


Article

A Neural Network-Based Navigation Approach for Autonomous Mobile Robot Systems

Yiyang Chen ¹ , Chuanxin Cheng ¹, Yueyuan Zhang ^{1,*}, Xinlin Li ^{2,*} and Lining Sun ¹

¹ School of Mechanical and Electrical Engineering, Soochow University, Suzhou 215031, China; yychen90@suda.edu.cn (Y.C.); chuanxin_cheng@163.com (C.C.); lnsun@hit.edu.cn (L.S.)

² Department of Digital Media, Soochow University, Suzhou 215031, China

* Correspondence: zhangyueyuan@suda.edu.cn (Y.Z.); xlli1227@suda.edu.cn (X.L.)

Abstract: A mobile robot is a futuristic technology that is changing the industry of automobiles as well as boosting the operations of on-demand services and applications. The navigation capability of mobile robots is a crucial task and one of the complex processes that guarantees moving from a starting position to a destination. To prevent any potential incidents or accidents, navigation must focus on the obstacle avoidance issue. This paper considers the navigation scenario of a mobile robot with a finite number of motion types without global environmental information. In addition, appropriate human decisions on motion types were collected in situations involving various obstacle features, and the corresponding environmental information was also recorded with the human decisions to establish a database. Further, an algorithm is proposed to train a neural network model via supervising learning using the collected data to replicate the human decision-making process under the same navigation scenario. The performance of the neural network-based decision-making method was cross-validated using both training and testing data to show an accuracy level close to 90%. In addition, the trained neural network model was installed on a virtual mobile robot within a mobile robot navigation simulator to interact with the environment and to make the decisions, and the results showed the effectiveness and efficacy of the proposed algorithm.

Keywords: neural network; mobile robot navigation; obstacle avoidance



Citation: Chen, Y.; Cheng, C.; Zhang, Y.; Li, X.; Sun, L. A Neural Network-Based Navigation Approach for Autonomous Mobile Robot Systems. *Appl. Sci.* **2022**, *12*, 7796. <https://doi.org/10.3390/app12157796>

Academic Editors: Mingcong Deng, Augusto Ferrante, Mihaiela Iliescu and Tai Yang

Received: 4 July 2022

Accepted: 1 August 2022

Published: 3 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The navigation problems of mobile robots require a method that can successfully lead a robot to its target destination, and they are of key significance in industry [1]. For example, in an intelligent warehouse, the goods are frequently carried or transported by mobile robots from one position to another for storage management purposes. In such a navigation problem, the feature of obstacle avoidance must be considered with highest priority in the navigation algorithm design process, otherwise there may be serious accidents, such as a collision between two mobile robots or another object. To address these kinds of navigation problems, previous research has used various methods, such as convex optimization [2], virtual and behavioral structures [3], model predictive control [4], and reinforcement learning [5], to achieve an obstacle avoidance design target in practical applications. For more details, please see the overview of navigation as well as obstacle avoidance problems in [6–8].

The existing methods for navigation tasks may require a high computation load to compute the desired path for the mobile robots or ensure they follow a predefined route [9]. However, these requirements are too high for the performance of existing equipment. For instance, the microprocessors installed on the mobile robots may not have an efficient enough computation load to determine the correct path in real time. In addition, the limited choices of the predefined routes do not actually encourage cooperation between multiple mobile robots, i.e., a quick vehicle behind a slower vehicle has to queue and wait

in the same route, which is not efficient. Although the work in [10–12] has attempted to overcome these concerns, the proposed algorithms were adhoc, and their applications were restricted to the specific navigation tasks with convex shaped static obstacles or global environmental information. An artificial potential field is considered an efficient method for motion planning, but it runs the risk of falling into a local optimum.

This paper intends to deal with the class of navigation problems using obtained human decision data from the aspect of artificial intelligence. As pointed out in [13], artificial intelligence is considered as a technique aiming at replicating human behavior by making inferences according to historical learned data. It usually establishes a neural network model and tunes the parameters within this model using the obtained realistic data, so that the trained model can provide useful information for the users [14]. As explained in [15], this technique is particularly strong at making intelligent decisions for certain design problems with variability, uncertainty, complexity, and fuzziness [16]. This advantage enables its applications to many research fields, such as image recognition [17–19], data analysis [20], system identification [21–23], and mechanical fault diagnosis [24]. Note that most of the applications mentioned above deal with data using artificial intelligence, whose structures and labels are both sparse. However, the data labels of the navigation problem are not purely sparse, since one specific occasion can have multiple potential labels.

