Intelligent Service Robotics (2024) 17:401–418
https://doi.org/10.1007/s11370-024-00537-2

ORIGINAL RESEARCH PAPER

MAP3F: a decentralized approach to multi-agent pathfinding and collision avoidance with scalable 1D, 2D, and 3D feature fusion

Marzie Parooei[1] · Mehdi Tale Masouleh[1] · Ahmad Kalhor[1]

Received: 7 October 2023 / Accepted: 16 March 2024 / Published online: 22 April 2024

Abstract
Path planning and collision avoidance are vital aspects of successful development and utilization of robots in complex and multi-agent environments. With the integration of robots into social settings, the significance of this issue becomes more apparent. This paper introduces a decentralized management approach based on deep reinforcement learning, where each agent learns independently based on its local observations. The proposed method employs a feature fusion technique which combines 1D, 2D, and 3D features. In order to streamline computation and optimize the training process, an established separation index method is utilized. This approach strategically selects a subset of the most informative features. The presented approach outperforms classical and learning-based methods in various environments with differing densities. Performance evaluation metrics include the interaction index, which indicates the percentage of collision-free scenarios, the reachability index, measuring the time for the slowest agent to reach its goal, the field of view index, demonstrating reduced computation time by narrowing the field of view without compromising interaction, and the scalability index, quantitatively measuring a system's capability to efficiently handle increasing amounts of work or its ability to be enlarged to accommodate that growth. The performance of this method, compared to PRIMAL, ORCA, and ODRM* methods, has shown an increase of over 30% in situations where the environment is more complex and the number of agents is higher.

Keywords Multi-agent · Deep reinforcement learning · Scalability · Path finding · Feature fusion

1 Introduction

In the realm of multi-agent systems, the pervasive challenge of pathfinding and collision avoidance has emerged as a foundational concern, attracting the attention of researchers across various domains. This challenge has notably captivated experts in fields such as automated guided vehicles, autonomous mobile robots, quadcopter robots, and quadruped robots [1–4]. The primary objective is to equip agents with the ability to navigate through a common environment while simultaneously avoiding collisions with other agents and obstacles. This challenge is pertinent to a wide range of applications, including robotics, transportation systems, and video game development [5, 6].

Multi-agent pathfinding (MAPF), a subset of multi-agent systems (MAS), focuses on delineating collision-free paths for multiple agents, taking them from initial positions to predefined goals. The complexity of MAPF is particularly pronounced in densely populated environments where agents should harmonize their movements, further compounded by diverse sensing capabilities, distinct movement speeds, and varying objectives [7, 8].

The data-driven approach begins with the concept of deep learning, a subset of the machine learning domain that employs neural networks to analyze and learn from large datasets [9]. This technique enables the development of sophisticated algorithms capable of handling complex tasks, such as robot motion planning. Robots can utilize deep learning to extract patterns and insights from extensive datasets, allowing them to generalize from past experiences. Building on this, reinforcement learning (RL) emerges as a specialized field of machine learning where agents learn to make

✉ Mehdi Tale Masouleh
m.t.masouleh@ut.ac.ir

Marzie Parooei
parooie@ut.ac.ir

Ahmad Kalhor
akalhor@ut.ac.ir

1 Human and Robotic Interaction Laboratory, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

decisions based on local observations through interactions with their environment, as outlined in [10]. RL is particularly effective in scenarios where decision-making is sequential and dependent on the state of the environment. This learning process makes RL a popular technique in multi-agent path finding (MAPF) algorithms, as noted in [11].

Deep reinforcement learning (DRL) combines the depth and complexity of deep learning with the decision-making process of RL. This hybrid approach is adept at tackling intricate tasks in robot motion planning, especially in complex environments populated by numerous agents, as demonstrated in [12]. By integrating the robust pattern recognition capabilities of deep learning with the dynamic decision-making strategies of RL, DRL equips robots with enhanced adaptability and efficiency in diverse and challenging scenarios.

An innovative framework called Pathfinding via Reinforcement and Imitation Multi-Agent Learning (PRIMAL) [12], for MAPF is introduced which combines RL and imitation learning to train both fully centralized and decentralized policies. This approach [12] allows agents to reactively plan paths online in a partially observable world while exhibiting implicit coordination. PRIMAL, in environments characterized by low obstacle density, agents can maneuver around each other with relative ease. However, the performance of such systems can diminish in denser environments where coordinated actions between agents are required to reach their goals, often necessitating significant alterations in their paths. The training of policies in PRIMAL requires a large amount of data, which is a common challenge in RL-based approaches. Nevertheless, decentralized MAPF algorithms, such as those proposed in PRIMAL, have been shown to be effective in complex environments with a large number of agents [12]. The framework [13] leverages imitation learning, a prominent paradigm in machine learning and robotics, which involves training an agent to acquire a desired behavior or skill by observing and mimicking the actions of an expert demonstrator. This method [13] leverages the wealth of knowledge embodied in expert demonstrations, enabling the agent to learn complex tasks and make informed decisions in a manner akin to how humans acquire skills through observation and mimicry. Imitation learning [13] typically employs techniques such as behavioral cloning and inverse reinforcement learning to model the expert's actions and underlying decision-making processes. This learning paradigm finds applications in a wide range of domains, including autonomous driving, robotic manipulation, and natural language processing, where it facilitates the rapid acquisition of valuable expertise and can significantly enhance the performance of autonomous systems [13]. As such, PRIMAL represents a significant advancement in MAPF, energizing the strengths of RL and imitation learning to provide a robust solution for navigating multi-agent systems in complex environments.

Traditional $Q(\lambda)$-learning algorithm is enhanced through the incorporation of the obstacle area expansion strategy, as described in [14]. The resulting algorithm, termed as OAE -$Q(\lambda)$-learning, is applied to address path planning challenges in complex environments. The contributions of OAE -$Q(\lambda)$-learning are as follows. Primarily, the concave obstacle area within the environment is expanded, aimed at preventing repetitive erroneous actions when the agent becomes positioned within the obstacle zone. Secondarily, the extended obstacle area is eliminated, resulting in a reduction in the dimensionality of the learning state space. This reduction subsequently leads to an acceleration in the rate of algorithmic convergence. Empirical investigations substantiate the effectiveness and feasibility of the OAE -$Q(\lambda)$-learning technique for path planning within intricate environmental settings. Exploration of DRL methods for multi-agent domains is presented in [15]. The authors discuss the challenges faced by traditional algorithms in multi-agent scenarios, highlighting that Q-learning struggles with the environment's inherent non-stationarity, and policy gradient methods grapple with escalating variance as the number of agents increases. Subsequently, they introduce a modified actor-critic approach that incorporates the action policies of other agents, enabling the successful learning of policies necessitating intricate multi-agent coordination. The methodology presented in [15] undergoes evaluation across a variety of benchmark tasks, with its performance juxtaposed against other leading-edge approaches. The outcomes indicate that this method surpasses its counterparts in both learning velocity and ultimate efficacy. This study furnishes valuable perspectives on the application of DRL for tackling intricate issues in multi-agent environments, offering a beneficial resource for both researchers and practitioners engaged in related fields. The work presented in [16] introduces a decentralized multi-agent collision avoidance algorithm based on a novel application of DRL. This approach effectively offloads the online computation, necessary for predicting interaction patterns, to an offline learning procedure.

In this study, a significant contribution is made in the field of MAPF by introducing a decentralized approach employing DRL. This novel approach empowers agents to independently make decisions based on local observations, thus eliminating the need for centralized control. Each agent is outfitted with a camera which can rotate to capture images from a multitude of angles. By amalgamating these images, a composite is constructed which emulates the perspective and information a top-view image would offer. This approach ensures that each agent can independently form a detailed understanding of its surroundings, facilitating a truly decentralized mode of operation. The proposed method enhances

Table 1 Comparative overview of key previous methods from multiple perspectives

Reference	Centralized/decentralized	Continuous/discrete	Scalable/not Scalable	Field of view	Single/multi-Target	Uncertainty acceptance
[17]	Decentralized	Continuous	Scalable	Full	Multi	Very low
[18]	Centralized	Discrete	Not scalable	Full	Multi	No
[19]	Centralized	Continuous	Not scalable	Full	Multi	No
[15]	Hybrid	Continuous	Not scalable	Full	Single	Yes
[16]	Centralized	Continuous	Not scalable	Full	Multi	Yes
[20]	Hybrid	Continuous	Not scalable	Full	Multi	Yes
[21]	Centralized	Discrete	Not scalable	Full	Multi	No
[22]	Centralized	Continuous	Scalable	Full	Single	Yes
[23]	Hybrid	Continuous	Not scalable	Full	Multi	Yes
[12]	Decentralized	Discrete	Scalable	partially	Multi	Yes
The proposed method	Decentralized	Discrete	Scalable	partially	Multi	Yes

path planning accuracy and integrates 1D, 2D, and 3D features, making it adaptable to environments with varying density levels. An in-depth analysis of ten influential works in the field is conducted, as outlined in Table 1, highlighting aspects such as centralization, scalability, environmental characteristics, field of vision, objective types, and the capacity to handle uncertainty.

