DC-HEN: A Deadline-aware and Congestion-relieved Hierarchical Emergency Navigation Algorithm for Ship Indoor Environments

1st Xiaoling Zeng
School of Navigation
Wuhan University of Technology
Wuhan, China
xlzeng@whut.edu.cn

2nd Kezhong Liu
School of Navigation
Wuhan University of Technology
Wuhan, China
kzliu@whut.edu.cn

3rd Yuting Ma
School of Navigation
Wuhan University of Technology
Wuhan, China
278827@whut.edu.cn

4th Mozi Chen
School of Navigation
Wuhan University of Technology
Wuhan, China
chenmz@whut.edu.cn

***Abstract*—Emergency evacuation is critical following a ship accident, as passengers are required to escape the dynamic hazards and reach the muster station before the deadline. In the existing efforts, users are guided to a safe path away from the danger, but unconstrained detours may mislead users to miss the ship capsizing deadline. Another major drawback is the heavy congestion during crowd evacuation. Therefore, this paper proposes DC-HEN, a hierarchical emergency navigation algorithm with both deadline and congestion awareness for ship indoor environments. Taking advantage of reinforcement learning techniques, DC-HEN can provide an individually customized evacuation route for each user in a real-time manner. We validate the proposed approach in a large-scale simulation environment with different population sizes based on a real-ship indoor scenario. Compared with the state-of-the-art solutions (CANS, ECSSN), experimental results show that DC-HEN can trade off between path efficiency and congestion to guide users to the exit safely.**

***Index Terms*—hierarchical emergency navigation, deadline-awareness, congestion-awareness, ship indoor environment**

I. Introduction

Safely guided evacuation is the first priority after a passenger ship accident, especially in recent tragic cruise ship disasters in recent years, highlighting the importance of emergency navigation strategy [1], [2]. In the absence of real-time emergency information, passengers only rely on a single floor map to find a way to the exit. Such unreliable solutions cannot ensure that pedestrians can avoid dynamic hazards and complete evacuation before the deadline [3]. Therefore, based on the ship's indoor environment, an efficient emergency navigation system capable of guiding pedestrians away from hazards in real-time is necessary.

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 51979216, and the Natural Science Foundation of Hubei Province, China, under Grants No.2021CFA001 and 20221J0059.

Rapidly advancing Internet of Things (IoT) provides new solutions for emergency navigation, with the objective of connecting everyone and everything within the system through the Internet [4]. In particular, Low-Power Wide-Area Networks (LPWAN) technologies provide better solutions to the challenge of power consumption, coverage, reliability and connectivity. Recently, the application of LPWAN involves environmental surveillance, domestic appliances, alerting systems, industrial control, etc [5]. Each specific LPWAN technology (e.g., LoRa, NB-IoT, sigfox, etc.) possess different capabilities, providing options for different application scenarios and requirements [6], [7], [8].

In emergency navigation, a diversity of solutions has been proposed. However, there are many inherent constraints preventing its direct deployment and application in ship evacuation. Firstly, part of the work relies on the navigation backbone, which comprises the key nodes deployed in the environment [9], [10]. Even if these approaches are able to guide users away from hazards, considering the practical situation where different users near the same navigation node are guided by the same path, which can lead to heavy congestion, increase the user's exposure time to hazards and decrease the evacuation success rate, as shown in Fig. 1a. Therefore, emergency navigation methods for congestion avoidance are proposed, which do not rely on the navigation backbone but on the entire navigation network [11], [12], [13], [14], [15], Unfortunately, as the above approaches rely on numerous sensor nodes, there is an issue with algorithm efficiency due to the frequent reconstruction of the entire network in a highly dynamic navigation environment. In addition, considering a ship environment with limited evacuation time, the existing solutions with unconstrained detours may increase the danger to users, as shown in Fig. 1b.

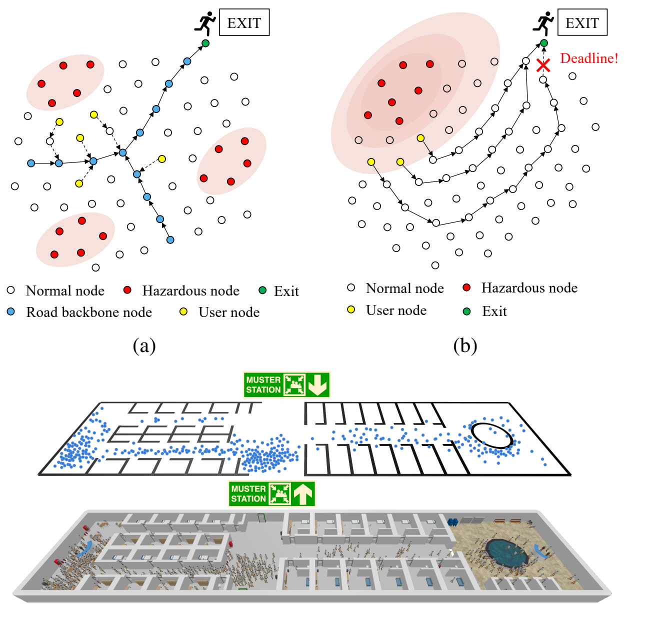


Fig. 1. Illustrative examples of the emergency navigation problem. (a) A navigation scenario based on a road backbone, where users are guided to a navigation route equidistant from the hazard area. (b) A navigation scenario based on a compound map, where users are guided to different navigation paths away from the hazard area. (c) Simulation of crowd evacuation under ship inclination.

Motivated by the above-mentioned problems, we aim to: (1) assist users to evacuate before the deadline, (2) adapt navigation instructions according to the hazards and crowds in the environment, and (3) ensure real-time performance. In order to propose an effective emergency navigation algorithm for the ship's indoor environment, we first built a model in the visualization simulation platform Anylogic [16] and simulated the crowd evacuation process based on the existing evacuation plan as shown in Fig. 1c. It is intuitively observable that unexpected congestion exists in some areas, which may lead to uneven distribution and even accelerate the process of ship capsizing. Based on these issues, we propose DC-HEN, a Deadline-aware and Congestion-relieved Hierarchical Emergency Navigation strategy for a ship indoor environment, which means that the navigation decisions are driven by both global planning as well as local environmental information. Firstly, DC-HEN extracts the spatial features of the ship layout and constructs a look-up table, which simplifies the global planning process with the awareness of the evacuation deadline. Then the RL-based local navigator follows the global guidance while dynamically adjusting to the hazards and congestion around the user.

Experiments are carried out in a simulated scenario, the training results demonstrate that DC-HEN achieves a faster convergence and the navigation success rate rises to a higher level more quickly than the classical RL method DDQN [17]. We also evaluate the navigation routes generated by DC-HEN by comparing it with state-of-the-art congestion-adaptive emergency navigation methods, CANS and ECSSN. DC-HEN can adaptively avoid congestion while ensuring no excessive path stretch.