To solve this specific class of navigation problems using artificial intelligence, some work has already been completed. The work in [25] employed the Q-learning method to interact with a dynamic environment and limited the state number of its state space to fulfill a navigation algorithm. In addition, it was reported in [26] that neural network-based reinforcement learning was used to handle the navigation tasks with both static and dynamic obstacles in an uncertain workspace. Deep reinforcement learning to solve navigation problems is increasingly favored by researchers and has progressed since 2016 [27]. The exploration process is decomposed into decision-making, planning, and mapping modules to increase the modularity of the robot system. In each framework, a decision algorithm based on variant deep Q-learning (DQN) was proposed to learn and explore strategies from local maps in [28]. An actor–critic model was trained to learn navigation policy, and expert knowledge was utilized to generate subgoals in [29]. In [30], a heuristic dynamic programming method was proposed to demonstrate a Markov decision process during a maze navigation task, which included a goal network to model the reward. The research in [31] developed an obstacle avoidance method using a fuzzy neural network for wheeled mobile robots, which reduced the oscillation during obstacle avoidance and achieved a smooth path. In [32], symbolic spatial information was employed to assist robots to move in an unseen scene, which achieved good performance. Note that the aforementioned methods all incorporated a relatively heavy real-time computation load as a key component in their design framework, and it was difficult to set new plans when receiving instantaneously a large amount of information. In [33], artificial and natural landmark recognition-based simultaneous localization and mapping (SLAM) provided absolute position feedback in real time to complete the task of robot localization and modeling. For motion planning, a novel quadrupole potential field (QPF) approach was used to realize collision-free path planning. However, the modeling environment based on SLAM was time-consuming and had no ability to cope with real-time changes in scenarios, which increased the difficulty of navigation without global environmental information.

To overcome the real-time computation issue in navigation problems, this paper formulates a mobile robot navigation problem within a 2D plane with various shapes of obstacles. It divides the motion behaviors of a given autonomous mobile robot system into a finite number of motion types, and collects the environmental information (system inputs) as well as appropriate decisions (labels) of these motion types made by human users at various scenarios with random generated obstacles. These collected data are further exploited to train a neural network model that is employed to replicate the decision-making behavior of a human being. The trained model only responds to the real-time detected environmental information and does not have to perform any optimization procedures. To

determine the feasibility of this neural network based method, a simulator was established in Python to present an exemplary mobile robot navigation task. The tuning procedures using supervised learning were made on the structure of the neural network and the parameters of the training algorithm to guarantee the high accuracy of this method with respect to the training and testing data sets. To validate its performance, the trained model was applied to the control system of the mobile robots in the simulator multiple times. The resulting paths confirmed its ability to handle navigation tasks and achieve obstacle avoidance as well. The scientific and technical contributions of this paper are highlighted as follows:

- A autonomous mobile robot navigation problem was formulated with a finite number of motion types, requirements on obstacle avoidance in 2D space, and no global environmental information.
- An implementation algorithm is provided on how to process the obtained data and train a neural network model using a supervised learning method to approach the human decision-making process.
- A simulator was established in the Python programming environment to model each step of an exemplary mobile robot navigation task with visualization outcomes, and the human decision data were collected under the same navigation scenario of the simulator.
- The proposed algorithm used the data collected from the human decision-making process under the same navigation scenario to train a neural network model to achieve high estimation accuracy rates.
- The trained neural network model was validated in the simulator to perform the required task with the desired performance, which solves the real-time computation issue.

The nomenclature used in this paper is listed in Abbreviations part.

2. Problem Formulation

In this section, an autonomous mobile robot navigation task is first described, and then its design objectives are given. Together, these form a navigation problem with obstacle avoidance.

2.1. Navigation Task Description

The task operation environment within a mobile robot system is a 2D plane with random generated obstacles as shown in Figure 1, for example. In this figure, the red dot and green dot are denoted as the start position (x_0, y_0) and the destination position (x^d, y^d) in the 2D plane, respectively. This task requires the mobile robot to start from the red dot and then reach the green dot without colliding with any obstacles. To perform this task, an omnidirectional mobile robot with a state vector

$$s(t) = [x(t) \ y(t) \ \theta(t)]^T \quad (1)$$

at each sample decision-making time index t is considered in this paper, where $x(t)$ and $y(t)$ are the horizontal and vertical coordinates of the mobile robot, and $\theta(t)$ is the heading direction of the mobile robot. The motion of this vehicle is conducted by a finite number of motion types $\mathcal{M}_i, i = 1, \dots, n$. Here, each motion type should be linked to an implementable action of the mobile robot, e.g., go ahead, turn left or turn right, and associate with a particular state change function