The remainder of this paper is organized as follows. Section 2 provides a general description of the proposed method, referred to as MAP3F. In order to the details of the proposed method, it is divided into two subsections: the network architecture (Sect. 2.1) and feature selection (Sect. 2.2). Section 2.2 discusses the selection of important features using the separation index (Sect. 2.2.1) and the multi-head attention layer (Sect. 2.2.3) methods. Section 3 introduces the methodologies employed for goal assignment to agents (Sect. 3.1), examines and assesses scenarios generated by the presented approach, and conducts a comparative analysis against previously established techniques using four distinct indices, namely the interaction index (Sect. 3.2), the reachability index (Sect. 3.3), the field of view index (Sect. 3.5), and scalability index (Sect. 3.4). Finally, Sect. 4 provides a concise summary of the findings and outlines potential avenues for future research.

2 Multi-agent pathfinding with feature fusion method

The multi-agent pathfinding with feature fusion (MAP3F) method presents a decentralized DRL method in which each agent acts as an independent decision-making unit with a limited field of view. This model is generally trained using DRL, but to accelerate the training process, the initial weights are not considered randomly. Using a dataset prepared from previous methods in this field, the network is first trained in a supervised manner, and its weights are then used as the initial weights for DRL. The general architecture and stages of the proposed method are described in this section. The overall process presented in this paper is shown in Fig. 1, and the proposed stages consist of four main parts.

In the first part, a dataset is prepared using successful previous methods which is shown in Fig. 2. As it can be observed from Fig. 3, obstacles are indicated by hatched cells and goals are represented by destination icons in different colors for each agent. Each agent is identified by a specific color, which corresponds to the goal assigned to that agent. In order to collect data, ORCA [17], ODRM* [24], and PRIMAL [12] methods were used. In each scenario, agents are assigned distinct objectives based on the strategies outlined by these methods. Subsequently, these algorithms engage in path planning for the agents, capturing an image of the environment's top view at each timestep. This process is systematically repeated for 1000 different scenarios, ensuring a diverse and comprehensive dataset is accumulated. In the preprocessing phase, a deliberate strategy was employed to label the data. Acknowledging the importance of a discrete and streamlined representation for efficient decision-making, the environment was conceptualized as discrete. It was assumed that each agent could move in one of four principal directions: up, down, left, or right, or choose to stay still. This simplification is compatible with the capabilities of the ORCA, ODRM*, and PRIMAL methods and is crucial for enhancing the efficiency of the decision-making process. The data were carefully organized based on the subsequent action chosen by each agent. For instance, if an agent opts to move 'up' in the next step, that particular instance is classified as 'up.' This systematic approach to classification is uniformly applied across all possible actions, resulting in the data being sorted into five distinct categories: up, down, left, right, and stop.

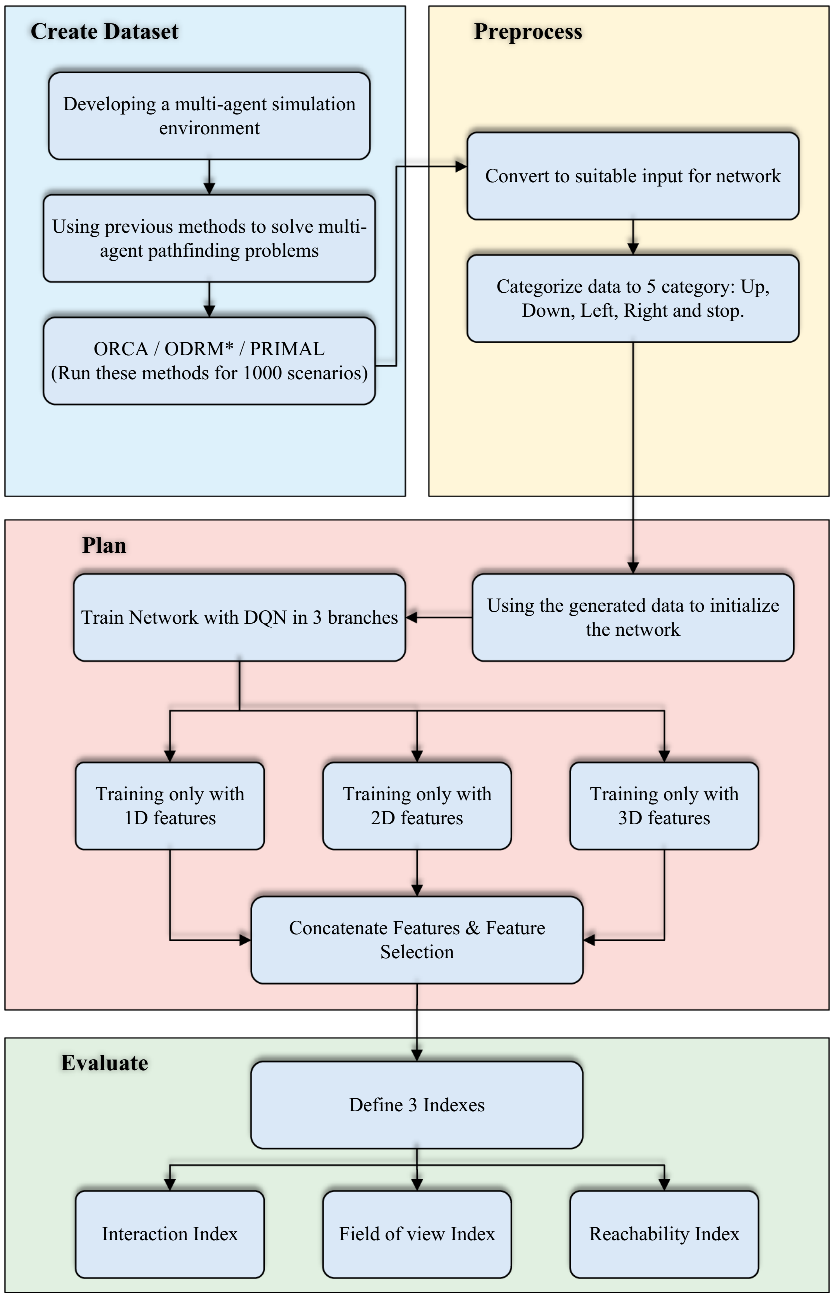
Create Dataset
Developing a multi-agent simulation environment
Using previous methods to solve multi-agent pathfinding problems
ORCA / ODRM* / PRIMAL (Run these methods for 1000 scenarios)
Preprocess
Convert to suitable input for network
Categorize data to 5 category: Up, Down, Left, Right and stop.
Plan
Train Network with DQN in 3 branches
Using the generated data to initialize the network
Training only with 1D features
Training only with 2D features
Training only with 3D features
Concatenate Features & Feature Selection
Evaluate
Define 3 Indexes
Interaction Index
Field of view Index
Reachability Index

Fig. 1 Overall workflow of the proposed method, the so-called MAP3F

In the third stage, a combined network is presented for training the data. This network consists of three separate branches. Each branch extracts one-dimensional (1D), two-dimensional (2D) and three-dimensional (3D) input image features. After combining all features, to the end of selecting important and influential features accurately and remove features which have repetitive information and make the network complex, even in complex scenarios which sometimes cause confusion for the agent in decision-making, two feature selector functions have been used whose process is fully explained in Sects. 2.2.1 and 2.2.3. In the final stage, the proposed method is evaluated using three defined indicators.

2.1 Structuring MAP3F within a deep reinforcement learning paradigm

Deep learning techniques have made significant advancements in various fields in recent years. The unique structure of convolutional neural networks (CNNs) enables them to extract deep features from images. This paper proposes a feature fusion model which leverages deep features extracted from 1D CNN, 2D CNN, and 3D CNN in order to develop a higher-performance classifier. The network architecture, as presented in this paper and illustrated in Fig. 4, takes three current frames per agent as input. These frames are fed

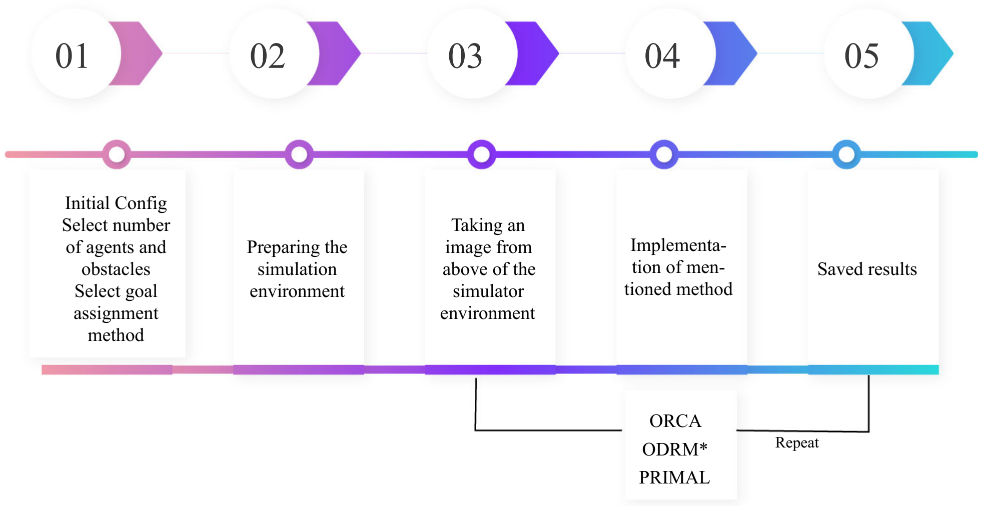
01
02
03
04
05
Initial Config Select number of agents and obstacles Select goal assignment method
Preparing the simulation environment
Taking an image from above of the simulator environment
Implementation of mentioned method
Saved results
ORCA
ODRM*
PRIMAL
Repeat