The contributions of our work are summarized as follows:

- We constructed a crowd movement data set of ship indoor evacuation via a simulation platform Anylogic for the DC-HEN training.
- We proposed a method for constructing a graph model with environmental structural features. It can reduce the redundancy of navigation nodes, and facilitate the route query in the global planning process. Furthermore, it helps to quickly provide global guidance that can guarantee users reach the muster station before the ship capsizes.
- We developed a hierarchical emergency navigation algorithm that combines the global reference path and local environmental information based on reinforcement learning technology. And the corresponding reward structure can motivate the agent to follow the global guidance while avoiding dynamic hazards and congestion.

The rest of the paper is followed as: the related work of path planning approaches and emergency navigation methods is showed in Section II; the problem formulation is detailed in section III; the proposed algorithm design is discussed in Section IV, and it is validated by a real case study in Section V; finally, conclusions are summarized in Section VI.

II. Related Work

In this section, we briefly review the relevant work of path planning approaches and emergency navigation methods.

A. *Traditional path planning*

In robot navigation tasks, path planning is a fundamental phase before actually controlling a mobile robot. Methods can be divided into two categories according to the scale of the planning task: global and local path planning. When the prior information about the environment is complete, global path planning can deal well with a static environment and plan an optimal path based on obstacle information. For instance, the Dijkstra algorithm and A* algorithm are classical algorithms used for solving the shortest path problem [18], [19]. When applied in mobile robotics, improvements based on the A* algorithm have been proposed to deal with the dynamics in the environment, such as the D* algorithm [20]. However, when the environment is highly dynamic, the global planner needs to collect environmental information and then call the algorithm to re-plan the optimal path, which cannot guarantee real-time navigation. In contrast, the local one is more useful for dealing with the unknown or dynamic environment, such as the Artificial Potential Field (APF) [21].

To reduce overall search time and avoid local minimum, many researchers combine global and local planners for navigation [22], [23]. Wang et al. proposed a globally guided reinforcement learning approach based on the A* algorithm and the results showed its superiority among distributed methods [24]. To accelerate the convergence speed and obtain a smooth path, Dai et al. took the advantage of A* algorithm and Ant

Colony algorithm and achieved efficient searching capabilities [25]. In the crowd evacuation application, Li et al. divided the process into two parts, the optimal evacuation path is obtained at the top layer and the RVO algorithm is used for collision avoidance at the bottom layer [26]. To obtain an optimal path for each pedestrian, our work is inspired by previous work about hierarchical path planning.

B. Reinforcement learning based path planning

In recent years, reinforcement learning methods have been extensively studied in the fields of games, robots, and emergency evacuation and achieved some success [27], [28]. The main process of the reinforcement learning algorithm is that an agent in the initial state interacts with the environment by selecting an action from the action space in the current state, then transfers to the next state, and obtains the corresponding reward. Constantly repeat the exploration process and learn the optimal strategy with the highest cumulative reward [29].

In the study of path planning, the RL agent learns the optimal path without collision by exploring the surrounding environment information [30]. And RL-based methods demonstrate outstanding performance in a highly dynamic environment. However, these methods are not applicable to large-scale navigation environments due to the lack of reward information in the learning process, the learning efficiency of the algorithm is low and it is difficult to learn the optimal strategy [28], [31].

To improve evacuation efficiency, Sharma et al. abstracted the evacuation environment into a graph, taking into account fire spread as well as bottlenecks, and employed reinforcement learning to find the shortest path [32]. Xue et al. took the scene image as input, and the output after reinforcement learning directly controls the dynamic guidance signs arranged in the environment to help guide the crowd. One drawback of these methods is that they can lead to heavy congestion as the crowds are navigated to the same target [33].

C. Emergency navigation

Many researchers have taken path planning techniques from mobile robotics and adapted these for emergency navigation. And the decision support system can be divided into centralized and distributed methods. Based on the collected information about the entire environment, a centralized approach calculates the shortest path for each robot under dense information. In contrast, distributed solutions are mainly proposed in navigation tasks due to their scalability and efficiency. Li et al. proposed a distributed algorithm based on artificial potential field with a the self-organizing sensors network that is able to represent environmental information [34]. During navigation, the repulsive potential of the hazards and the attractive potential of the exit work together to guide the user along the route with the decreasing potential value. the major drawback of this is its high message overhead.

In order to reduce communication expense, Wang et al. designed a road map system in the sensors network that can adapt to the changes of emergencies [10]. In this way, many users may be directed to the identical safe path, which can cause congestion and prolong the evacuation time.

To alleviate the congestion on the navigation route, Wang et al. proposed a WSN-assisted emergency navigation algorithm, users near the emergencies are allocated to different paths at the cost of a slight detour [11]. Benefiting from IoT-enabled WSNs, Jindal et al. built a composite map that takes into account the distance to the exit and the hazard level and proposed a clogging-free navigation strategy for users [14]. Li et al. proposed to predict the environment dynamics based on the information of WSNs and then construct a potential field for each node based on the user's location as well as the corresponding edanger values [15].

However, these related works cannot be directly applied to our ship indoor environment since they ignore the hard-deadline constraint. Detours may prevent users from reaching the exit before deadline, and periodic recalculation of network parameters fail real-time planning.

III. Problem formulation

In this section, we first present our overall basic idea, introduce the navigation model as well as the definitions used in this article, and then formulate the hierarchical emergency navigation problem.

A. Basic idea

With the challenges described in Section I, the basic idea of our intended emergency navigation system is as follows: a hierarchical combination of global guidance and local emergency navigation.

We consider the complex indoor environment of a ship, where there are many walls resulting in limited traversable area. On this basis, for the time sensitivity of ship evacuation, passengers are required to reach the exit before the deadline. And we calculate the typical delay and the worst-case delay on each global segment and constrain the duration time for the user to experience. In this way, it is easy to select a particular global navigation node as the subgoal to help guide emergency navigation. Secondly, for a crowded and highly dynamic environment, the navigation instructions by the local planner need to enable the user to avoid potential congestion and dynamic emergencies. In addition, the efficiency of the algorithm is of importance, the system is called to provide real-time guidance for the user based on the sensors information until the muster station is reached.

B. Model and Definitions

Navigation model: We present the navigation environment as a 2-dimensional space with the size of $H * W$. And the navigation network is modeled as an undirected graph $G = (V, E)$, where $V = \{v_i\}$ is the set of navigation nodes, and $E = \{e_{ij} = (v_i, v_j)\}$ represents the set of the traversable path between two neighboring nodes. We denote the nearest node from the user's initial position as the start v_s and let the exit v_e locates at the node closest to the muster station. At each time step, each node determines its status based on the sensor information and pedestrians' locations explored by LPWAN. Based on the navigation model, users can only be guided to obstacle-free spaces and avoid encounters with hazards and congestion on the road. And the navigation path refers to a sequential set of all the navigation nodes from the initial node to the exit, denoted as $p_{se} = \{v_s, \dots, v_i, v_j, \dots, v_e\}$.