$$s(t+1) = \mathcal{F}_i(s(t)), \ i = 1, \dots, n. \quad (2)$$

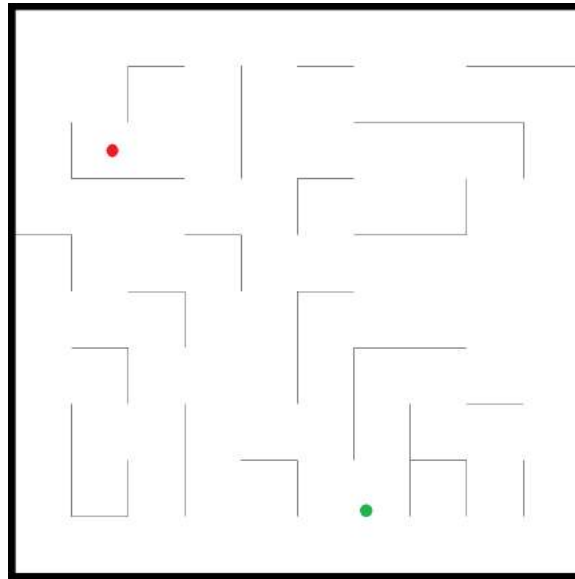


Figure 1. An exemplary operation environment of the mobile robot navigation task.

Since the motion function is simplified to motion types, it enables this task to be carried out by sequentially making appropriate decisions of motion types at each sample time index t . To help the achievement of the latter design, two assumptions are made as below.

Assumption 1. *The motion types of this mobile robot guarantee it to reach any feasible position within the given 2D plane.*

Assumption 2. *The motion types must differ significantly from each other, i.e., for a given state s , there exists*

$$\|\mathcal{F}_i(s) - \mathcal{F}_j(s)\| \geq \sigma, \text{ for any } i, j, \quad (3)$$

where σ is a nonnegative scalar.

Assumption 1 makes sense in practice, since it guarantees the mobile robot task achievement, as there is no need to design any autonomous mobile robot navigation approach for an impossible task. Assumption 2 ensures all motion types will be different, which increases the success rate of classifying the appropriate motion type using a neural network in Section 3.

2.2. Task Design Objective and Problem Definition

To interact with the external environment in the mobile robot system, various sensors are installed on the mobile robot to detect real time useful information. This paper uses the detection mechanism as shown in Figure 2 for visual representation, which means only local environmental information is available. It enables the mobile robot to have a total number of m laser distance sensors to detect whether there are any obstacles around it and how far away the obstacle is from its current position, and this information is denoted as $d_i(t)$, $i = 1, \dots, m$.

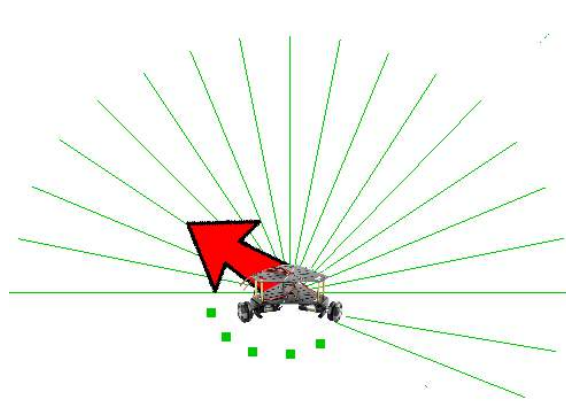


Figure 2. A visual example of the detected local environmental information of the mobile robot.

Note that the green lines in the figure are the laser beams, which will reflect after touching an obstacle. These laser beams can detect the obstacles within a maximum distance of d_{max} ; hence, there exists $0 \leq d_i(t) \leq d_{max}$. Furthermore, a location sensor is incorporated to calculate the bias of the heading direction, which is the red arrow in the figure, and is denoted as $\phi(t)$ satisfying $-180^\circ < \phi(t) \leq 180^\circ$.

In this sense, the design objective of the autonomous mobile robot navigation task is to develop a mechanism, which makes the vehicle respond to the real time environmental information in order to complete the task. The navigation problem discussed in this paper is explained in the next definition.

Definition 1. *The autonomous mobile robot navigation problem with obstacle avoidance aims to make appropriate decisions of the given motion types based on the real-time detected information $d_i(t)$, $i = 1, \dots, m$ and $\phi(t)$ at each sample time index t , such that the vehicle moves from its start position (x_0, y_0) to a position (x^*, y^*) close to its destination position (x^d, y^d) without colliding with any single obstacle, i.e.,*

$$\|(x^*, y^*) - (x^d, y^d)\| \leq \epsilon, \quad (4)$$

where ϵ is a small positive scalar determining an acceptable distance from the destination to complete the navigation task.

The above definition articulates a nontrivial decision-making problem, which is not deterministic or straightforward to solve. Therefore, this paper will employ the tools of a neural network and supervised learning to address this problem in Section 3 in the sense of replicating the human decision process using obtained data. Note that the human decision-making process is the procedure where human drivers make the motion decisions when in the same situation, i.e., with the same local information. In this process, the human drivers would try their best to reach the destination via a relatively short path without colliding with the surrounding obstacles.