Fig. 2 Workflow of dataset creation

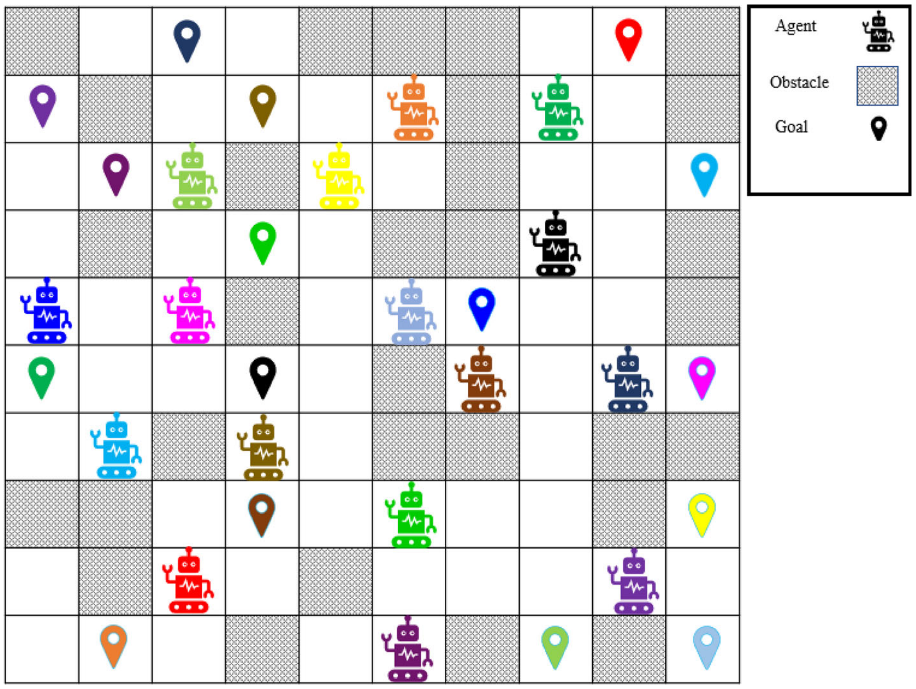

Fig. 3 Schematic representation of the MAP3F simulation environment

into three parallel branches, each responsible for extracting different features. The two branches which extract 1D and 2D features are responsible for general feature extraction, whereas the 3D feature extraction branch is responsible for partial feature extraction [25]. Since the input images may be crowded or sparse, feature extraction is crucial to ensure robust performance. The heterogeneous features extracted from the three branches are combined using a hybrid network.

2.1.1 State representation

Using multidimensional complexity function, whether 1D, 2D, or 3D, has different effects on agent decision-making in dense and sparse environments. In dense environments where there are numerous obstacles and nearby agents, the choice of dimensionality of the complexity function can influence the agents' ability to perceive and reason about the environment. 1D features operate along a single axis and is typically used for analyzing sequential data. In the context of agent decision-making, 1D features can be useful for processing temporal sequences of agent states or actions. This complexity function can capture temporal dependencies and behavioral patterns of the agent over time. In dense environments, 1D features can be employed to analyze the temporal dynamics of motion and agent interactions, assisting agents in decision-making based on past experiences and observed

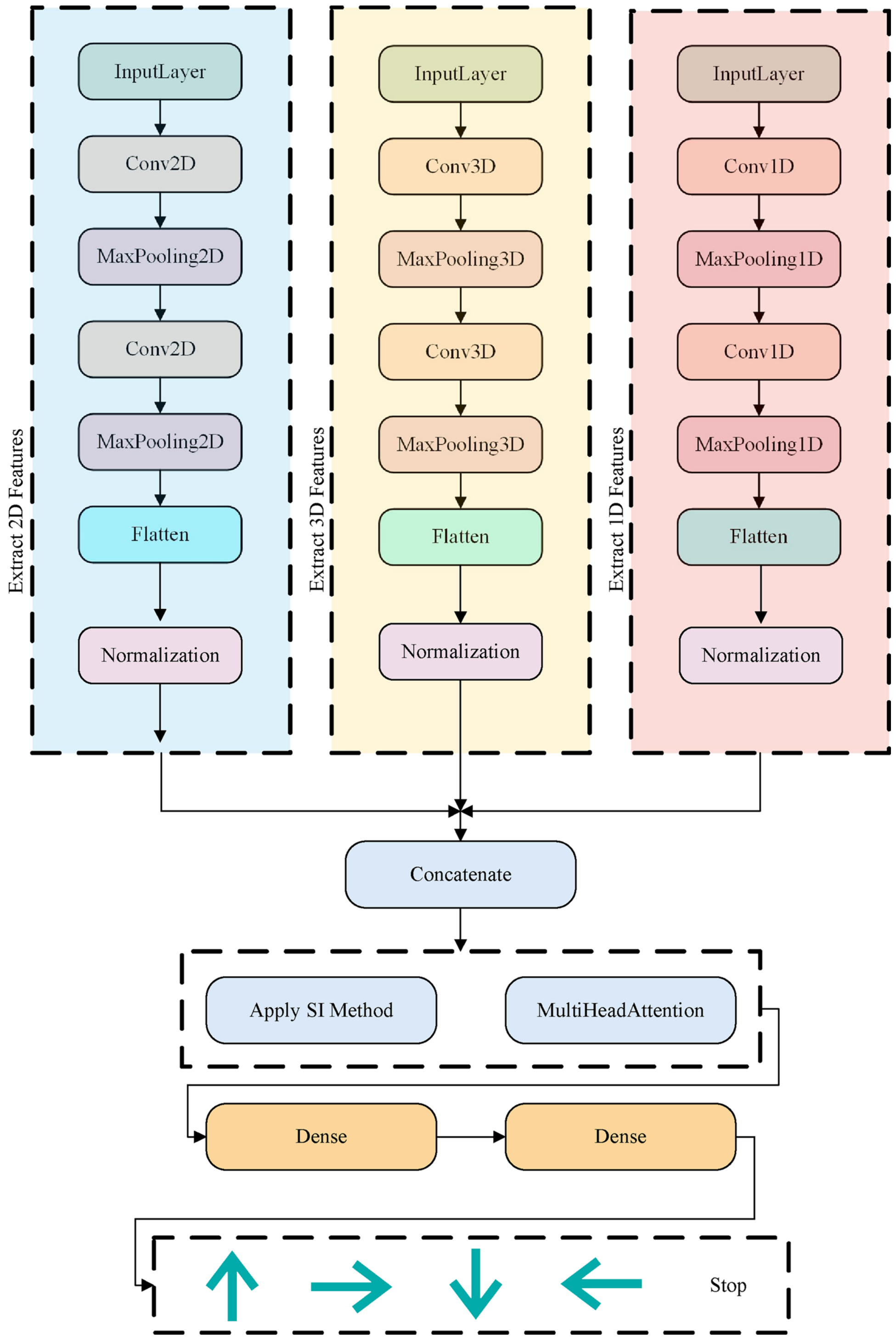

Fig. 4 The MAP3F neural network architecture

patterns. Consider a 1D CNN scenario with an input signal $x[n]$ and a filter $h[k]$. Here, k is the index used to iterate over the elements of the filter. The convolution of $x[n]$ with $h[k]$ is defined as:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{1}$$

where:

- $x[n]$ is the input signal at time n;
- $h[k]$ is the filter response at time k;
- $y[n]$ is the output signal at time n;
- k iterates over the elements of the filter in the 1D convolution;

2D features operate on two-dimensional grids and are commonly used for image analysis. In the context of agent decision-making, 2D features can be valuable in less dense environments where agents have more spatial freedom for movement and interaction. It allows agents to observe and analyze spatial relationships, such as distances between agents and obstacles or access to free paths. By applying 2D complexity to sensory input, agents can extract spatial features and make informed decisions based on their perception of the environment. In the context of a 2D CNN, the convolution operation applied to an input image and a filter can be mathematically described as follows:

$$\begin{aligned} S(i,j) &= (I * K)(i,j) \\ &= \sum_m \sum_n I(m,n) \cdot K(i-m, j-n) \end{aligned} \tag{2}$$

- I: The input image to the CNN;
- K: The kernel or filter applied to the input image;
- m, n: Indices iterating over the dimensions of the filter K in a 2D CNN. These typically represent the rows and columns of the filter, respectively;
- i, j: Indices representing the position in the output feature map;

Equation (2) represents the sum of element-wise products between the filter K and the input image I as the filter is slid across the image. Each position (i, j) in the output feature map S is computed by this convolution, resulting in a new representation of the original input, capturing the presence of specific features defined by the filter K.