Assumptions: We assume that the spatial structure of the navigation environment is fixed and acquired in advance. The wireless sensors deployed in the scene can monitor the information of their detection range in real-time, and the entire evacuation scene can be covered under the guarantee of LPWAN. Then the corresponding navigation nodes in hazardous areas can switch the status in time. When danger threatening the lives of passengers occurs, an adaptive navigation strategy should guide users away from danger while avoiding congestion on the road. During the evacuation process, we assume that the smartphone carried by the user can access the status information of the surrounding navigation nodes as well as the global guidance information. Other users' navigation information is unavailable, and only their current location can be obtained.

Objectives: The purpose of emergency navigation is to guide users away from hazards and reach the exit before the deadline while reducing unnecessary congestion in the process. There are four aspects of objectives expected for this hierarchical navigation algorithm: Path safety and efficiency, Congestion avoidance, and Algorithm efficiency.

- Path safety: The navigation path is required to ensure that the user can reach the exit before the deadline and keep away from hazardous areas in the process.
- Path efficiency: Users are expected to complete evacuation with the shortest path length.
- Congestion avoidance: Congestion should be avoided while following the optimal navigation path.
- Algorithm efficiency: The execution time of the algorithm should be reduced to guarantee real-time navigation.

Our proposed hierarchical emergency navigation algorithm takes into account the above features. In the global navigation phase, a rapid routing algorithm with guaranteed evacuation time is adopted to select the subgoals in the process, and based on the reinforcement learning algorithm, an adaptive navigation strategy is proposed according to the actual situation in the local planning phase.

IV. System Desgin

In this section, we first give the outline of our algorithms, introduce the overall system design, and then detail each step.

A. System overview

As shown in Fig. 2, the implementation of the proposed algorithm mainly involves the following modules:

- Feature graph construction: To provide a basis for the global planner and facilitate it in determining navigation nodes. Based on the real navigation scenario, the set of feature nodes with structural information is extracted. Then, the rapid routing with guaranteed delay bounds algorithm is applied to construct the look-up table so that each feature node possesses two types of delay parameters to the exit.
- Constrained global guidance: To ensure that users reach the destination before deadline and to accelerate the decision process of local emergency navigation. The delay values of the nodes queried in the look-up table must be within the remaining evacuation time.
- Adaptive local navigation: To relieve congestion during the evacuation process and keep users away from hazards. The reinforcement learning method is applied for decision-making in the highly dynamic environment.

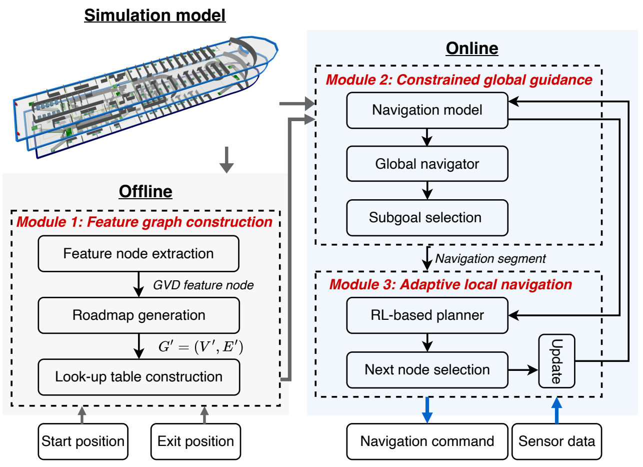


Fig. 2. Overview of hierarchical navigation system.

TABLE I
Notations and Definitions

Notation	Definition
C_s	the set of cells where static obstacles locate
C_f	the set of cells denote obstacle-free space
$G(V,E)$	an undirected graph of navigaiton network
V	the set of navigation nodes
E	the set of segments between adjacent nodes
v_s	the node corresponding to user location
v_e	the node corresponding to exit location
V_d	the set of nodes occupied by other users
V_h	the set of nodes where emergency occurs
$G'(V',E')$	a directed graph with structural features
V'	the set of feature nodes
E'	the set of segments between adjacent feature nodes
$d_T(\overrightarrow{v'_i v'_j})$	the typical delay for corresponding segment $\overrightarrow{v'_i v'_j}$
$d_W(\overrightarrow{v'_i v'_j})$	the worst-case delay for corresponding segment $\overrightarrow{v'_i v'_j}$
D	the evacuation deadline

With notations and definitions presented in Table 1, we will introduce the offline feature graph construction, online emergency navigation, and how we implement the hierarchical emergency navigation system in probable situations.

B. Offline feature graph construction

In this part, we aim to construct a global feature graph, which facilitates the selection of subgoals in the online phase. To begin with, we detail the extraction of the feature nodes and give the definition of the roadmap. On this basis, the look-up table is established on each global feature node to guarantee the evacuation deadline.

1) Feature node extraction: First, we utilize the Generalized Voronoi Diagram (GVD) construction algorithm to extract a set of nodes on the free space which have the same Euclidean distance to the closet obstacles, as shown in Fig. 3a. Let $C_g = \{g_1, \ldots, g_{N_g}\}$ be the set of GVD nodes and constitute the traversable path of global navigation which ensures equidistance to the obstacles and accelerates the process of path searching. Therefore, each GVD node satisfies:

$$\begin{cases} s_{\text{nearest}} = \arg\min_{s_i \in C_s} \|g - s_i\| \\ \text{num}\ (s_{\text{nearest}}) \geq 2 \\ g \in C_f \end{cases} \tag{1}$$

where $C_s = \{s_1, \ldots, s_{N_s}\}$ is the set of nodes corresponding to the static obstacles, and the distance between g_i and its nearest static nodes s_{nearest} is defined as the radius of g_i.

Second, based on the GVD, we further filter the nodes for simplifying path query while reflecting the connections between navigation paths and ensuring the completeness of the navigation network. We denote each GVD node by $g_i = (x_i, y_i, r_i)$, where (x_i, y_i) represents the position of node and r_i represents the shortest distance to the nearest obstacle. We first sort the elements in the set C_g according to r_i and find the node g_{m_i} with the largest r_{m_i}.Then we add g_{m_i} to the set V_{pre} and delete the nodes inside the area with g_{m_i} as the center and r_{m_i} as the radius to reduce the redundancy of navigation nodes. The preliminary filtered GVD nodes are shown in Fig. 3b. In the set V_{pre}, we further extract the nodes with structural information according to the pseudo-code of the feature node extraction in Algorithm 1. Similarly, it starts from the node with the largest radius, identifies its neighboring nodes, and determines the connectivity of its secondary neighbors. After obtaining the set V' shown in Fig. 3c, we check the connectivity among these nodes and denote it as a segment e'_{ij} and the set of segments between navigation node v'_i and its neighboring nodes as e'_i. In this way, we can build a global navigation roadmap which is denoted by a graph model $G' = (V', E')$.

