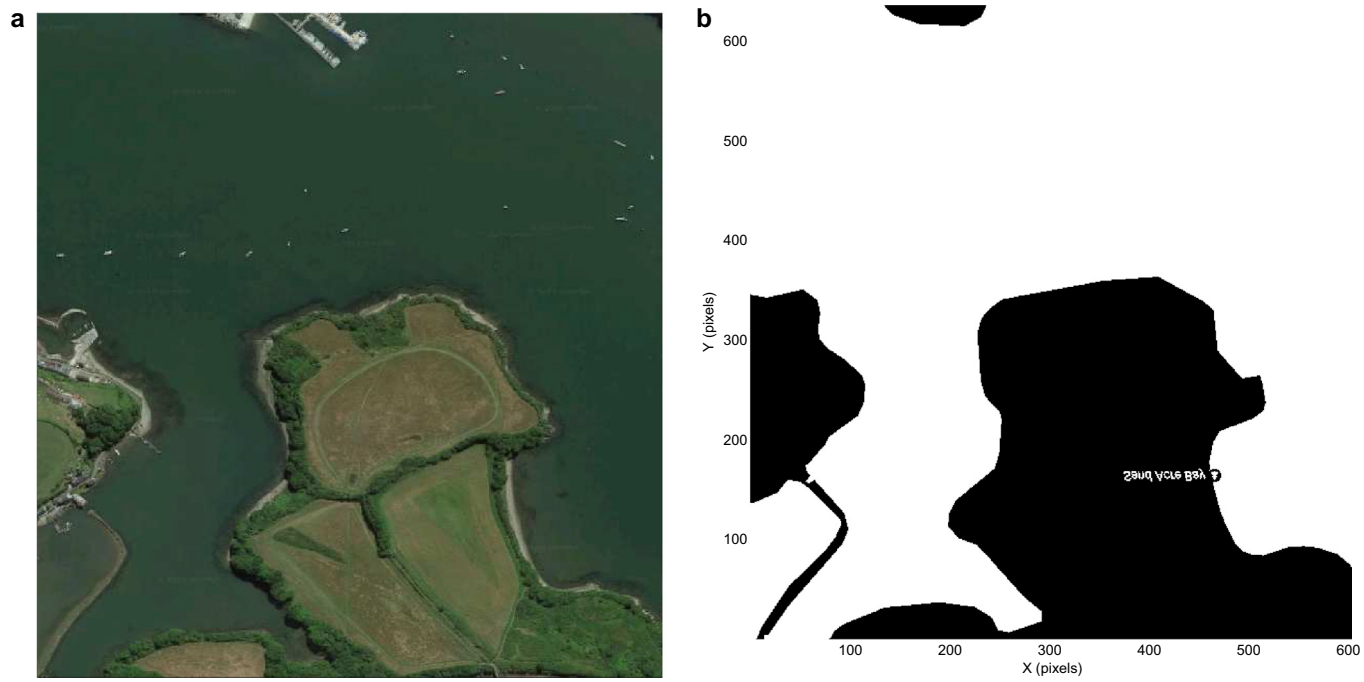

Fig. 12. A practical simulation environment. (a) The environment near Forder Lake, UK with the region ranging from N52°23'34.08"/W4°14'6" to N50°23'34.08"/W4°13'48". (b) The converted binary map of the electronic map.

in Fig. 11a. However, less obstacles are included in the right hand side of the environment while obstacles in the left hand side remain the same. The reason for such a configuration is to compare the algorithm's capability in areas with dense and sparse obstacles, respectively. To evaluate the algorithm's performance against different safety requirements, two different values for *propagation scale limit* (α) have been used, with the values being 0.1 and 0.5, respectively. A small value for α means the algorithm has less concerns on the safety (e.g. the distance to the obstacles) and more emphasise on the total distance.