3. Process for Training a Neural Network

This section describes the extraction of the useful information from the decisions made by human beings in the identical scenario to the navigation problem in Definition 1. Then, we formulate this problem into a supervised learning problem and exploit the information to train a neural network model to replicate the learned human behavior in terms of decision making using the recorded data.

3.1. User Data Collection and Problem Reformulation

In the identical scenario to the navigation problem in Definition 1, human users were asked to make a decision based on the observed scenes at each occasion. First of all, the users were only able to observe the space within the detection range of the sensor, where the distance between the robot and obstacles as well as the direction to the destination point were provided. Then the users judged how the robot should move based on the known information and controlled its movement to the next space through the keyboard. Note that to reduce the correlation of each datum in time and space, the position and orientation of the robot were randomly placed after performing an action. The decisions made by these users were recorded for a total number of N occasions. On the j -th occasion, a set of labeled data was generated and stored in a database Λ as

$$\lambda_j = [d_1^j \ d_2^j \ \dots \ d_m^j \ \phi^j \ l^j], \quad j = 1, \dots, N, \tag{5}$$

where j is the occasion index number, $d_i^j, i = 1, \dots, m$, are the obstacle distances detected by the laser distance sensors, ϕ^j is the bias of the heading direction measured by the location sensor, and l^j is the label of human made decision \mathcal{M}_i , i.e.,

$$l^j \in \{1 \ 2 \dots \ n\}. \tag{6}$$

To ensure the collected data were meaningful, an assumption was made as follows:

Assumption 3. *The human users tried their best to achieve the given navigation task, and they were capable of successively completing this task, i.e., they did not aim to make the mobile robot collide with an obstacle or move toward the wrong destination.*

Based on the above assumption, it is acknowledged that the navigation problem in Definition 1 can be solved by a mechanism, which replicates the human decision-making process. Therefore, using the stored labeled data, the navigation problem can be transform into a classification task. As explained in [34], after obtaining the data set Λ , η elements were randomly selected to form the training set Λ_{train} , and the rest were regarded as the test set Λ_{test} . An $(m + 1)$ -input and n -output neural network model was employed to obtain the probability of each navigation action, whose input was $[d_1^i \ d_2^i \ \dots \ d_m^i \ \phi^i] \in \mathbb{R}^{m+1}$ and output was $z_i \in \mathbb{R}^n$. As a result, the robot navigation task was reformulated as a supervised learning problem defined below.

Definition 2. *The Navigation Decision-Making Problem is to iteratively update the parameters of network model using the elements cut into the training set Λ_{train} from collected data, such that the cross-entropy loss function*

$$\mathcal{L}(z_i, \hat{z}_i) = - \sum_{j=1}^n \hat{z}_i \log z_i(j), \quad i = 1, \dots, N \tag{7}$$

is minimized for all N labeled datasets, where $\hat{z}_i \in \mathbb{R}^n$ is the expected output defined by l^i as

$$\hat{z}_i(j) = \begin{cases} 1, & j = l^i, \\ 0, & \text{else.} \end{cases} \tag{8}$$

The cross-entropy loss function (7) was utilized to evaluate the loss between the output of the neural network and the human behavior label. The weights of the neural network were continuously adjusted to reduce the loss value through the error back-propagation algorithm. When the cross entropy loss became small enough, the neural network model was considered to completely fit the mapping relationship between input and output.

3.2. Neural Network Structure

The structure of the neural network affects the final training results, so it should be carefully and reasonably designed. The network was fully connected, as shown shown in Figure 3. The input layer was composed of $(m + 1)$ neurons, which was represented as the vector

$$z_0^{in} = [d_1^i \ d_2^i \ \dots \ d_m^i \ \phi^i]^T. \tag{9}$$

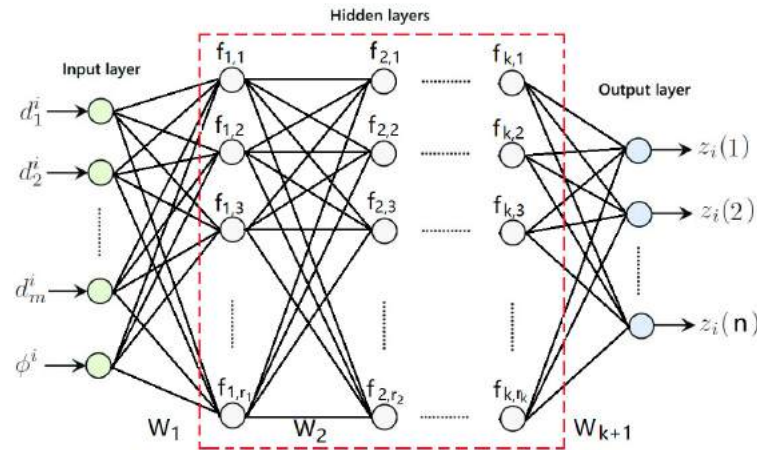


Figure 3. The structure of the neural network model.