2) Look-up table construction and its implementation: After generating the navigation roadmap, each node knows the connection to its neighbors and the distance between them. To ensure that users can successfully reach the exit before the evacuation deadline, we set two delay parameters on each navigation segment: typical delay $d_T(\overrightarrow{v'_i v'_j})$ and worst-case delay $d_W(\overrightarrow{v'_i v'_j})$, which represent the delay usually encountered based on pedestrian speed and the upper bound for traversing the segment, respectively. Then, based on a rapid routing with guaranteed delay bounds [35], we further construct the look-up table for navigation node query.

At each node v', we establish a 3-tuple table denoted as Tab $[v'] = (d'_v, \pi'_v, \delta'_v)$ to look up the outgoing segment from it. As shown in the Fig. 4, each tuple in the look-up table Tab $[v'_i]$ reflects the outgoing segment of the path from the current node to the exit, and its corresponding typical delay and worst-case delay bound. Specifically, the path from node v' to the exit v_e has the minimum typical delay of δ'_v and the guaranteed worst-case delay d'_v as the upper bound, where π'_v is selected as the next-hop node.

Algorithm 1: Feature Node Extraction Algorithm

```
Input: Navigation scenario, $S$; The set of extracted
       GVD nodes, $V_{pre} = (x, y, r)$;
Output: The set of filtered GVD nodes, $V'$;
1  Initialize $V'$ = $V_{pre}$;
2  for each $g_j \in V_{pre}$ do
3      $m_j = argmax(r_j),\ \forall r_j \in V_{pre}$;
4      set $g'_{m_j} \leftarrow$ neighboring nodes of $g_{m_j}$;
5      for each $g'_{m_j}(i) \in g'_{m_j}$ do
6          calculate two tangent points $t_1, t_2$ of $g_{m_j}$ to the
           circle with center $g'_{m_j}(i)$, radius $r'_{m_j}(i)$;
7          $free_1$ = $freeCheck(g_{m_j}, t_1)$;
8          $free_2$ = $freeCheck(g_{m_j}, t_2)$;
9          if $free_1$ = TRUE and $free_2$ = TRUE then
10             set $g''_{m_j} \leftarrow$ neighboring nodes of $g'_{m_j}$;
11             $count = 0$;
12             for each $g''_{m_j}(j) \in g''_{m_j}$ do
13                 $free_j$ = $freeCheck(g_{m_j}, g''_{m_j}(j))$;
14                 if $free_j$ =FALSE then
15                     Break
16                 end
17                 $count + +$;
18             end
19             if $count = |g''_{m_j}|$ then
20                 remove $g'_{m_j}(i)$ from $V'$;
21             end
22         end
23     end
24 end
25 return $V'$
```

On this basis of look-up table, we utilize an adaptive routing strategy to generate the global guidance, ensuring that users can reach the destination from their initial position within the deadline, while typical delays are minimal. Considering the evacuation deadline D, we use the example in Fig. 4 to illustrate the implementation of the look-up table in the global phase. The initial position of the user corresponds to the global navigation node v'_u. According to the 3-tuple look-up table of v'_u, the alternative next-hop navigation nodes include v'_1 and v'_e. Being aware of the evacuation deadline, we first compare it with the guaranteed worst-case delays of 25 and 30 for v'_1 and v'_e, respectively. Let us take into account the following case:

- If $D \geq 30$: The user is firstly guided to v'_1. Then considering the actual delay for traversing $\overrightarrow{v'_u v'_1}$, if $D - d_{\overrightarrow{v'_u v'_1}} \geq 25$, then the user will be navigated to v'_2 and v_e sequentially. If $15 \leq D - d_{\overrightarrow{v'_u v'_1}} < 25$, the user will be

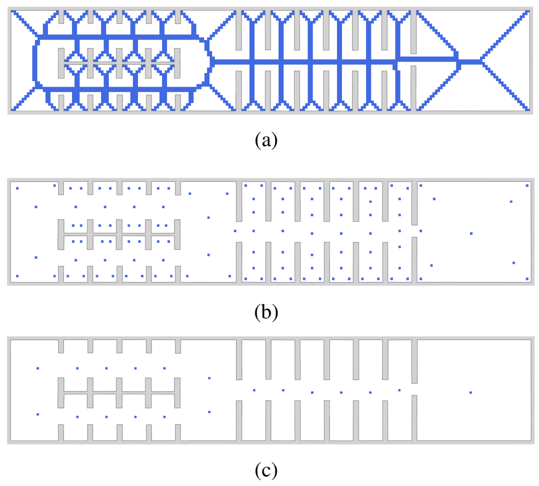
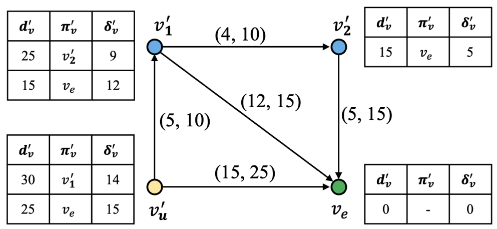


Fig. 3. Process of feature node extraction. (a) Grid-based GVD of a ship indoor environment. (b) Preliminary GVD nodes. (c) Extracted structure-aware feature nodes.



d'_v	π'_v	δ'_v
25	v'_2	9
15	v_e	12

d'_v	π'_v	δ'_v
15	v_e	5

d'_v	π'_v	δ'_v
30	v'_1	14
25	v_e	15

d'_v	π'_v	δ'_v
0	-	0

Fig. 4. A simple example of look-up table to the exit v_e.

navigated to v_e directly.

- If $25 \leq D < 30$: Then the user will be navigated to v_e directly.
- If $D < 25$: The user will not be able to safely complete the evacuation.

C. Online emergency navigation

In this part, we will introduce the adaptive emergency navigator based on the DC-HEN algorithm. We start by defining the essential components of the reinforcement learning method, and then detail the training process with the neural network structure.

1) RL-based emergency navigation: In order to independently generate safe navigation instructions for each user while considering congestion, we propose to train the reinforcement learning agent in an emergency evacuation environment assisted by LPWAN to find the optimal navigation solution.

Local observation: Based on the previously mentioned undirected graph $G = (V, E)$, we simulate the evacuation environment as a grid map and each grid cell reflects the environmental information. In particular, the environment state is replaced by the local observation with a limited range $H_o * W_o$. Taking the agent as the center, the surrounding environmental information and global guidance are collected at each time step t. We denote the local observation of agent at time t as $O_t = \left\{o_t^f, o_t^w, o_t^d, o_t^h, o_t^g\right\}$, which collects the set of locations of free space, walls, other users, hazards, and global guidance segments within the observation range respectively.

Action space: In this reinforcement learning task, an agent takes an action on the behalf of the user at each time step and move from current node to a neighboring navigation node. Therefore we restrict the action space as a discrete set, which is defined as: $A = \{a_1, a_2, \dots, a_9\}$.