Simulation results have been presented in Fig. 11. Fig. 11a and Fig. 11 b show the obtained task executing sequences with α values of 0.1 and 0.5, respectively, and Fig. 11c and Fig. 11d illustrate the generated repulsive force for the associated α. By comparing Fig. 11a and b, it can be observed that similar sequence can be obtained under different values of α. Especially, on the right hand side of the environment where less obstacles exist, an exactly same result has been obtained for visiting the task from 11 to 30. However, in the area with dense obstacles (left hand side of the environment), a different executing sequence has been obtained. In Fig. 11a, a more aggressive approach has been adopted placing the generated sequence closer to obstacles. In particular, the straight line connecting tasks 2 and 3 has already violated the obstacle, which requires the path planning algorithm to further refine this route when the USV is carrying out the tasks. On the contrary, in Fig. 11b, as the value of α is increased to 0.5, a safer strategy can be formed by keeping the straight lines connecting each pair of tasks away from obstacles, and this means that the USV can simply follow this sequence to visit each task without additional path refinement. The main reason behind this phenomenon is that sparsely located obstacles create the repulsive force field with less influence in the left hand side of the environment (shown in Fig. 11c and d) making the algorithm update the SOM with minor effect from the repulsive force. However, in the area marked by the red circle in Fig. 11b, densely located obstacles generate the repulsive force field which fills up the space (shown in Fig. 11d), and when the SOM is being updated, neurons tend to stay along the areas with neutralised force (marked by the red line in Fig. 11d) making the final formed ring further away from the obstacles. It however should also be noted that such a difference is established on the sacrifice of the total distance. As shown in Table 2, the total distance for the test with α equal to 0.1 is 1570 pixels, which is 3% less than that of the test having an α value of 0.5.

4.2. Simulations of multi-task allocation and path planning

In this section, simulations have been undertaken to validate the capability of the proposed algorithm dealing with multi-task allocation as well as path planning. It is assumed the USV is carrying out an environmental monitoring mission, which requires the vehicle to collect the water sampling data from multiple water monitoring stations. The area of interest has been selected near Forder Lake, UK with the region ranging from N52°23'34.08"/W4°14'6" to N50°23'34.08"/W4°13'48". The electronic map of the given area is shown in Fig. 12a, which is further converted to a 640 pixels $\times$ 602 pixels binary map represented in Fig. 12b.

Apart from the general requirements of the environmental monitoring mission, i.e. the USV should visit all the water monitoring stations with optimal cost, another important issue that needs consideration is the effect of the high and low tide has. As discussed in Introduction, different states of tide will significantly influence the dimension of an obstacle making some of the water sampling stations beyond the USV's capability to visit during low tide periods. It therefore requires the algorithm to determine which stations (or tasks) can be executed in accordance with the specific situation and subsequently allocate the tasks and plan the trajectory.

Fig. 13 presents simulation results when considering the tide effect. In Fig. 13a, 20 water sampling stations (displayed as the blue dots) are located in the area. Among them, three stations (marked by red dash circles) have been chosen to stay in the littoral areas. During the low tide period, because of the low water level

Table 2
Simulation configurations for different safety requirements.

No. of simulation	No. of obstacles	No. of tasks	Propagation scale limit	Total distance
1	14	30	0.1	1570 pixels
2	14	30	0.5	1619 pixels