Since the data values may vary in rather different ranges, a rescaling step was performed at the input layer such that the output of the input layer was computed as the vector

$$z_0^{out} = [d_1^i/d_{max} \ d_2^i/d_{max} \ \dots \ d_m^i/d_{max} \ \phi^i/360]^T. \tag{10}$$

This rescaling step adjusted the input values to a relatively similar level, overcoming the potential slow convergence problem, which would occur at the large-scale dimension.

As shown in Figure 3, the model included k hidden layers in the dashed red box, and the i -th hidden layer had r_i units. The output of the i -th layer was a vector $z_i^{out} \in \mathbb{R}^{r_i}$ and was transferred to the input $z_{i+1}^{in} \in \mathbb{R}^{r_{i+1}}$ of the $(i + 1)$ -th layer as

$$z_{i+1}^{in} = W_{i+1} z_i^{out}, \tag{11}$$

where $W_{i+1} \in \mathbb{R}^{r_{i+1} \times r_i}$ is a weight matrix. An activation function $f_{i,j}(\cdot)$ was applied before the network output of each layer, which is defined as

$$z^{out} = f_{i,j}(z^{in}) = \text{RReLU}(z^{in} + b_{i,j}), \tag{12}$$

where $\text{RReLU}(\cdot)$ is called a randomized rectified liner unit function defined as

$$\text{RReLU}(x) = \begin{cases} x, & x \geq 0, \\ ax, & \text{else.} \end{cases} \tag{13}$$

In Equation (13), coefficient a is randomly sampled from a uniform distribution. The function $\text{RReLU}(\cdot)$ serves is that of a nonlinear factor to solve problems that cannot be solved by linear models. Compared to other activation function such as Sigmoid, it saves computation load during the backward updating process of the neural network.

The output corresponds to the possibility of each label, so the input vector of the output layer $z_{k+1}^{in} \in \mathbb{R}^n$ is processed by

$$z_{k+1}^{out} = g(z_{k+1}^{in}), \tag{14}$$

where $g(\cdot)$ is a softmax function defined as

$$g(x)_i = \frac{e^{x(i)}}{\sum_{j=1}^n e^{x(j)}}. \tag{15}$$

3.3. A Supervised Learning Algorithm

Given the obtained data set, Definition 3.1 is transformed into a supervised learning task, which is solved by Algorithm 1. Firstly, we determined the relevant parameters of the supervised learning algorithm, such as learning rate, iteration times, batch size, etc. Secondly, we randomly initialized the weights W_i and thresholds $b_{i,j}$. Thirdly, to balance memory efficiency and memory capacity, the training data set was divided into several batches of elements. Fourthly, we started an iterative process in which the network parameters were continuously updated until the loss value dropped to an acceptable value.

Algorithm 1 Training process for neural network model

Input: Collected human database Λ , cross entropy loss function (7), initial network weights W_i and thresholds $b_{i,j}$, total number of training data η , batch size n_{batch} , and iteration epoch number n_{epoch} .

Output: A trained model with W_i and $b_{i,j}$.

- 1: **initialization:** Set training epoch number $i = 1$.
 - 2: Randomly select η elements of Λ to a training set Λ_{train} and the rest to a testing set Λ_{test} .
 - 3: Cut Λ_{train} into $\text{int}(\eta/n_{batch})$ batches of elements with batch size n_{batch} .
 - 4: **while** $i \leq n_{epoch}$ **do**
 - 5: **for** $j \leq \text{int}(\eta/n_{batch})$ **do**
 - 6: Calculate the sum of the cross entropy loss (7) of all data in the j -th batch.
 - 7: Compute the gradients of W_i and $b_{i,j}$ with respect to the obtained sum.
 - 8: Perform Adaptive Moment Estimation (Adam) gradient descent training to update W_i and $b_{i,j}$ according to loss back-propagation algorithm.
 - 9: **end for**
 - 10: Calculate the accuracy rates of the updated model in Λ_{train} and Λ_{test} by Equation (16).
 - 11: Set $i = i + 1$ to the next iteration.
 - 12: **end while**
 - 13: **return** The trained model with W_i and $b_{i,j}$.
-

To evaluate the training effect of the neural network, the output accuracy of the training network on Λ_{train} and Λ_{test} were computed as

$$\phi_{train} = \frac{\sum_{\lambda \in \Lambda_{train}} \lambda^{acc}}{\text{len}(\Lambda_{train})}, \quad \phi_{test} = \frac{\sum_{\lambda \in \Lambda_{test}} \lambda^{acc}}{\text{len}(\Lambda_{test})}, \tag{16}$$

where λ_i is a mark used to determine whether the character corresponding to the maximum value of neural network output was consistent with the label as

$$\lambda_i^{acc} = \begin{cases} 1, & \arg \max z_i = l^i, \\ 0, & \text{else.} \end{cases} \tag{17}$$

Remark 1. Since the elements in testing set Λ_{test} and the training set Λ_{train} are completely independent of each other, the evaluation results are able to examine the training performance in terms of overfitting problems.