Reward function: In our proposed navigation system, to ensure that the user reaches the exit safely and efficiently, the RL agent in the local path planner needs to meet the following requirements simultaneously: 1) avoiding collide with walls and congestion with other users; 2) following the global optimal guidance as possible; 3) keeping away from hazardous areas. Then we design a novel reward function as follows: 1) a small negative reward at each time step to encourage the agent to reach the exit with less time compensation; 2) a penalty of when the agent collides with walls or other users; 3) a great penalty for exposure to hazards; 4) a positive reward denoted as $r_4 = N_t \times 10$ for following the global guidance. We first compute the parameter N_t once the agent returns to the global navigation path at time t, which is obtained by subtracting the sequence number of the currently occupied global navigation cell from the total length of the remaining global reference path. Then we remove the current cell and the cells after it on the global navigation sequence; 5) a great positive reward when the agent reaches the exit.

Double DQN: Q-learning bases its solution on temporal difference and stores the updated Q-values in a table after each step until convergence. However, Q-learning cannot cope with complex environments with high dimensional state-action spaces. Thus, Deep Q network (DQN) [36] is proposed, which combines the exploration mechanism of reinforcement learning with the deep neural networks. In particular, the experiences explored by the agent are stored in the replay memory and a Q-network is then used to approximate the Q-value function, where the network takes the state as the input and outputs the Q-value of each state-action pair. To update the parameters of the network, a minibatch of transitions (s_t, a_t, r_t, s_{t+1}) is randomly sampled and at iteration i the loss function is as follows:

$$L_i(\theta_i) = \mathbb{E}[(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i^-) - Q(s_t, a_t; \theta_i))^2] \tag{2}$$

where r_t can be calculated according to the reward function, γ is the discount factor.

Based on DQN, Double DQN (DDQN) [17] is proposed to solve the problem of overestimation by decoupling action selection and evaluation using two networks. Firstly, two network models with the same structure are constructed. An action with maximum Q-value is selected by current Q-network, while the target Q-network calculates its value, and the network parameters are denoted as θ and θ' respectively.

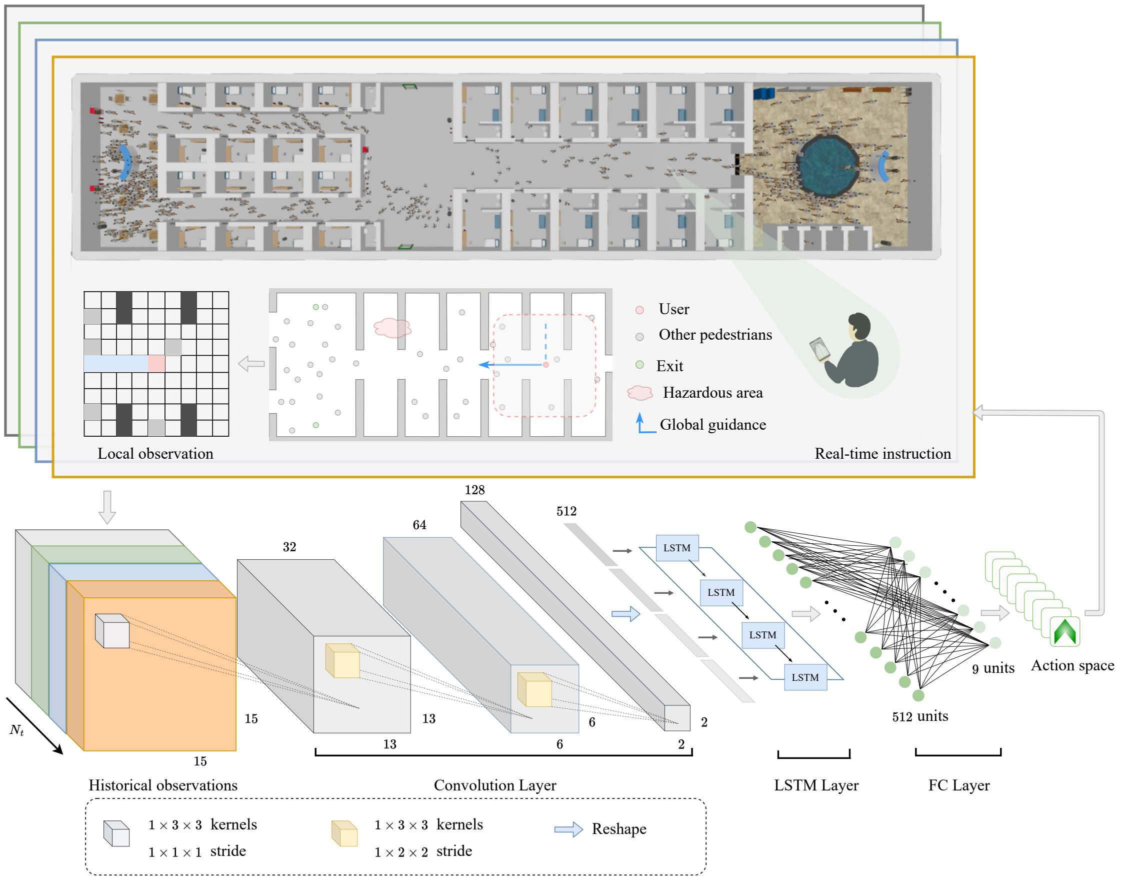


Fig. 5. The neural network structure of DC-HEN.

And the target Q-value is denoted as:

$$Y_t^{DoubleDQN} = r_t + \gamma Q(s_{t+1}, \arg\max_{a_t} Q(s_{t+1}, a_t; \theta); \theta') \tag{3}$$

In this paper, we use the DDQN algorithm and adapt it to the proposed emergency navigation objectives. During each training episode, the agent continues to explore and takes the action from the action space, and the observations update at each time step. We center the observation image on the user's current location, with a range of $H_o \times W_o$. And the black, red, and gray cells in each frame indicate the location of walls, hazards, and other pedestrians, respectively. In particular, we provide global guidance for the agent, as represented by the segments composed of blue cells. Then the sequential frames of observations are stacked to compose the input of the network as shown in Fig. 5. Based on the DDQN algorithm, multiple layers of 3D CNN are used in this network to extract image features, which are subsequently connected to an LSTM layer for temporal information. And finally, a fully connected layer outputs the Q-value corresponding to each state-action pair. The exploration will continue until the completion of K episodes. For training the target Q-network, we store the tuple $< s_t, a_t, r_t, s_{t+1} >$ into the replay memory set with the size N_r, then extract N_b memories for updating the current Q-network, and the loss function can be written as:

$$L(\theta) = \frac{1}{N_b}\sum_{i=1}^{N_b}[Y_t^i - Q(s_t^i, a_t^i; \theta)]^2 \tag{4}$$

To update the parameters θ', θ is moved to the target Q-network every U time steps. In the real-time navigation process, users can be directed to the optimal next-hop navigation node based on the output value of the target Q-network, as described in the pseudo-code of Algorithm 2.

D. Hierarchical emergency navigation strategy

In this part, we will detail the process of the proposed hierarchical navigation strategy which directs users to follow the global reference path safely while avoiding congestion.