3D features extend this concept to three dimensions and is typically used for volumetric data analysis. In the context of agent decision-making, 3D features may be applicable in dense environments where agents need to reason simultaneously about spatial and temporal aspects. This complexity function enables agents to comprehend spatial relationships and temporal patterns in a unified manner. By utilizing 3D features, agents can analyze the three-dimensional structure of the environment, including the positions of other agents, obstacles, and their previous trajectories, to make decisions which account for both the current state and the historical context of the environment [26]. While 3D data may encompass elements of 1D and 2D data, it is a distinct entity with its own unique characteristics. 3D data is not merely a combination of 1D and 2D data; rather, it adds a critical dimension which is essential for understanding and processing spatial-temporal information. This distinction is crucial for the effective application of convolutional neural networks in diverse data types, ensuring that each dimension's inherent properties are optimally utilized. For a given input volume V and a filter D, where m, n, and l are the indices iterating over the three dimensions of the filter in a 3D CNN (representing width, height, and depth or time respectively), the convolution operation can be defined as:

$$\begin{aligned} T(i,j,k) &= (V * D)(i,j,k) \\ &= \sum_m \sum_n \sum_l V(m,n,l) \cdot D(i-m, j-n, k-l) \end{aligned} \tag{3}$$

- $T(i, j, k)$ represents the output of the convolution at position (i, j, k);
- $V(m, n, l)$ refers to the value at position (m, n, l) in the input volume;
- $D(i-m, j-n, k-l)$ is the value at the corresponding position in the filter;

The summations over m, n, and l iterate through the entire filter, applying it to the input volume to produce the convolved output. In summary, the choice of multidimensional complexity function depends on the specific characteristics of the environment and the type of information agents need to process for decision-making. While 1D features can be useful for analyzing temporal sequences, 2D features are effective for spatial analysis in less dense environments, and 3D features are suitable for understanding simultaneous spatial and temporal aspects in dense environments. The selection should align with the agents' perception requirements and the nature of the decision-making tasks they need to perform [27].

Combining the outputs of 1D, 2D, and 3D CNNs can be beneficial for decision-making agents in environments with varying densities. By combining these different convolutional dimensions, agents can leverage the strengths of each dimension to enhance their perception and decision-making abilities. In environments with variable densities, the information available to agents can significantly differ. Using a combination of 1D, 2D, and 3D CNNs allows agents to process and analyze various aspects of the environment simultaneously. By integrating the outputs of 1D, 2D, and 3D CNNs, agents can create a comprehensive understanding of the environment. This composite representation provides rich and diverse inputs for decision-making processes, enabling agents to make informed decisions while considering the contextual information of the environment across different densities.

To illustrate the significance of multi-dimensional feature fusion, a test was conducted using data derived from the database created for this study. The data were divided into two categories, training and testing. The result of this comparison is displayed in Sect. 3.

2.1.2 DQN training formulation in MAP3F

In the MAP3F framework, each state s_t is an amalgamation of observations derived from 1D, 2D, and 3D data sources, offering a comprehensive environmental context that informs the agent's decision-making process. At any given timestep, the agent is presented with the possibility to execute a movement in one of four cardinal directions: up, down, left, right and stop. This set of potential movements constitutes the agent's action space a_t.

The reward function within MAP3F is meticulously crafted to mirror the intricacies of the decision-making environment encountered by the agent. It encompasses rewards for successful goal attainment (R_{goal}), penalties for any collisions ($R_{\text{collision}}$), and incentives designed to promote efficient and strategic pathfinding (R_{step} and $R_{\text{efficient}}$). In this structured methodology, the focus is on steering the agent toward behaviors that epitomize the principles of efficient navigation and secure interactions within the multi-agent framework. The intent is to align the agent's actions with the overarching objectives of optimizing pathfinding efficiency while ensuring safety and collaboration among agents in the system.

The aggregate reward guides agents toward effective decision-making:

$$R_{\text{total}} = R_{\text{goal}} + R_{\text{collision}} + R_{\text{step}} + R_{\text{efficient}}.$$

The adaptive deep Q-learning with feature fusion for trajectory planning algorithm encapsulates the innovative essence of the MAP3F method, marking a significant leap in the domain of autonomous decision-making within multi-agent systems. This algorithm commences with the initialization of a replay memory and an action-value function, both equipped with pre-trained weights. This preparatory step lays the foundation for an advanced integration of deep learning insights into the decision-making process. The algorithm further defines a comprehensive action set, which, through the process of feature fusion, is augmented to ensure that every decision made by an agent is informed by a rich, multi-dimensional understanding of the environment.

This feature fusion process is pivotal to MAP3F's capability to extract and synthesize insights from diverse data dimensions-spanning temporal, spatial, and spatiotemporal dynamics-thereby enabling agents to navigate complex environments with unprecedented precision and adaptability. A novel aspect of this algorithm is the introduction of a context score (C_t), a metric designed to dynamically modulate the exploration rate (ϵ) based on the current state and performance of the agent. This dynamic adjustment mechanism allows for a more nuanced exploration-exploitation balance, catering to the evolving challenges and complexities encountered by agents. High context scores prompt increased exploration to tackle unfamiliar or challenging scenarios, whereas lower scores reduce exploration in well-understood contexts, optimizing the learning trajectory.

Through the strategic application of DRL principles, complemented by the integration of 1D, 2D, and 3D features, the MAP3F approach not only enhances the agents' capability to make informed decisions but also ensures continuous learning and adaptation. This algorithmic realization, underscored by the dynamic adjustment of ϵ and the strategic fusion of multi-dimensional features, sets new benchmarks for efficiency, adaptability, and decision-making acumen in multi-agent systems. It illustrates the potential of leveraging deep learning and feature fusion in harmony to navigate the complexities of autonomous pathfinding and collision avoidance. DQN integrates deep neural networks with Q-learning, updating Q-values based on the equation [28]:

$$\begin{aligned} Q_{\text{new}}(s, a) &\leftarrow Q(s, a) \\ &+\alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right], \end{aligned} \tag{4}$$

Incorporating experience replay and fixed Q-targets, DQN iteratively adjusts the neural network weights to minimize the difference between predicted and target Q-values, optimizing the policy for navigating complex environments [28]. By detailing the convolutional operations of 1D CNN, 2D CNN, and 3D CNN alongside the DQN training formulation, this section provides a holistic view of how MAP3F processes environmental data and learns optimal paths for multi-agent systems in diverse and dynamic settings.

This comprehensive integration of deep learning architectures with reinforcement learning principles forms the core of the MAP3F approach, offering a nuanced and effective methodology for addressing the challenges of pathfinding and collision avoidance in multi-agent systems.

2.2 Feature selection methods employed in MAP3F

After integrating the features, in this section, two indexes have been used to extract important features, namely the separation index (SI) criterion [29] and multi-head attention layer. The model's response quality exhibits significant differences both before and after applying feature selectors. Before using a feature selector, the model's response quality without employing the feature selector would depend on the architecture' and training of the model.

The architecture of the proposed model, is trained on a wide range of data, but it may not always produce accurate or contextually appropriate responses. Without explicitly incorporating feature selection mechanisms like multi-head attention or SI, the model's responses may not be optimized for selecting the most relevant features from the input. It may generate responses based on the overall input information without explicitly emphasizing or downplaying specific features. After employing a feature selector, the model gains the ability to assign diverse attention weights to various input features. This helps in emphasizing the most relevant features for the given task. Consequently, the model's responses are more likely to focus on the crucial information and exhibit improved relevance. Feature selector allows the model to capture diverse patterns and dependencies within the input sequence. This enhances the model's understanding of the contextual relationships between different features.

As a result, the model can generate responses which exhibit better contextual coherence and understanding. Furthermore, feature selector enables the model to assign higher attention weights to features which are more informative and relevant. This mechanism aids in feature selection, as important features receive more attention, while less relevant or noisy features receive lower attention weights. Consequently, the model's responses are likely to be more focused on the salient features, improving the overall quality of the generated output. The maximum impact of feature selection was observed in complex scenarios.

2.2.1 Separation index

A novel distance-based index is introduced in [29] for evaluating the flow of input data through layers of a CNN in classification problems. This index focuses on assessing the complexity of the dataset and the separability of classes. A dataset is considered complex when its samples are distributed in overlapping regions across different classes, posing challenges for classification. On the other hand, class separability refers to the extent to which samples of a class are not located in overlapping regions. Let $\{x^q, l^q\}_{q=1}^{Q}$ denote the pairs of input patterns and output target labels, and let $\{x_L^q\}_{q=1}^{Q}$ represent the dataflow in layer $L \in (1, 2, \ldots, n_{\text{Layer}})$. The SI at layer L is defined as follows:

$$\text{SI}_L = \{x_L^q\}_{q=1}^{Q} \tag{5}$$

$$\text{SI}(\{x_L^q\}_{q=1}^{Q}, \{l^q\}_{q=1}^{Q}) = \frac{1}{Q}\sum_{q=1}^{Q} \phi(l^q - l^{q_{\text{near}}^L}) \tag{6}$$

$$q_{\text{near}}^L = \text{argmin} \left\| x_L^q - x_L^h \right\| \tag{7}$$

In the context of evaluating the data flow through the layers of a CNN for classification tasks, a distance-based index, as introduced in [29], serves as a metric for assessing dataset complexity and class separability. Here's an explanation of the variables used in Equations (1) to (3), which define the separability index (SI) at a given layer L within the CNN:

- x^q: Represents an input pattern to the CNN, where q indexes the input instances, ranging from 1 to Q, with Q being the total number of instances in the dataset.
- l^q: Denotes the output target label associated with the input pattern x^q, indicating the class to which x^q belongs.
- $\{x^q, l^q\}_{q=1}^{Q}$: This notation encapsulates the pairs of input patterns and their corresponding output target labels for all instances in the dataset.
- x_L^q: Refers to the representation of the input pattern x^q at layer L of the CNN, effectively capturing the data flow within that layer. Layer L can be any layer within the network, from the first to the n_{Layer}th layer, where n_{Layer} represents the total number of layers in the CNN.
- SI_L: The separability index at layer L, initially defined as $\{x_L^q\}_{q=1}^{Q}$, which represents the set of all transformed input patterns at layer L.
- $\text{SI}(\{x_L^q\}_{q=1}^{Q}, \{l^q\}_{q=1}^{Q})$: Represents the computed separability index, which quantifies the class separability based on the transformed input patterns at layer L and their corresponding labels. It is calculated as the average over all input instances q, where $\phi(l^q - l^{q_{\text{near}}^L})$ evaluates the difference between the label of x^q and the label of its nearest neighbor within the same layer.
- ϕ: A function applied to the label difference, which could be an indicator function or another function designed to measure the discrepancy between labels.
- q_{near}^L: Identifies the index of the nearest neighbor to x_L^q at layer L, based on a distance metric. The nearest neighbor is the instance x_L^h that has the minimum distance from x_L^q, as determined by the argmin operation over the norm $\left\| x_L^q - x_L^h \right\|$.