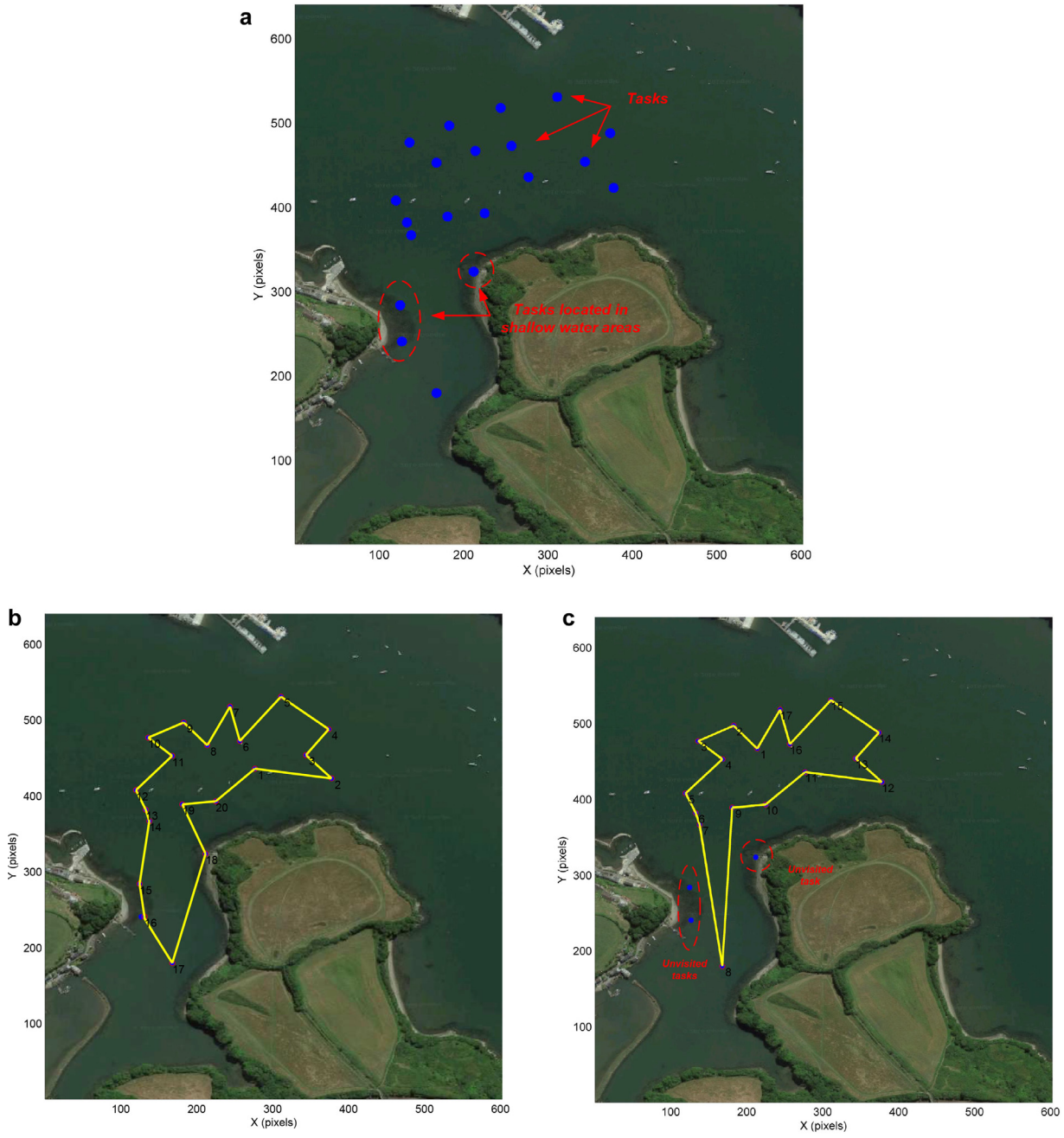

Fig. 13. Simulation results with 20 water sampling stations. (a) The simulation environment. Three stations (marked by red dash circles) have been chosen to stay in the littoral areas, which will be affected by the tide. (b) Results when $\alpha = 0.1$. (c) Results when $\alpha = 0.3$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the USV is prevented from visiting these three stations. The tide height effect has mainly been accommodated for by adjusting the *propagation scale limit* (α) in this paper. During the rising tide period, the decreasing dimension of the obstacle area makes the free space larger, small value of α has therefore been selected to make the algorithm able to explore more tasks. The result when $\alpha = 0.1$ has been presented in Fig. 13b. It can be observed that a closed tour shown in yellow line is able to cover all 20 stations (tasks). On the contrary, Fig. 13c shows the results with $\alpha = 0.3$. Note that tasks located in the shallow water areas have been excluded because of the generated repulsive force; whereas the rest of tasks can still be visited by the closed tour shown by the yellow line.