Based on the graph model $G = (V, E)$, where each node can embody the real-time environmental information, we design an emergency navigation system with a hierarchical mechanism. In the global phase, the obtained sequence of feature nodes guarantees that the user reaches the exit safely before deadline. Then based on the DC-HEN algorithm, the trained target network model outputs the action with the highest value based on the real-time observations, which is then used to send navigation instructions through the device carried by the user. Thus when the user reaches the exit, we can obtain a

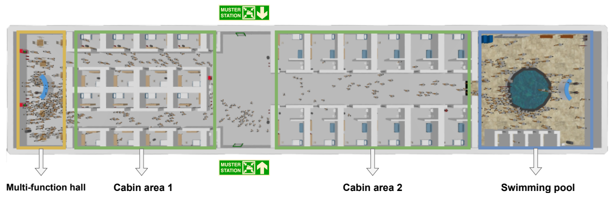
Algorithm 2: Locally RL-based Emergency Navigation Algorithm

```
Input: User node, $v_u$; Subgoal $v'$; Target Q network model;
Output: Real-time evacuation instruction, $v_i$; Traversal time, $d_t$;
1  Load the trained target Q network model;
2  Get the sequence of global reference nodes:
     $p_{v_u v'} \leftarrow \{v_u, \dots, v'\}$;
3  Remove hazardous nodes from $p_{v_u v'}$:
     $p_{v_u v'} \leftarrow p_{v_u v'} - p_{v_u v'} \cap V_h$;
4  Input the stacked observations:
     $s_t \leftarrow \{O_t, O_{t-1}, \dots, O_{t-N_t-1}\}$;
5  $d_t$ = 1;
6  while True do
7      $a_t \leftarrow \arg\max Q(s_{t+1}, a_t; \theta')$;
8      $v_i \leftarrow$ the navigation node corresponding to the execution of action $a_t$;
9      $d_t$ = $d_t$ +1;
10     if $v_i \in p_{v_u v'}$ then
11         remove set $\{v_u, \dots, v_i\}$ from $p_{v_u v'}$;
12     end
13     Update $v_u$;
14     if $p_{v_u v'}$ is empty then
15         Break
16     end
17 end
18 return $d_t, v_i$;
```

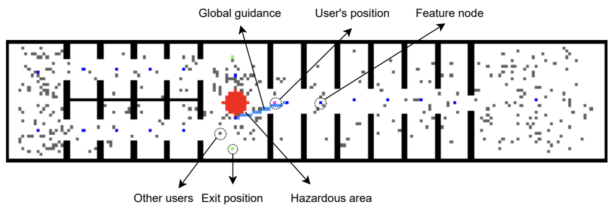
congestion-relieved safe path consisting of the experienced nodes, denoted as $p_{se} = \left\{v_s, \dots, v_i, v'_j, \dots, v_e\right\}$.

Depending on the node where user's initial position is located, we consider the following three situations, respectively.

- Situation 1: The initial user node is a global feature node. In this situation, the algorithm queries the look-up table of the current node directly and Algorithm 2 is then executed to navigate the user to the next feature node. And the actual delay to experience each segment is subtracted from the total evacuation time, which in turn updates the remaining deadline. And the above operation is looped until the user reaches the exit node.
- Situation 2: The initial location of the user corresponds to a normal node in the free space. In this case, we first determine the global node v'_i corresponding to the feature area occupied by the user, and calculate the shortest path $p_{si} = \{v_s, \dots, v'_i\}$ from the user's position to this node. Then we combine the global guidance reaches out from v'_i based on the look-up table with p_{si} as the first segment of the global reference path. The subsequent execution will be implemented according to the situation 1.
- Situation 3: The initial user node is surrounded by hazards. In this situation, there is no accessible navigation path and the user is expected to wait for rescue.



(a)



(b)

Fig. 6. A single-deck crowd evacuation simulation model: (a) 3D view of the simulation process in Anylogic. (b) Training environment based on scenario (a).

V. Experimental Studies and Results

A. Data set generation

Before the experimental setup, to make the experiment more realistic, we first simulate the evacuation process on a single deck using the visualization simulation platform Anylogic and generate the crowd movement data set. The scenario is modeled according to the single-deck layout of a real cruise ship. Then the initial locations of simulation agents are randomly generated according to the proportion of different service areas, and the movement logic of agents is constructed according to the existing evacuation plan. More details of the crowd evacuation modeling are as follows:

Agent attributes: Following IMO MSC Circ 1238 [37], the agents in the model are set up according to the age, gender, and travel speed described in the regulations.

Start locations: The start locations of the agents are mainly distributed in four parts of the deck, such as Multi-function hall, Cabin area 1, 2, Swimming pool. The specific locations of each service area are shown in the Fig. 6a.

End locations: The target locations of crowd evacuation is the muster station on each deck and there are two muster stations A and B.

Response time distribution: Referring to the response time distribution (RTD) defined in [37], the agents RTDs in our experiment is set based on four distributions as shown in Fig. 7.

B. Experimental setup

To validate the proposed hierarchical navigation strategy, we set up a ship indoor simulation environment as shown in Fig. 6b. Then we set the destinations of navigation in the map according to the locations of the real muster stations. Considering the unpredictable emergencies, we randomly set the hazardous area, and we extend its edges to form a protection zone in order to prevent pedestrians from reaching the hazards. With the aforementioned crowd movement data set,

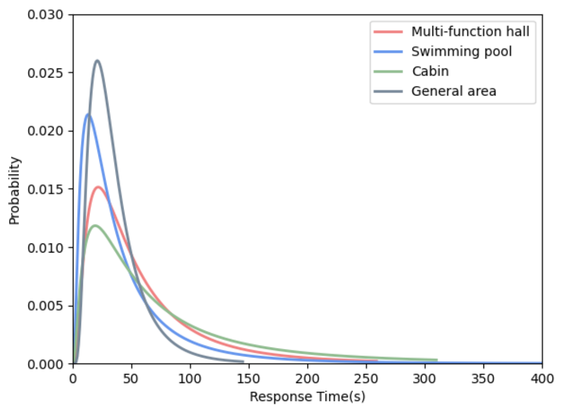


Fig. 7. Response time distribution of different service area.

TABLE II
MODEL AND TRAINING PARAMETERS

Parameters	Configuration
Observation field	$H_o = W_o = 15$
Episode	$k = 30000$
Replay memory size	$N_r = 10000$
Batch size	$N_b = 64$
Update frequency	$U = 20$

the training environment and the navigator's decision update once per second. On this basis, the navigation goal is to guide the user from the initial position to the muster station. In the experiments, the default parameters are set based on Table 2. The open-source code as well as the crowd movement data set will be released at https://github.com/xlz0011/DC-HEN. Training was performed on a NVIDIA GTX 2080ti GPU in Python 3.8 with TensorFlow 2.3.0.

C. *Results and performance evaluation*

In order to determine the local observation size and the length of the input historical sequence in the experiment, we use the generated data set of different populations for training. As shown in Table 3, with the increase in observation size, the navigation path stretch decreases. When the observation field reaches a certain size, such as 15×15, the path stretch tends to be stable. Then, in the case of fixed local observations, we study the impact of input sequence length on the results. As shown in Table 4, with the increase of the sequence length, the temporal information becomes richer and the navigation path stretch decreases. When $N_t >= 4$, there is no significant improvement in path stretch performance.