4. Mobile Robot Simulator Platform

To validate the performance of Algorithm 1, a mobile robot simulator was developed using the Pygame package in Python environment. This simulator can generate a 2D plane

with random obstacles as that shown in Figure 1. It enables the vehicle to perform three motion types, such as go straight ahead, rotate to the left, and rotate to the right, i.e., $m = 3$, and the state change functions are defined as

$$\begin{aligned} \mathcal{F}_1(s(t)) &= [x(t) + v_m \cos(\theta(t)) \ y(t) + v_m \sin(\theta(t)) \ \theta(t)]^\top, \\ \mathcal{F}_2(s(t)) &= [x(t) \ y(t) \ \theta(t) + v_r]^\top, \\ \mathcal{F}_3(s(t)) &= [x(t) \ y(t) \ \theta(t) - v_r]^\top, \end{aligned} \tag{18}$$

where v_m is the motion speed, and v_r is the rotation speed. Note that the values of v_m and v_r can be customized in the configuration window of the simulator.

This simulator can generate various occasions with random obstacles and collect the decisions made by the human test candidates, and an exemplary situation is shown in Figure 4. As can be seen from this figure, there were 17 laser detectors, i.e., $m = 17$, installed on the front half plane of the mobile robot. This simulator replicated a radar with 5 m measuring distance, and each pixel in the screen denoted 0.01 m. Therefore, the maximum detection distance was set as $d_{max} = 500$ in units of pixels. Since the laser detectors can only measure the information in the front 180° direction, the bottom half of the circular sight in this figure was covered when recording the human user’s decisions to provide a fair navigation environment. The color c_i of the i -th laser beam followed the mechanism

$$c_i = \begin{cases} \text{Green,} & 0.6d_{max} < d_i \leq d_{max}, \\ \text{Yellow,} & 0.3d_{max} < d_i \leq 0.6d_{max}, \\ \text{Red,} & 0 < d_i \leq 0.3d_{max}, \end{cases} \tag{19}$$

which alerted the human test candidates to potential nearby obstacles. Note that only a circular sight is shown in Figure 4, and it represents the exact region the mobile robot could detect using its laser sensors. The obtained decisions were recorded as the label l^i , using the reference equation

$$l^i = \begin{cases} 0, & \text{Go ahead,} \\ 1, & \text{Rotate to left,} \\ 2, & \text{Rotate to right.} \end{cases} \tag{20}$$

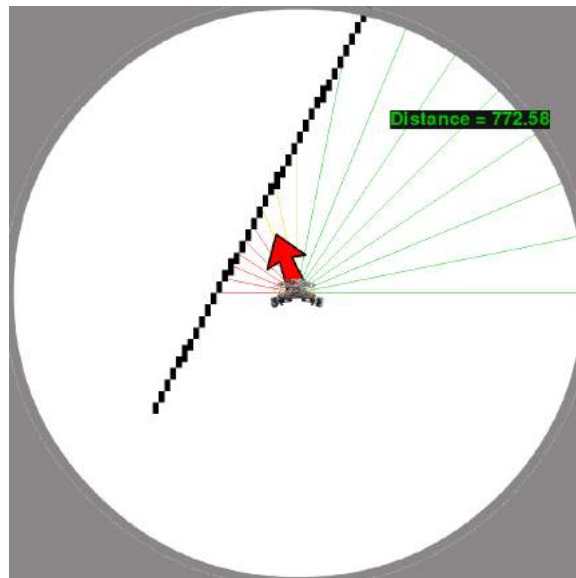


Figure 4. A straightforward example of user data collection.

The obtained labels were saved in the database together with the corresponding environmental information $d_i, i = 1, \dots, 17$ and ϕ^i , which formed a complete labeled element.

Remark 2. *The generated occasions only provide the human test candidates with the exact information that the mobile robot can have. Therefore, with both having the same scope of obtained environmental information, a neural network model can be trained to replicate the human decision-making process in a fair manner.*

Remark 3. *This simulator is capable of accurately modeling a navigation task in a 2D plane with random obstacles, and each human labeling process can be completed within 1 s. However, while using the automated algorithms, e.g., reinforcement learning, on a similar experimental established test platform in practice, it took time to build a limited range of real navigation scenarios. The mobile robot should not move too fast, i.e., less than 0.2 m/s, to avoid safety issues, which means they might take more than 10 s to obtain useful data. Therefore, the human labeling approach using this simulator has the advantages of data collection speed and data diversity.*

Remark 4. *Note that the algorithm performance was affected by the number m of laser sensors, as the knowledge of the surrounding environmental information definitely increases with more sensors. In this sense, more detailed information helps the neural network to make more appropriate decisions. However, the implementation space of sensors is restricted on a remote robot, which means the number m cannot be too large.*

5. Numerical Simulation Results

This section describes the performance of Algorithm 1 to train a neural network model based on the collected labeled data obtained from the simulator. The trained model was applied to the mobile robot to serve its navigation module in the simulator, and the corresponding results are presented and discussed.