Fig. 14 shows another simulation case where a total number of 30 stations have now been selected. These stations now all stay

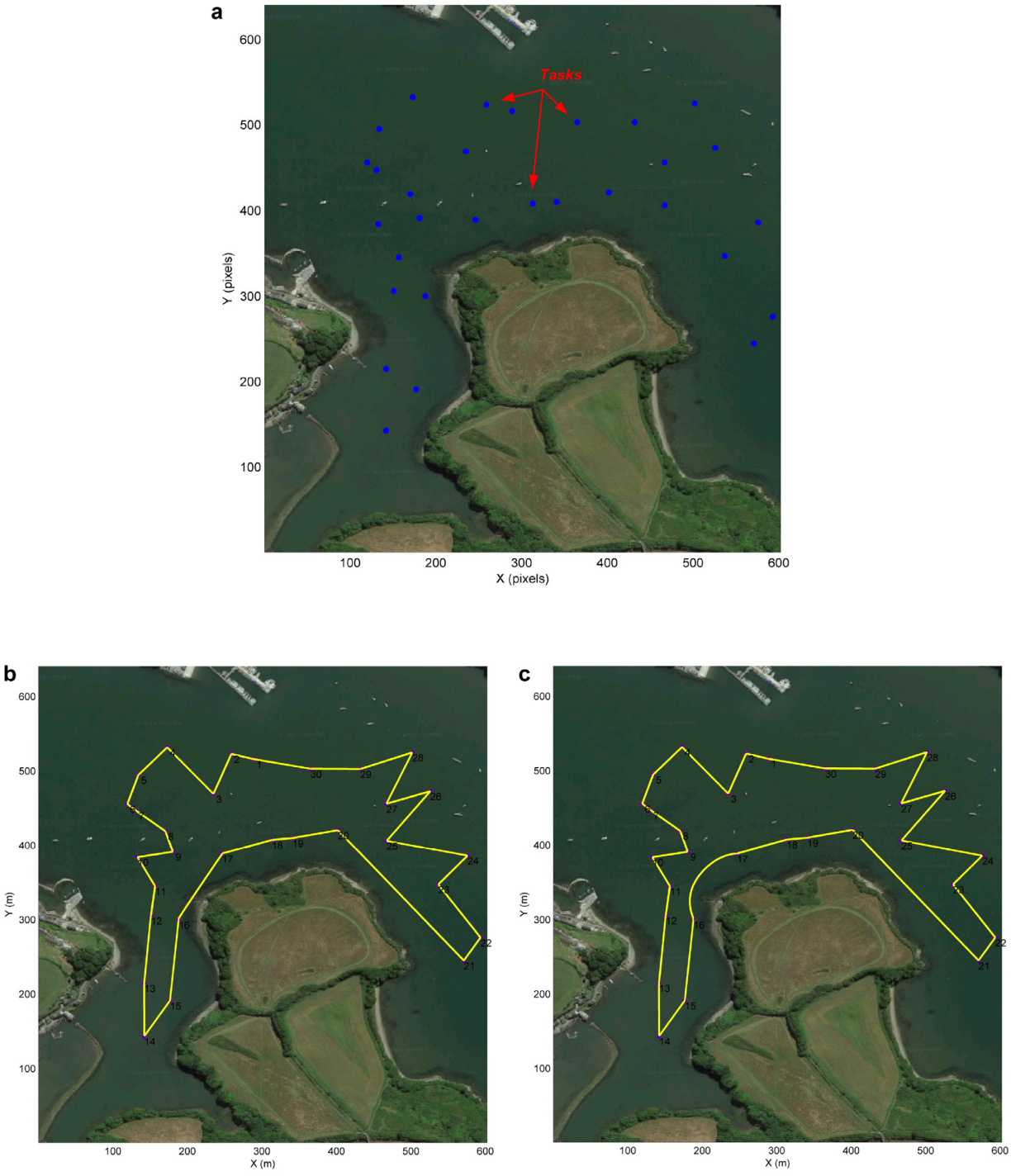

Fig. 14. Simulation results with 30 water sampling stations. (a) The simulation environment. All stations now all stay well away from shorelines however the straight line connecting each two stations may be too close to land in different tide periods. (b) Results when $\alpha = 0.1$. (c) Results when $\beta = 0.3$.

well away from shorelines however the straight line connecting each two stations may be too close to land. Fig. 14b shows the result when $\alpha = 0.1$. The proposed algorithm can still generate a closed tour to visit these stations. However, it should be noted that the straight line between task 16 and 17 crosses the shallow water areas making such a route may not be feasible during the low tide period. Therefore, a path planning algorithm is required for this route when the USV is executing the tasks, and this result has been presented in Fig. 14c. The FMS algorithm is utilised by selecting the safety parameter (β) equal to 0.3. It can be observed that the updated route forms a curve to successfully avoid the prohibited areas.