To evaluate the training process of our proposed hierarchical navigation strategy, we first compare it with the DDQN algorithm in the same environment, which lack global guidance. Then, simulations are conducted using the trained model and results are compared with other congestion avoidance navigation algorithms.

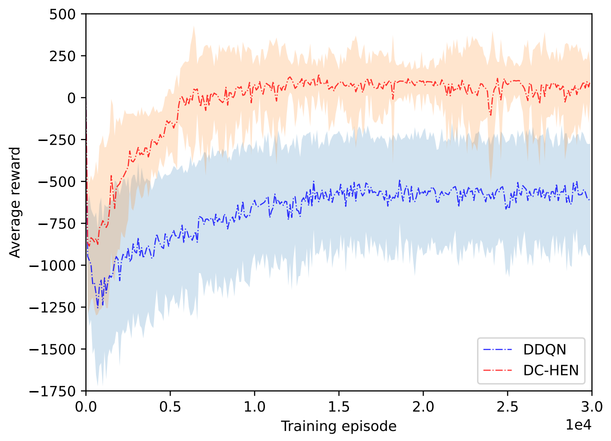


Fig. 8. Comparison of training returns of different algorithms.

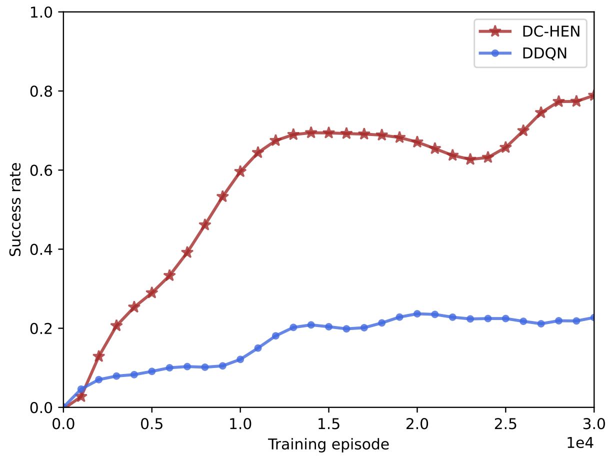


Fig. 9. Comparison of success rate of different algorithms.

1) *Evaluation of the reward returns:* We first perform 30000 episodes of training based on the DDQN and DC-HEN in the OpenAI gym environment, and we score each algorithm by calculating the average reward over the last 100 episodes. The training results are shown in the Fig. 8, the red line represents the DC-HEN algorithm and the blue line represents the average reward of the DDQN algorithm. From Fig. 8, the average rewards obtained by the proposed algorithm decreases rapidly in the early stage of training with the novel reward function. After 1500 episodes, the red curve shows an increasing trend and the DC-HEN algorithm converges after 10000 episodes of training. In comparison, in the case of large-scale training environment with sparse reward values, the training curve of the DDQN algorithm changes slowly, and the average reward value after convergence is lower than our proposed method.

2) *Evaluation of the simulation results:*

- Navigation success ratio: We also count the number of times when the agent reaches the target every 1000 episodes during the training process. As shown in the Fig. 9, DC-HEN's navigation success rate rises rapidly with

TABLE III
ABLATION STUDY ON DIFFERENT OBSERVATION SIZES

$H_o \times W_o$	9×9	11×11	13×13	15×15	17×17
Crowd-50	1.12	1.10	1.08	1.08	1.07
Crowd-100	1.25	1.24	1.18	1.12	1.10
Crowd-150	1.44	1.37	1.25	1.15	1.15

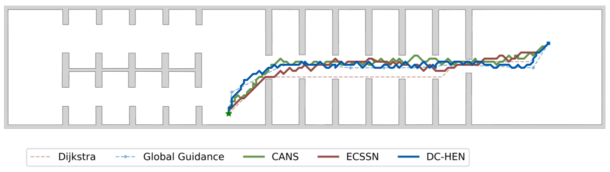


Fig. 10. Possible evacuation trajectories by different methods.

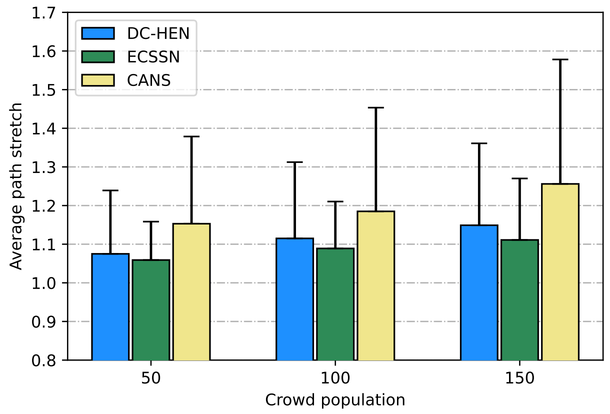


Fig. 11. Average path stretch.

the increase of training times and finally reaches 78.3%. In contrast, the curve of DDQN algorithm changes slowly and the navigation success rate after convergence is low.

- Average path stretch: For the same ship indoor environment, we implement three approaches with different crowd data sets, which are set to 50, 100, and 150 respectively. Each set of comparisons contains 100 pairs of data with a randomly generated user's initial location, total traverse time, and navigation trajectory. As shown in Fig. 10, we show the possible evacuation trajectories for a selected user generated by three methods in a large-scale ship indoor environment. Compared to ECSSN, which follows the shortest path algorithm, the route planned by DCHEN has a certain distance from the static obstacles, while the trajectory of ECSSN is close to the wall. And the path stretch is calculated as the actual navigation path length divided by the length of the shortest possible path from the user's start location to the exit. Fig. 11 shows the average and the maximum path stretch of DC-HEN, CANS, and ECSSN with different crowd movement data sets in a ship indoor environment. It shows that the path stretch results for DC-HEN are similar to those of ECSSN in all cases. ECSSN always achieves the lowest path

TABLE IV
ABLATION STUDY ON DIFFERENT INPUT SEQUENCE LENGTHS

N_t	1	2	3	4	5
Crowd-50	1.28	1.19	1.12	1.08	1.06
Crowd-100	1.43	1.33	1.27	1.12	1.10
Crowd-150	1.47	1.36	1.28	1.15	1.15

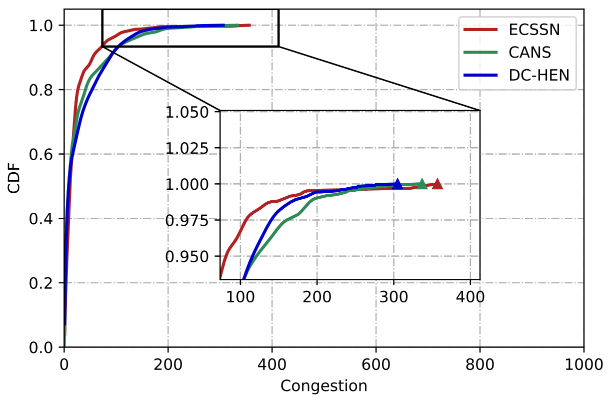


Fig. 12. Congestion distribution.

stretch rate since it follows the path which is determined by the minimum distance from any location to the exit. The results demonstrate that the path efficiency of our method approaches that of the shortest path algorithm.