These equations collectively provide a methodological framework for quantifying how well-separated the classes are within the feature space defined by the activations at layer L of the CNN. This separation is a crucial factor in the network's ability to distinguish between different classes, thereby directly impacting classification performance. Equations (4) to (6) describe the core logic of method SI. The proof of these equations and further details of this method are discussed in [29]. This index quantifies the proportion of samples within the overlapped area relative to the total sample count, offering an insight into the dataset's complexity. Saffar et al. (2023) delve deeper into categorizing features as exclusive or common and delineate conditions for class separability using this framework. Additionally, they underscore the integration of the separation index (SI) into the suite of complexity measurement indices, underpinned by probabilistic evidence.

Several case studies [29, 30] are presented to demonstrate the benefits of the SI. Firstly, the index is employed to rank various supervised datasets based on their level of challenge. Secondly, the separability of dataflow is evaluated across layers of two pre-trained VGG-16 networks in classifying CIFAR-10, CIFAR-100, and Caltech-101 datasets. Additionally, the dataflow evaluation is extended to a trained ResNet-18 model using the Fashion-MNIST dataset. The

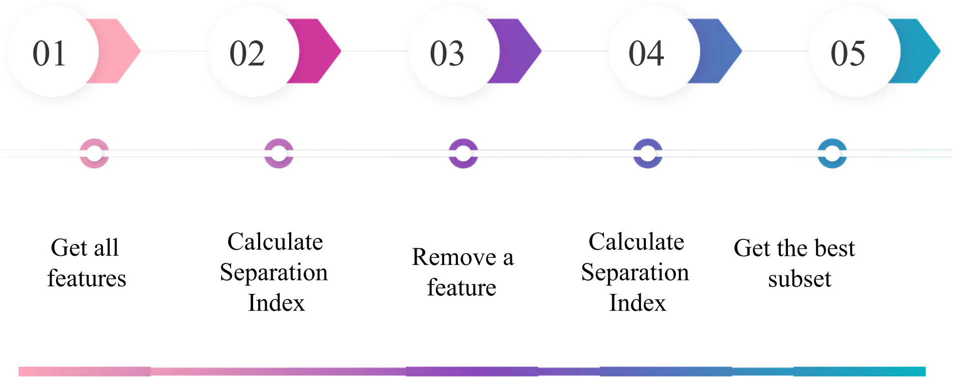

Fig. 5 Separation index in path planning

paper [29] also observes a strong correlation between the correct classification rate and the SI throughout the layers, particularly as the SI increases.

2.2.2 Separation index in path planning

In this section, the procedure for employing a SI as a feature selector is elucidated in the study. After integrating 1D, 2D, and 3D features into a set of attributes, a subset of these attributes with significance needs identification. In each step of this method, one attribute is removed through the backward selection approach, and the fully connected layers are retrained while assessing the SI. This iterative process persists until the backward selection method successfully identifies the most optimal features. As Fig. 5 demonstrates, the workflow of this index is as follows: Initially, it receives all the features and calculates the value of the SI. Then, at each stage, it removes one of the features using the backward selection algorithm and recalculates the value of the SI. This process continues until an optimal and influential subset of these features is selected.

2.2.3 Multi-head attention layer for feature selection

The multi-head attention layer can be beneficial for feature selection in certain contexts. The multi-head attention mechanism is commonly used in transformer-based models, such as the Transformer architecture, which has gained significant popularity in natural language processing tasks. In a multi-head attention layer, the input features are transformed using multiple attention heads, each capturing different aspects of the input representation. These attention heads allow the model to focus on different parts of the input sequence simultaneously, enabling it to extract relevant and informative features. The multi-head attention layer for feature selection is a sophisticated component in neural network architectures, particularly beneficial in the realm of path planning and decision-making tasks. This layer operates on the principle of attention mechanisms, which allow the network to focus on different parts of the input sequence, crucial for understanding complex patterns. In the context of feature selection for path planning, the multi-head attention layer simultaneously processes multiple ’heads’ of attention. Each head independently attends to different parts of the input, capturing diverse aspects of the data, such as spatial relationships and temporal dynamics in a path planning scenario. This parallel processing enhances the model’s ability to discern pertinent features from a vast set of input data, leading to more informed and accurate decision-making. By integrating multiple attention heads, the layer offers a comprehensive view of the input, significantly improving the model’s performance in complex environments where understanding of intricate patterns is essential.

By using multiple attention heads, the model can learn diverse and complementary representations, as each head attends to different parts of the input. This helps in capturing various patterns and dependencies within the data. Additionally, the attention mechanism assigns weights to different input elements based on their relevance, effectively performing a form of feature selection. Features which are more relevant to the task at hand tend to receive higher attention weights, while less relevant features receive lower weights. Through this process, the multi-head attention layer can effectively capture and emphasize important features while downplaying or ignoring less relevant ones. This inherent feature selection capability can enhance the model’s ability to focus on relevant information, improving its overall performance on tasks such as sequence classification, machine translation, text generation, and more. It is worth noting that while the multi-head attention layer can assist with feature selection, it is typically used as part of a larger neural network architecture. The effectiveness of feature selection also depends on the specific task, dataset, and the overall architecture and training process of the model [31].

Table 2 Comparative analysis of model performance across four scenarios with distinct feature combinations

Feature extractor	Confusion matrix					Description
		UP	DOWN	LEFT	RIGHT	
1D Feature	UP	0.64	0.18	0.15	0.03	1D features extract, and the training and evaluation stage perform only with these features
	DOWN	0.13	0.55	0.23	0.09	
	LEFT	0.06	0.12	0.71	0.11	
	RIGHT	0.02	0.08	0.18	0.72	
2D Feature	UP	0.7	0.15	0.07	0.08	2D features extract, and the training and evaluation stage perform only with these features
	DOWN	0.1	0.6	0.3	0.0	
	LEFT	0.06	0.07	0.8	0.07	
	RIGHT	0.03	0.04	0.16	0.77	
3D Feature	UP	0.8	0.15	0.02	0.03	3D features extract, and the training and evaluation stage perform only with these features
	DOWN	0.15	0.6	0.15	0.1	
	LEFT	0.15	0.01	0.78	0.06	
	RIGHT	0	0.02	0.18	0.8	
Feature fusion	UP	0.8	0.05	0.15	0	1D, 2D, and 3D features extract. The training and evaluation stage performed with the combination of all features
	DOWN	0.17	0.68	0.07	0.08	
	LEFT	0.07	0.01	0.82	0.10	
	RIGHT	0.03	0	0.17	0.80	

3 Results and discussion

The approach presented in this article has been examined and compared with previous methods from three distinct perspectives. These perspectives have been defined under three specific indicators: interaction index, reachability index, and field of view index. Considering the negative correlation between obstacle density and the performance of the proposed method observed in this study, the results of these indicators have been reported at three different levels of density to compare with previous published articles. The interaction index indicates the percentage of scenarios where no collision occurred, while the reachability index represents the time taken by the slowest agent to reach its goal. The field of view index shows the observable region for an agent. The results of the presented method have been compared with benchmark methods such as ORCA [17], ODRM* [24], and PRIMAL [12], which can be regarded as state-of-the-art approaches presented in the motion planning literature.

The approach presented in this article utilizes the combination of 1D, 2D and 3D features for action selection by each agent. The goal of this approach is to select appropriate actions by agents in various types of environments (both dense and sparse). By combining the outputs of these different CNN dimensions, agents can gain a comprehensive understanding of the environment and make decisions which consider both global and local perspectives. This combination enables agents to maintain a balance in their decision-making process by considering the overall context and patterns (global decisions) while also taking into account immediate spatial interactions and local interactions (local decisions). This combination provides agents with the ability to make more complex and informed decisions which consider both macro and micro aspects of the environment.

Table 3 Performance metrics for different features

Feature type	Accuracy	Precision	Recall	F1 Score
1D Feature	0.655	0.6653	0.655	0.6601
2D Feature	0.7175	0.7307	0.7175	0.7240
3D Feature	0.7450	0.7487	0.7450	0.7469
Feature fusion	0.7750	0.7901	0.7750	0.7825

To illustrate the significance of multi-dimensional feature fusion, an experiment was conducted using a specifically prepared dataset, which was split into training and testing sets. The proposed architecture, detailed in this paper, underwent training in four distinct modes, with the results subsequently evaluated. This network is characterized by its three-branch structure.

Table 2 aids in understanding the impact of each branch. Specifically, the first row presents the Confusion Matrix for the scenario where only the 1D branch was utilized. The second and third rows reveal the outcomes when exclusively the 2D and 3D branches were employed, respectively. Contrastingly, the fourth row provides insights into the performance when all three branches were concurrently trained. These results clearly indicate that the integration of 1D, 2D, and 3D features significantly enhances the model's decision-making capabilities. Furthermore, Table 3 offers a comparative analysis of the four aforementioned scenarios, employing relevant metrics. Notably, the combined-feature scenario outperforms the individual ones, as evidenced in the table. The feature selection methods employed in this study are elaborated upon in Sect. 2.2.