5. Conclusions and future work

In this paper, a new multi-task allocation and path planning algorithm for USV operating in cluttered maritime environment has been proposed with the main contribution on addressing the safety issues. In terms of the multi-task allocation, an improved SOM with collision avoidance capability is designed and developed.

It has been demonstrated that the algorithm can fast assign the tasks and generate an execution sequence according to different safety requirements imposed by obstacles. Using the optimised execution sequence, the fast marching square (FMS) based path planning algorithm will then be used to calculate the trajectory. Paths can be adaptively modified based upon the dimensions of obstacles to ensure the safety while the USV is navigating.

To further develop the algorithm, improvement will be first carried out to design a new decision making algorithm to help determine the value of *propagation scale limit* (α). As mentioned in Section 2.2, α is influenced by a number of different factors such as the safety and distance requirements, and the value of it is calculated according to specific environment situation. For example, when the USV is navigating in a cluttered environment, safety issue is of more importance than the distance requirement, therefore α should be calculated with more weighting assigned to the safety factor. According to this and to fast determine α, a priority assessment strategy will be designed and incorporated within current algorithm to intelligently assign weightings to different factors and consequently calculate the value of α.

The second improvement is to consider more practical environmental constraints, especially the influences generated by the surface currents. Strong currents may affect the tracking performance of a USV making it unfeasible for the vehicle to execute certain tasks that are located in the areas with complicated hydrology condition. Algorithms should be designed to be able to monitor these environmental constraints and dynamically update the task execution sequence. In addition, from the path planning level, the effect from surface currents could be compensated for by generating trajectories that make the best use of the currents flow. Another important environmental constraint is the water depth between the reef top and the sea level. It is possible for a USV to collide with the underwater reef top, which is difficult to be observed from the image information. Therefore, to effectively avoid such a risk, additional sensing devices such as underwater camera or sonar should be mounted on the USV to detect the environment, and the designed algorithm should be capable of considering those environment information to generate more secured task execution sequence and path planning results.

The third improvement is to add the dynamic obstacles avoidance capability into the path planning algorithm. Currently, only static obstacles have been considered; whereas in a complex maritime environment, the collision avoidance with moving ships is equally crucial. In particular, evasive actions should be generated by adhering to the maritime navigation regulations (the International Regulations for Preventing Collisions at Sea 1972 (COLREGs)). Therefore, to successfully achieve the dynamic collision avoidance capability, an intelligent reasoning model is required, and one of the effective approaches would be to use the self-organising fuzzy neural network (SOFNN) [19], which provides superior learning capability and is adaptive to disturbances in the reasoning environment. When developing the decision making system based upon the SOFNN, conventional collision avoidance conditions including the velocities of the ships and distance between two ships can be merged with COLREGs to effectively generate an optimised evasive action, i.e. the steering angle that a USV should take to avoid a moving vessel.

The final improvement is to expand the current algorithm to a larger scale unmanned vehicles system. Multiple USVs formations can be used to undertake more complicated missions which require long mission duration. In addition, a cross-platform system involving the cooperation between USVs and unmanned aerial vehicles can also be developed to support ocean operations. All of these systems would require more sophisticated task-allocation and path planning algorithms considering wider range constraints such as the scalability and flexibility of multiple vehicles.

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. N0014-10-1-0652, the group consists of universities and industry partners conducting small ships related researches. The authors are also indebted to Mr. Konrad Yearwood for his valuable critique of this paper.