- Congestion distribution: We further evaluate the performance of congestion distribution of our navigation algorithm by measuring the number of paths related to the nodes involved in the navigation environment at the end of each execution (100 users for 10 times test). As is shown in Fig. 12, nodes involved in DC-HEN are at most participate in about 300 navigation paths, and the curve quickly reaches 1. Compared with CANS and ECSSN, DC-HEN involves the least navigation paths and has better congestion mitigation performance in the navigation process.

VI. CONCLUSIONS

This paper developed a hierarchical emergency navigation algorithm - DC-HEN, involving both congestion and deadline awareness for ship indoor environments. Considering the structure of the navigation scenario, we extract the feature node and construct a look-up table based on it. This allows us to quickly determine the fastest route within the evacuation deadline. Referring to the global guidance, DC-HEN utilizes reinforcement learning and designs a novel reward function to provide congestion-relieved evacuation guidance for each user in real-time. The proposed DC-HEN outperforms other state-of-the-art methods in the case study. It showed that DC-HEN has a higher success rate with 78.3%, relatively short average path stretch, and better congestion avoidance performance. Additionally, DC-HEN can be easily deployed on emergency navigation for ship indoor environments due to its high training efficiency (i.e., fast convergence) and friendly assistance. In the future, our algorithm has the potential to be extended to other complicated situations, such as finite life-saving resources and muster station with limited capacity.

References

[1] J.-U. Schröder-Hinrichs, E. Hollnagel, and M. Baldauf, "From titanic to costa concordia—a century of lessons not learned," *WMU journal of maritime affairs*, vol. 11, no. 2, pp. 151–167, 2012.

[2] T.-e. Kim, S. Nazir, and K. I. Øvergård, "A stamp-based causal analysis of the korean sewol ferry accident," *Safety science*, vol. 83, pp. 93–101, 2016.

[3] X. Wang, Z. Liu, J. Wang, S. Loughney, Z. Zhao, and L. Cao, "Passengers' safety awareness and perception of wayfinding tools in a ro-ro passenger ship during an emergency evacuation," *Safety science*, vol. 137, p. 105189, 2021.

[4] S. H. Shah and I. Yaqoob, "A survey: Internet of things (iot) technologies, applications and challenges," *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pp. 381–385, 2016.

[5] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *ieee communications surveys & tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[6] J. de Carvalho Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. Aquino, "Lorawan—a low power wan protocol for internet of things: A review and opportunities," in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*. IEEE, 2017, pp. 1–6.

[7] J. Xu, J. Yao, L. Wang, Z. Ming, K. Wu, and L. Chen, "Narrowband internet of things: Evolutions, technologies, and open issues," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1449–1462, 2017.

[8] J. Purohit, X. Wang, S. Mao, X. Sun, and C. Yang, "Fingerprinting-based indoor and outdoor localization with lora and deep learning," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[9] M. Barnes, H. Leather, and D. Arvind, "Emergency evacuation using wireless sensor networks," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*. IEEE, 2007, pp. 851–857.

[10] M. Li, J. Wang, Z. Yang, and J. Dai, "Sensor network navigation without locations," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 391–392.

[11] C. Wang, H. Lin, and H. Jiang, "Cans: Towards congestion-adaptive and small stretch emergency navigation with wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 5, pp. 1077–1089, 2015.

[12] L.-W. Chen and J.-J. Chung, "Mobility-aware and congestion-relieved dedicated path planning for group-based emergency guiding based on internet of things technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2453–2466, 2017.

[13] Z. Zhang, H. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, "Congestion-aware evacuation routing using augmented reality devices," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2798–2804.

[14] A. Jindal, V. Agarwal, and P. Chanak, "Emergency evacuation system for clogging free and shortest-safe path navigation with iot-enabled wsns," *IEEE Internet of Things Journal*, 2021.

[15] Z. Li, X. Liu, and S. Wu, "Dynamic emergency navigation based on prediction via wireless sensor networks," *2020 The 8th International Conference on Information Technology: IoT and Smart City*, 2020.

[16] A. Borshchev, S. Brailsford, L. Churilov, and B. Dangerfield, "Multi-method modelling: Anylogic," *Discrete-event simulation and system dynamics for management decision making*, pp. 248–279, 2014.

[17] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[19] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[20] S. M. Persson and I. Sharf, "Sampling-based a* algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.

[21] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent unmanned ground vehicles*. Springer, 1997, pp. 203–220.

[22] B. Rivière, W. Hönig, Y. Yue, and S.-J. Chung, "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4249–4256, 2020.

[23] J. Zeng, L. Qin, Y. Hu, Q. Yin, and C. Hu, "Integrating a path planner and an adaptive motion controller for navigation in dynamic environments," *Applied Sciences*, vol. 9, no. 7, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/7/1384

[24] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.

[25] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 500–505.

[26] G. Li, A. Yamashita, H. Asama, and Y. Tamura, "An efficient improved artificial potential field based regression search method for robot path planning," in *2012 IEEE International Conference on Mechatronics and Automation*. IEEE, 2012, pp. 1227–1232.

[27] S. Zhou, X. Liu, Y. Xu, and J. Guo, "A deep q-network (dqn) based path planning method for mobile robots," in *2018 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2018, pp. 366–371.

[28] Z. Zhang, D. Lu, J. Li, P. Liu, and G. Zhang, "Crowd evacuation simulation using hierarchical deep reinforcement learning," in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2021, pp. 563–568.

[29] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[30] Y. Zhang, Z. Chai, and G. Lykotrafitis, "Deep reinforcement learning with a particle dynamics environment applied to emergency evacuation of a room with obstacles," *Physica A: Statistical Mechanics and its Applications*, vol. 571, p. 125845, 2021.

[31] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[32] J. Sharma, P.-A. Andersen, O.-C. Granmo, and M. Goodwin, "Deep q-learning with q-matrix transfer learning for novel fire evacuation environment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 12, pp. 7363–7381, 2020.

[33] X. Yiran, W. Rui, and L. Jiafeng, "Crowd evacuation guidance based on combined action-space deep reinforcement learning," *Harbin Gongye Daxue Xuebao/Journal of Harbin Institute of Technology*, vol. 53, no. 8, 2021.

[34] Q. Li, M. De Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 313–325.

[35] S. Baruah, "Rapid routing with guaranteed delay bounds," in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2018, pp. 13–22.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[37] "Guidelines for evacuation analysis for new and existing passenger ships." IMO MSC/Circ 1238, 2007.