5.1. Design Parameter Specifications

The parameter setting in Algorithm 1 directly determines whether the ideal network model can be obtained. In this paper, the model was specified as a five-layer fully connected neural network, and the neuron numbers within each hidden layer were set as

$$r_1 = 64, r_2 = 128, r_3 = 64, r_4 = 32, r_5 = 8, \quad (21)$$

respectively. Theoretically, deepening the neural network improves the training effect, but it consumes more computing resources and increases the risk of overfitting. As shown in Figure 5a, net2 and net3 deepened the number of layers and increased the number of neurons in a certain layer on the basis of net1, but their accuracy in the test set was not improved or even reduced, and the time required for training increased. Therefore, it was considered appropriate to set the neural network to 5 layers. The learning rate determines the convergence speed of neural network, which has a great impact on the training of neural network. Too small a learning rate leads to the slow convergence speed of the neural network and increased training time while too large a learning rate results in missing the optimal solution. As shown in Figure 5b, setting the learning rate to 0.0008 was considered to be a suitable scheme that accounted for both the training speed and the optimal solution. Furthermore, the batch size and the total number of iterations were selected as

$$n_{batch} = 10, n_{epoch} = 1000, \quad (22)$$

which affected the total time spent in the whole training process. During the data collection phase, 800 labeled elements were collected into data set Λ , and 640 were randomly extracted to generate the training set Λ_{train} , i.e., $n = 640$.

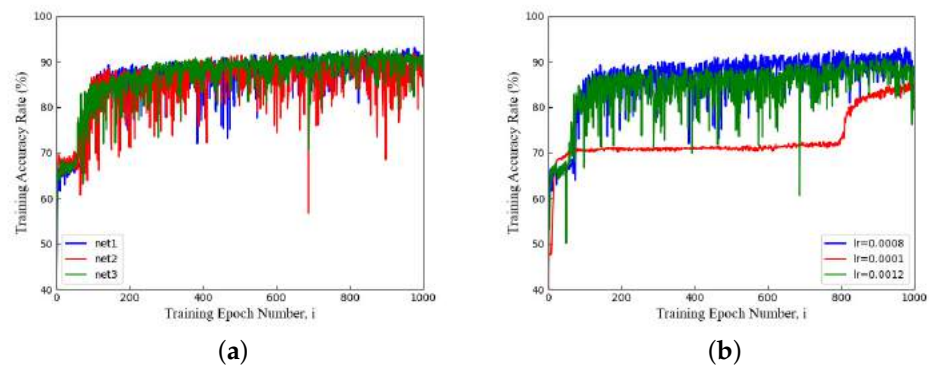


Figure 5. Different layers, neurons, and learning rates affect the accuracy of the neural network replicating human decision-making process. (a) neural network; (b) learning rate.

5.2. Training Performance Analysis

The accuracy rate is an index to judge the ability of the neural network to replicate human behavior. Therefore, the accuracy rate in Λ_{train} and Λ_{test} was calculated by (16) and (17) at the end of each iteration, as shown in Figure 6. The accuracy increased with the number of iterations and tended to be stable after 900 epochs, which proves that the network parameters were continuously updated and finally converged. The accuracy rate in Λ_{train} exceeded 90% while it reached 85% in Λ_{test} , which confirmed that the trained model was theoretically capable of making a decision similar to a human being based on the input information. Other alternative design parameter specifications were also tested and the final estimation accuracy rates were either lower than or similar to the above results.

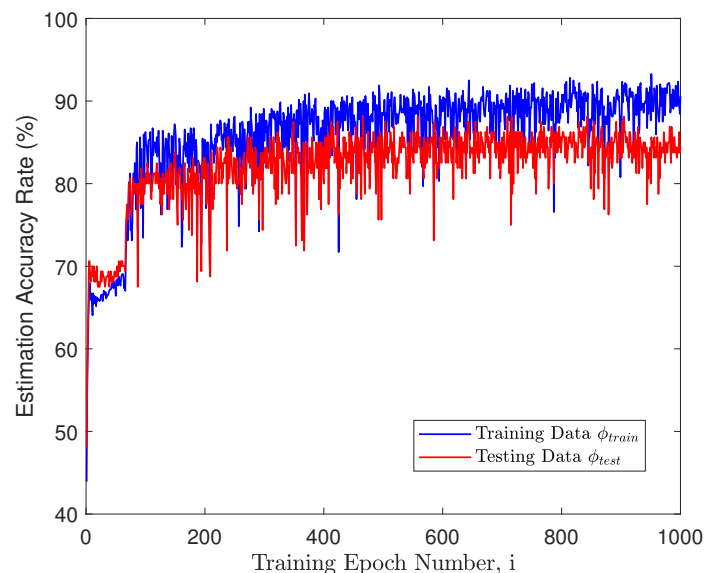


Figure 6. The estimation accuracy rates of Algorithm 1 with respect to Λ_{train} and Λ_{test} along the training epochs.