References

[1] B. Angeniol, G.D.L.C. Vaubois, J.-Y. Le Texier, Self-organizing feature maps and the travelling salesman problem, Neural Netw. 1 (4) (1988) 289–293.
[2] E. Cochrane, J. Beasley, The co-adaptive neural network approach to the Euclidean travelling salesman problem, Neural Netw. 16 (10) (2003) 1499–1525.
[3] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[4] J. Faigl, On the performance of self-organizing maps for the non-Euclidean traveling salesman problem in the polygonal domain, Inf. Sci. 181 (19) (2011) 4214–4229.
[5] J. Faigl, An application of self-organizing map for multirobot multigoal path planning with minmax objective, Comput. Intell. Neurosci. 2016 (2016) 1–15.
[6] J. Faigl, M. Kulich, V. Vonásek, L. Přeučil, An application of the self-organizing map in the non-Euclidean traveling salesman problem, Neurocomputing 74 (5) (2011) 671–679.
[7] J. Faigl, L. Přeučil, Inspection planning in the polygonal domain by self-organizing map, Appl. Soft Comput. 11 (8) (2011) 5028–5041.
[8] S. Garrido, L. Moreno, D. Blanco, Exploration of a cluttered environment using voronoi transform and fast marching, Robot. Auton. Syst. 56 (12) (2008) 1069–1081.
[9] S. Garrido, L. Moreno, D. Blanco, F. Martin, FM2: a real-time fast marching sensor-based motion planner, in: Proceedings of the 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics., 2007, pp. 1–6.
[10] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, 1, Basic Books, 1991.
[11] D.S. Johnson, L.A. McGeoch, Experimental analysis of heuristics for the stsp, The Traveling Salesman Problem and its Variations, Springer, 2007, pp. 369–443.
[12] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[13] K.-S. Leung, H.-D. Jin, Z.-B. Xu, An expanding self-organizing neural network for the traveling salesman problem, Neurocomputing 62 (2004) 267–292.
[14] Y. Liu, R. Bucknall, Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment, Ocean Eng. 97 (2015) 126–144.
[15] R.R. Murphy, E. Steimle, C. Griffin, C. Cullins, M. Hall, K. Pratt, Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane Wilma, J. Field Robot. 25 (3) (2008) 164–180.
[16] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. 93 (4) (1996) 1591–1595.
[17] S. Somhom, A. Modares, T. Enkawa, A self-organising model for the travelling salesman problem, J. Oper. Res. Soc. 48 (9) (1997) 919–928.
[18] R. Song, Y. Liu, R. Bucknall, A multi-layered fast marching method for unmanned surface vehicle path planning in a time-variant maritime environment, Ocean Eng. 129 (2017) 301–317.
[19] N. Wang, M.J. Er, J.-C. Sun, Y.-C. Liu, Adaptive robust online constructive fuzzy control of a complex surface vehicle system, IEEE Trans. Cybern. 46 (7) (2016) 1511–1523.
[20] Z. Wu, T.W. Chow, Neighborhood field for cooperative optimization, Soft Comput. 17 (5) (2013) 819–834.
[21] L. Xu, T.W.S. Chow, Self-organizing potential field network: a new optimization algorithm, IEEE Trans. Neural Netw. 21 (9) (2010) 1482–1495.
[22] X.-s. Yan, H.-m. Liu, J. Yan, Q.-h. Wu, A fast evolutionary algorithm for traveling salesman problem, in: Proceedings of the Third International Conference on Natural Computation (ICNC 2007), 4, IEEE, 2007, pp. 85–90.
[23] J. Zhang, X. Feng, B. Zhou, D. Ren, An overall-regional competitive self-organizing map neural network for the Euclidean traveling salesman problem, Neurocomputing 89 (2012) 1–11.

Dr. Yuanchang Liu received the B.S. degree in Control Engineering from Dalian University of Technology in 2010, M.Sc. degree in Power System Engineering and Ph.D. degree in Marine Control Engineering from University College London (UCL) in 2011 and 2016, respectively. He is now the Research Fellow in Surrey Space Centre, University of Surrey, UK. His research interests include marine autonomy, robotics, motion planning and multi-agent systems.

Prof. Richard Bucknall is a Professor of Marine Systems in the Department of Mechanical Engineering at UCL and a Visiting Professor in Marine Systems at Stevens Institute of Technology (NY USA). Having gained experience working in both the shipping and rail industries as a practising engineer across the world he joined UCL as a Researcher in 1995 and was promoted professor in 2012. His research interests are in exploiting technologies in the transport sector examining how new technologies should be developed and will impact the design and operation of new vehicles and is presently interested in unmanned surface vessels. He is currently working on autonomous operation of vehicles with a particular interest in unmanned surface vessels He is running many projects that support 4 Research Assistants and 10 Ph.D. studentships.