In terms of manageable agent count, ORCA, PRIMAL, and MAP3F are capable of handling a large number of agents, while ODRM* incurs higher computational costs with an increase in the number of agents. In terms of the quality of generated paths, ODRM* and PRIMAL guarantee global optimization, while ORCA may lead to local optima. In terms of scalability, ORCA, PRIMAL, and MAP3F are considered efficient methods. ORCA is an online method capable of managing a large number of agents in real-time, while PRIMAL and MAP3F are offline methods that can also handle a large number of agents but require more computational resources compared to ORCA. ODRM* is a more computationally expensive method that can manage a large number of agents but generally operates slower than ORCA, PRIMAL, and MAP3F. Additionally, the MAP3F and PRIMAL methods, which are learning-based approaches, have a higher capacity for accepting uncertainty compared to the classical methods of ORCA and ODRM*. The investigations conducted in this article demonstrate that the proposed method, MAP3F outperforms ORCA, ODRM*, and PRIMAL in dense and complex environments.

In terms of quality, all four methods have demonstrated effectiveness in solving multi-agent pathfinding problems. However, the findings indicate that ORCA is more conservative and cautious in its approach, leading to more conservative paths for the agents. ODRM* is more aggressive in its approach, resulting in more aggressive paths for the agents. PRIMAL and MAP3F achieve a balance between ORCA and ODRM*, leading to paths which are both efficient and safe. The results are summarized in Table 4.

3.1 Goal assignment

Goal assignment in multi-agent path planning refers to the process of determining which goals or tasks should be assigned to each individual agent in a system composed of multiple autonomous agents. It involves assigning specific objectives or destinations to agents in a coordinated manner to achieve the overall goals of the system. The importance of goal assignment in multi-agent path planning lies in its role in optimizing system performance and achieving efficient task completion [32]. In this article, the method of assigning the target to the agents is done based on the following three simple functions.

1. Worst-Case Assignment: In the context of the worst-case assignment, the primary objective is either to minimize the overall completion time or maximize the cost of the entire system. This form of goal assignment typically relies on the maximum distance or cost incurred by an agent to reach a specific goal. Considering the agents' movement in four directions: up, down, left, right, and stop. the distance between two points is defined as the number of cells an agent traverses between them. This definition of distance is integral to the calculations of goal priority (GP), formulated as follows:

$$GP_{\text{dis}} = \max_{i=1}^{n} \text{Dis}(\text{agent}_i, \text{goal}_i) \tag{8}$$

$$GP_{\text{cost}} = \max_{i=1}^{n} \text{Cost}(\text{agent}_i, \text{goal}_i) \tag{9}$$

In the above equations, GP_{dis} and GP_{cost} represent the maximum distances and costs, respectively, for all agents to reach their assigned goals. The term 'Dis' denotes the distance, calculated based on the number of cells an agent must navigate through, and 'Cost' reflects the resources or effort required for an agent to move from its current location to its goal. These formulations capture the essence of the

Table 4 Comparison of pathfinding methods

Feature/method	ORCA	ODRM*	PRIMAL	MAP3F
Handle agent count	High	Moderate	High	High
Path quality	Locally optimal	Globally optimal	Balanced	Balanced
Scalability	High	Low	High	High
Computational efficiency	High	Low	Moderate	Moderate
Uncertainty tolerance	Low	Low	High	High
Approach	Conservative	Aggressive	Balanced	Balanced
Performance in dense environments	Moderate	Low	High	Very High

approach to goal assignment in scenarios demanding maximal efficiency or cost.

2. Best-Case Assignment: The best-case assignment focuses on minimizing the completion time or cost for individual agents, irrespective of the overall system performance. Here, the goal assignment can be determined based on the minimum distance or cost to reach a specific goal. which can be formulated as follows:

$$\text{GP dis} = \min(\text{Dis}(\text{agent}, \text{goal})) \tag{10}$$

$$\text{GP cost} = \min(\text{Cost}(\text{agent}, \text{goal})) \tag{11}$$

3. Middle-Case Assignment: The middle case assignment aims to balance the completion time or cost among all agents, focusing on both individual and overall system performance. In this scenario, the goal assignment is determined based on optimizing a combination of distance or cost metrics. This optimization is articulated through the following formulas, where the objective is to maximize the respective distance and cost for each agent-goal pairing across all agents n. The maximum distance (GPdis) and cost (GPcost) are calculated as follows:

$$GP\text{dis} = \max_{i=1}^{n} \text{Dis}(\text{agent}_i, \text{goal}_i) \tag{12}$$

$$GP\text{cost} = \max_{i=1}^{n} \text{Cost}(\text{agent}_i, \text{goal}_i) \tag{13}$$

In these equations, GPdis represents the greatest distance any individual agent is required to travel to its assigned goal, while GPcost signifies the highest cost incurred by any agent to reach its goal. These calculations are crucial for ensuring a balanced distribution of tasks and resources among all agents.

3.2 Interaction index

The interaction index was defined as the percentage of scenarios, in which no collisions occurred.

$$\text{Success rate} = \frac{\text{number of non-collision scenarios}}{\text{All executed scenarios}} \tag{14}$$

Fig. 6 compares this index for the ORCA, ODRM*, PRIMAL, and MAP3F algorithms. These results are related to an environment with a density of 0.3, where the goal assignment to agents has been challenging. Table 5 displays the comparison results of the interaction index of these algorithms in environments with different densities and varying numbers of agents. As observed in Table 5, the interaction index significantly decreases as the environment becomes more complex. This reduction occurs with a lower slope in the proposed method compared to previous approaches. The interaction index increased by twenty percent after applying the feature selection phase. Among the two selected methods, the SI method led to the selection of higher-quality features and, consequently, higher accuracy in decision-making. The results of this method are presented in Sect. 3.

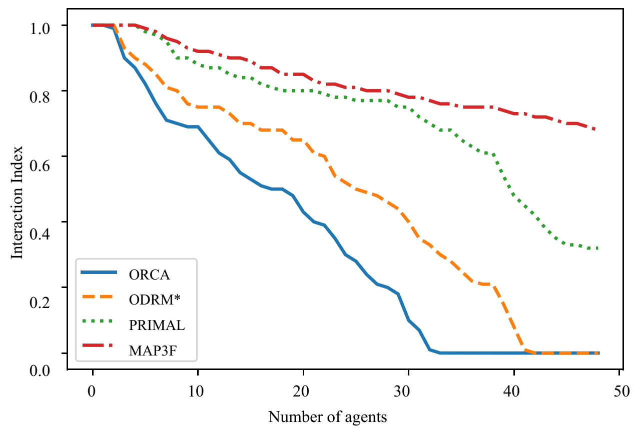

Fig. 6 Interaction index (worst-case assignment)

3.3 Reachability index

Figure 7 displays the reachability index, which compares the time it takes for the last agent to reach its target across different algorithms. As Fig. 7 clearly shows, as the number of agents increases, the computational time also increases. This increase follows different trends in various algorithms. Figure 7 compares this index in the ORCA, ODRM*, PRIMAL, and MAP3F algorithms. These results are related to an environment with a density of 0.3, where the goal assignment to agents is worst case. Table 6 displays the comparison results of the reachability index of these algorithms in environments with different densities and varying numbers of agents. As can be observed from Table 6, the execution time of a scenario significantly increases as the environment becomes more complex. This increase in time occurs in the proposed method with a lower slope compared to previous methods. To quantify these observations, Equations (15) and (16) express the reachability outcomes mathematically. Equation (15) defines RO as the time taken by the slowest agent in the RRT* algorithm, disregarding inter-agent collisions. Equation (16) defines RP as the corresponding time in the proposed algorithm.

$$RO = T_{\text{slowest}}^{\text{RRT*}} \quad \text{(Ignoring inter-agent collisions)} \tag{15}$$

$$RP = T_{\text{slowest}}^{\text{Proposed}} \tag{16}$$

3.4 Scalability index

In the domain of multi-agent systems, scalability serves as a pivotal metric for assessing a system's capability to effectively manage an escalating number of agents without a

Table 5 Interaction index for different methods

Number of agents	Density	ORCA	ODRM*	PRIMAL	MAP3F
10	0.1	1	1	1	1
20		1	0.99	1	1
30		0.99	0.97	1	1
40		0.98	0.95	1	1
50		0.97	0.92	1	0.99
10	0.2	0.95	0.9	0.99	0.99
20		0.94	0.8	0.97	0.98
30		0.91	0.87	0.94	0.97
40		0.86	0.75	0.89	0.93
50		0.76	0.65	0.81	0.89
10	0.3	0.7	0.8	0.9	0.95
20		0.5	0.68	0.8	0.85
30		0.2	0.46	0.77	0.8
40		0	0.21	0.61	0.75
50		0	0	0.32	0.68

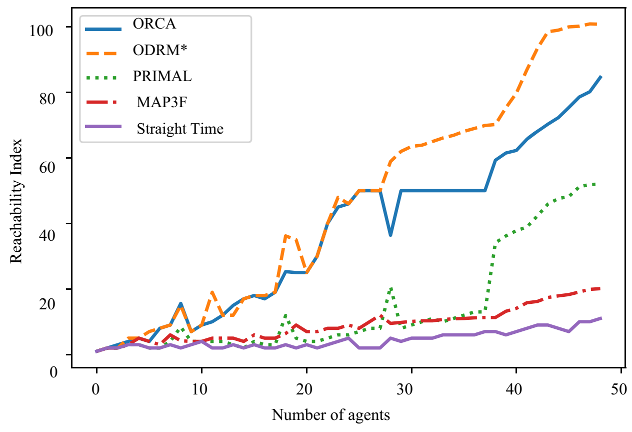

Fig. 7 Reachability index (worst-case assignment)

Table 6 Reachability index in different methods including MAP3F

Agents	Density	ORCA	ODRM*	PRIMAL	MAP3F
10	0.1	0.25	0.26	0.2	0.2
20		0.26	0.28	0.2	0.2
30		0.28	0.3	0.2	0.2
40		0.37	0.45	0.2	0.2
50		0.52	0.65	0.25	0.2
10	0.2	0.28	0.3	0.2	0.22
20		0.38	0.45	0.3	0.3
30		0.5	0.61	0.54	0.54
40		1.8	3.5	1.7	1.8
50		4.5	6.8	5.3	4.4
10	0.3	15.6	15	8.4	6.8
20		25.3	36.2	11.9	9.3
30		36.4	58.9	20.7	12.6
40		59.3	70.2	34.11	25.6
50		84.6	100.8	52	45.9

marked decline in performance. Here, performance specifically refers to the time each agent requires to reach its designated goal. The scalability of a system is quantitatively evaluated through a formula which inversely correlates the increase in the average goal-reaching time experienced by the system as the number of agents, A, grows, against a predetermined maximum allowable time increase. This relationship is encapsulated in the following mathematical expression:

$$\text{Scalability}(A) = 100 - \left(\frac{\text{Average Time Increase to Goal}(A)}{\text{Maximum Time Increase}}\right) \times 100 \tag{17}$$

In this refined context, scalability is gauged on a scale ranging from 0 to 100, where a perfect score of 100 signifies unparalleled scalability, implying no increase in the time required for agents to achieve their objectives as the system scales. The term "Average Time Increase to Goal" computes the escalation in the time it takes for agents to reach their goals as A expands, whereas "Maximum Time Increase" denotes a threshold, the highest tolerable increment in goal-reaching time, determined through empirical studies or theoretical estimations. This formula facilitates a standardized comparison of scalability across different algorithms or systems, highlighting their efficiency in accommodating a burgeoning agent count without compromising on the time efficiency of goal attainment. It's crucial to customize the "Maximum Time Increase" parameter in line with the spe-

Table 7 Scalability index across different methods, including MAP3F

Agents	Density	ORCA	ODRM*	PRIMAL	MAP3F
10	0.1	95	94	98	98
20		93	91	97	98
30		90	88	96	98
40		85	80	95	97
50		75	70	92	96
10	0.2	92	90	97	98
20		88	85	94	96
30		75	70	90	91
40		65	55	85	88
50		50	40	80	83
10	0.3	40	35	70	76
20		36	31	64	69
30		31	25	58	63
40		22	22	54	58
50		20	16	48	51

cific experimental framework or application scenario to yield precise scalability evaluations.

Table 7 exemplifies the scalability index recalculated to emphasize the time efficiency of goal attainment, providing a direct measure of how well various algorithms or systems sustain their performance in terms of time to reach goals as the number of agents increases.

3.5 FOV index

Increasing the size of the agent's perspective can have a significant impact on its ability to make informed decisions about its next actions. Having a broader perspective allows the agent to understand a wider range of the environment. By observing a larger portion of its surroundings, the agent gains access to more information about spatial composition, object presence, and motion of entities. This broader perspective enables the agent to develop a comprehensive understanding of the current state and potential future states, leading to more informed decision-making [33].

With a broader perspective, the agent can acquire background information which may be vital for decision-making. It can observe object relationships, detect patterns, and identify relevant spatial cues. This contextual awareness provides the agent with a better understanding of the environment and allows it to make decisions which consider the overall state and long-term consequences rather than just reacting to immediate local observations. Additionally, it enables the agent to anticipate potential future states and plan its actions accordingly. By observing distant objects or events, the agent can predict obstacles, identify opportunities, and plan complex action sequences. This ability to consider a broader context empowers the agent to make decisions which optimize long-term goals or adhere to specific path densities. While having a broader perspective brings benefits, it also comes with increased computational complexity.

Processing a large volume of visual information requires additional computational resources. The agent needs to analyze and interpret a large volume of sensory data, which can result in longer processing times and increased memory requirements. Therefore, when deciding on the size of the perspective, it is necessary to consider computational limitations and available resources. Additionally, expanding the perspective distributes the agent's focus over a larger area, which may lead to reduced clarity or details in the observed scene. Furthermore, the agent may need to spend more time processing the increased information, resulting in higher time demands or response times. Thus, finding the appropriate balance between a broad perspective for comprehensive understanding and a focused perspective for efficient decision-making is crucial. As a result, increasing the size of the agent's perspective provides benefits such as improved understanding, increased contextual awareness, and enhanced action planning. However, it also poses challenges in terms of computational complexity and efficiency. Finding the right balance between a broad perspective and efficient decision-making is essential to ensure optimal performance of the agent in its environment.

The proposed solution in this study considers a bounded range around the agent as the focus, where the combination of features extracted from 1D, 2D, and 3D CNN layers can help overcome the agent's perspective limitations and aid in decision-making about the next action. Each type of CNN layer has its strengths in capturing different types of information. 1D CNNs are effective in modeling temporal dependencies, 2D CNNs excel in detecting spatial patterns, and 3D CNNs are suitable for capturing spatial and temporal features in videos or volumetric data. By combining these different types of CNNs, the agent can leverage multiple perspectives and extract a richer set of features from its environment. The combination of 1D, 2D, and 3D CNN layers enables the agent to encompass multi-scale information from the environment. While 1D CNNs capture local temporal patterns, 2D CNNs capture spatial relationships in a local neighborhood, and 3D CNNs consider spatial and temporal dependencies in a larger region.

By integrating features from all three types of CNNs, the agent can attend to different extents of the background simultaneously and provide a comprehensive understanding of the environment. The combination of different CNN layers provides flexibility and adaptability to inputs with various types and sizes. The agent can adjust the settings of CNN layers based on the specific task and environmental characteristics. This flexibility allows the agent to adjust the field of view within the constraints of the perspective and leverage relevant

Table 8 Execution time comparison for different input resolutions on NVIDIA GPU GTX 2080 and CPU i7@3.6 GHz

	NVIDIA GPU GTX 2080		CPU i7@3.6 GHz	
Input resolution	Execution time for each stage (ms)	Execution time for worst scenario (ms)	Execution time for each stage (ms)	Execution time for worst scenario (ms)
256×256	3̃.49	2̃0.94	9̃7.73	5̃86.38
128×128	2̃.64	1̃6.44	3̃6.69	2̃20.14
64×64	2̃.63	1̃5.88	3̃1.46	1̃84.56
32×32	2̃.6	1̃4.33	2̃3.05	1̃28.40
16×16	2̃.55	1̃3.6	2̃0.45	1̃20.30
8×8	2̃.48	1̃3.44	1̃8.07	9̃3.30

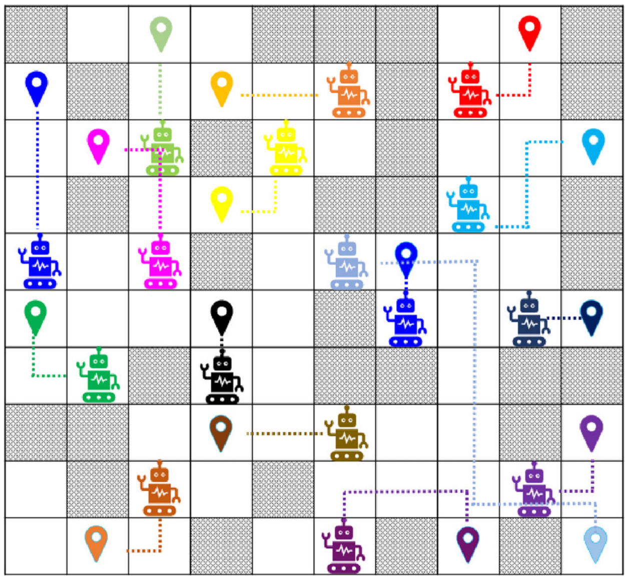

Fig. 8 Paths suggested by MAP3F in the best-case situation

information for optimal decision-making and more informed choices. Table 8 presents the results of testing a consistent scenario with different agent's field of view. As observed, the execution time significantly increases with the expansion of the field of view.

3.6 Paths generated by the proposed method

In this section, a scenario is examined using MAP3F algorithm. The same scenario is presented in Fig. 8, 9a, b. The number of agents and goals, as well as their positions in the scene, are identical. The distinction among these three figures lies in how the goals are assigned to the agents. In Fig. 8, the goals are assigned to the agents using the best-case method. In Fig. 9a, b, the worst-case method is utilized to assign the goals to the agents. Figure 8, corresponding to the Best Case, assigns the nearest goal to each agent and is solved by all four algorithms at approximately the same time. However, due to the overlap of the paths proposed by the algorithms and the density of the environment in Fig. 9a, b, this scenario has become more intricate than before. Nevertheless, Fig. 9a, b is exclusively solved by the MAP3F algorithm without any collisions. The proposed paths by the MAP3F method are exhibited in the subsequent figures.

4 Conclusions

The paper introduced a decentralized approach for navigating multiple agents and avoiding collisions within intricate environments. The proposed method, called MAP3F, utilized DRL and combined 1D, 2D, and 3D features to enhance agent decision-making. The paper compared MAP3F with other methods and evaluated its performance using three indices, namely the interaction index, reachability index, and FOV index. The outcomes demonstrate that MAP3F surpasses both classical and learning-based methods in a range of environments with varying densities. When benchmarked against the PRIMAL, ORCA, and ODRM* methods, MAP3F's efficacy shows a notable enhancement, surpassing these methods by more than 30% in complex environments with an increased number of agents. It achieved a higher interaction index in avoiding collisions, reduced the time for the slowest agent to reach its goal, and demonstrated that decreasing the field of view can decrease computation time without affecting interaction indices. Additionally, MAP3F exhibited scalability potential, indicating that it could handle an increased number of agents without compromising computation time. The paper highlighted the advantages of decentralized approaches in multi-agent systems, such as scalability, flexibility, and robustness. It also discussed the challenges and limitations of centralized methods. By utilizing DRL and leveraging multidimensional complexity functions, MAP3F enabled agents to make independent decisions based on their local observations. Furthermore, the paper provided insights into the architecture and stages of the proposed method. It described the dataset preparation, preprocessing, network architecture, and mathematical computations involved in MAP3F. The feature fusion model combined deep features extracted from 1D, 2D, and 3D CNNs to develop a higher-performance classifier. Overall, the paper contributed to the development of decentralized approaches for multi-agent pathfinding and collision avoidance. It demonstrated the effectiveness of MAP3F in complex environments and provided valuable insights for researchers and practitioners working in this field. Future work could further explore and refine the proposed method, considering

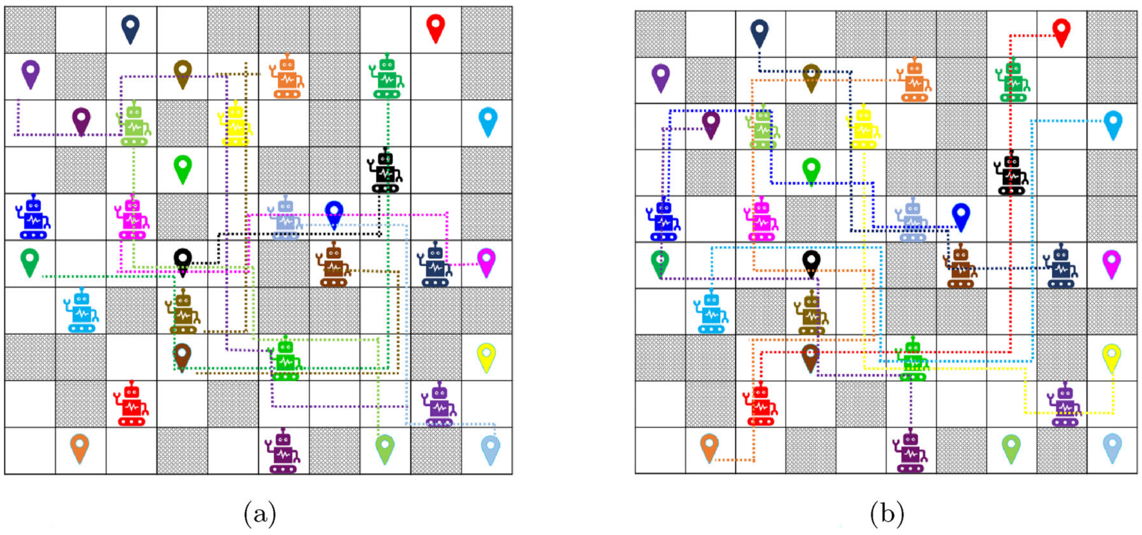
(a) (b)

Fig. 9 Paths suggested by MAP3F in the worst-case situation

different factors such as uncertainty, real-time constraints, and dynamic environments.

References

1. Le-Anh T, De Koster MBM (2006) A review of design and control of automated guided vehicle systems. Eur J Oper Res 171(1):1–23
2. Alatise MB, Hancke GP (2020) A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access 8:39830–39846
3. Mehdi F, Sefidgari BL, Barenji AV (2013) An adaptive neuro pid for controlling the altitude of quadcopter robot. In 2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR), 662–665. IEEE
4. Raibert M, Blankespoor K, Nelson G, Playter R (2008) Bigdog, the rough-terrain quadruped robot. IFAC Proc Vol 41(2):10822–10825
5. Walker Thayne T, Sturtevant Nathan R (2019) Collision detection for agents in multi-agent pathfinding. arXiv e-prints, pages arXiv–1908,
6. Cheng PDC, Indri M, Possieri C, Sassano M, Sibona F (2023) Path planning in formation and collision avoidance for multi-agent systems. Nonlinear Anal Hybrid Syst 47:101293
7. Andreychuk A, Yakovlev K, Surynek P, Atzmon D, Stern R (2022) Multi-agent pathfinding with continuous time. Artif Intell 305:103662
8. Pianpak P, Son TC (2021) Dmapf: A decentralized and distributed solver for multi-agent path finding problem with obstacles. Electronic Proceedings in Theoretical Computer Science 345
9. Chehelgami S, Ashtari E, Basiri MA, Masouleh MT, Kalhor A (2023) Safe deep learning-based global path planning using a fast collision-free path generator. Robot Auton Syst 163:104384
10. Qiang W, Zhongli Z (2011) Reinforcement learning model, algorithms and its application. In 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), 1143–1146. IEEE
11. Reijnen R, Zhang Y, Nuijten W, Senaras C, Goldak-Altgassen M (2020) Combining deep reinforcement learning with search heuristics for solving multi-agent path finding in segment-based layouts. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2647–2654. IEEE
12. Sartoretti G, Kerr J, Shi Y, Wagner G, Satish Kumar TK, Koenig S, Choset H (2019) Primal: Pathfinding via reinforcement and imitation multi-agent learning. IEEE Robot Autom Lett 4(3):2378–2385
13. Hussein A, Gaber MM, Elyan E, Jayne C (2017) Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR) 50(2):1–35
14. Chen H, Ji Y, Niu L (2020) Reinforcement learning path planning algorithm based on obstacle area expansion strategy. Intel Serv Robot 13(2):289–297
15. Lowe R, Wu YI, Tamar A, Harb J, Abbeel OAIP, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. Adv Neural Inf Process Syst 30:96
16. Chen YF, Liu M, Everett M, How JP (2017) Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In 2017 IEEE international conference on robotics and automation (ICRA), 285–292. IEEE
17. Niu H, Ma C, Han P (2021) Directional optimal reciprocal collision avoidance. Robot Auton Syst 136:103705
18. Sharon G, Stern R, Felner A, Sturtevant NR (2015) Conflict-based search for optimal multi-agent pathfinding. Artif Intell 219:40–66
19. Felner A, Stern R, Shimony S, Boyarski E, Goldenberg M, Sharon G, Sturtevant N, Wagner G, Surynek P (2017) Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In Proceedings of the International Symposium on Combinatorial Search 8:29–37
20. Long P, Fan T, Liao X, Liu W, Zhang H, Pan J (2018) Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In 2018 IEEE international conference on robotics and automation (ICRA), 6252–6259. IEEE
21. Hönig W, Kiesel S, Tinka A, Durham J, Ayanian N (2018) Conflict-based search with optimal task assignment. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems
22. Chen C, Liu Y, Kreiss S, Alahi A (2019) Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In 2019 international conference on robotics and automation (ICRA), pages 6015–6022. IEEE
23. Li Q, Gama F, Ribeiro A, Prorok A (2020) Graph neural networks for decentralized multi-robot path planning. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 11785–11792. IEEE

24. Ferner C, Wagner G, Choset H (2013) Odrm* optimal multirobot path planning in low dimensional search spaces. In 2013 IEEE International Conference on Robotics and Automation, 3854–3859. IEEE
25. Teixeira EÁ, Wesley B, Arjona RM (2022) Evaluation of 1d and 2d deep convolutional neural networks for driving event recognition. Sensors 22(11):4226
26. Li Q, Wang Q, Li X (2020) Mixed 2d/3d convolutional network for hyperspectral image super-resolution. Remote Sens 12(10):1660
27. Li J, Cui R, Li B, Li Y, Mei S, Du Q (2019) Dual 1d-2d spatial-spectral cnn for hyperspectral image super-resolution. In IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, 3113–3116. IEEE
28. Jafari R, Javidi MM, Rafsanjani MK (2019) Using deep reinforcement learning approach for solving the multiple sequence alignment problem. SN Appl Sci 1:1–12
29. Saffar M, Kalhor A (2023) Evaluation of dataflow through layers of convolutional neural networks in classification problems. Expert Syst Appl 224:119944
30. Haghpanah MA, Masouleh MT, Kalhor A (2023) Determining the trustworthiness of dnns in classification tasks using generalized feature-based confidence metric. Pattern Recogn 142:109683
31. Zhu H, Lee KA, Li H (2022) Discriminative speaker embedding with serialized multi-layer multi-head attention. Speech Commun 144:89–100
32. Zhong X, Li J, Koenig S, Ma H (2022) Optimal and bounded-suboptimal multi-goal task assignment and path finding. In 2022 International Conference on Robotics and Automation (ICRA), 10731–10737. IEEE
33. Qiu J, Chen C, Liu S, Zhang H-Y, Zeng B (2021) Slimconv: reducing channel redundancy in convolutional neural networks by features recombining. IEEE Trans Image Process 30:6434–6445