To determine why the accuracy rate could not further increase, the dimensionality reduction method called t-Distributed Stochastic Neighbor Embedding (t-SNE) was applied to convert the high dimensional elements λ_i in the database Λ into a 2D plane for a straightforward point of view. The results are shown in Figure 7, and it is clear that the data points in some particular regions had more than one label. Therefore, the data structure was not sparse enough to allow the trained model to reach a high estimation accuracy, and it was difficult for the model to identify the data type in those regions. This is due to the fact that there inevitably exist some occasions in which the human test candidates cannot make

deterministic decisions. Therefore, the obtained performance of Algorithm 1 is acceptable in the autonomous mobile robot navigation scenario.

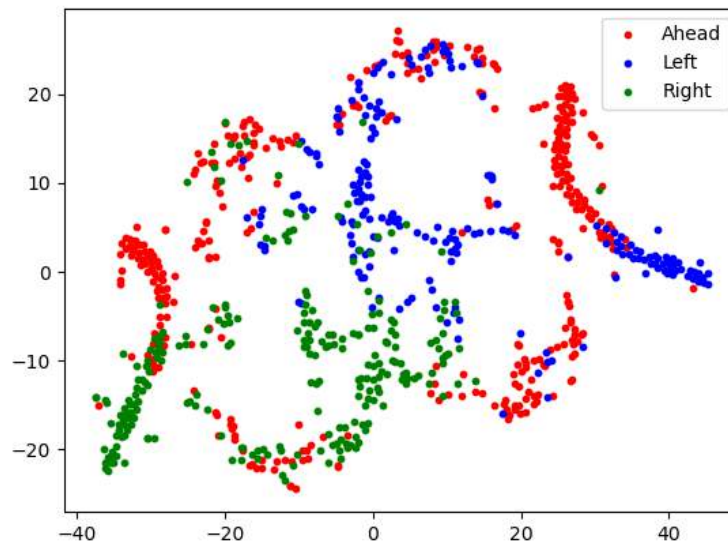


Figure 7. 2D plot of input data with marked labels using t-SNE.

5.3. Verification on the Simulator

Once the neural network converged, the decision model of the mobile robot was determined, which will not change with respect to propagation of scene and time. To validate its actual performance, the trained neural network model was applied to the autonomous mobile robot navigation task in the simulator. The simulator randomly generated a 2D plane with obstacles and selected the start and destination positions, which are marked as red and green dots in Figure 8. At each sample time index t , the measured obstacle distance $d_i(t), i = 1, \dots, 17$ and heading direction bias ϕ^i were sent to the trained model to generate the probabilities of each motion type \mathcal{M}_i at t , which was exactly the output of the model. Based on the obtained probabilities, the decision of the motion type was computed and the mobile robot changed its state according to (18) in the simulator test environment.

To evaluate the real-time performance of the neural network model, it was loaded onto a laptop with “AMD Ryzen7 5800H” CPU and “NVIDIA GeForce RTX 3060” GPU to obtain the time consumed by each decision as shown in Table 1. Note that the total time was consumed by the process of the model outputting an executed action based on the input information obtained. In Table 1, a total number of 5 navigation tasks were considered, where the time consumed by the model at 6 moments was randomly selected to compute the average time spent on executing an action. The neural network model took only approximately 1 millisecond to make a decision. In the simulator, the mobile robot’s state was represented by its pose $(x(t), y(t), \theta(t))$. Hence, the neural network performed real-time computation to choose a motion type based on acquired data, and a new state was obtained after executing an action. Connecting the coordinates of all occasions, a path from the starting point to the destination point was formed.

Table 1. Consumed time to output an action by the model.

Number	Consumed Time (s)						Average
	a_1	a_2	a_3	a_4	a_5	a_6	
1	0.00123	0.00100	0.00100	0.00100	0.00119	0.00100	0.00107
2	0.00100	0.00099	0.00100	0.00099	0.00100	0.00100	0.00100
3	0.00100	0.00099	0.00200	0.00100	0.00100	0.00100	0.001165
4	0.00100	0.00100	0.00100	0.00099	0.00100	0.00100	0.00100
5	0.00100	0.00100	0.00100	0.00200	0.00099	0.00100	0.001165

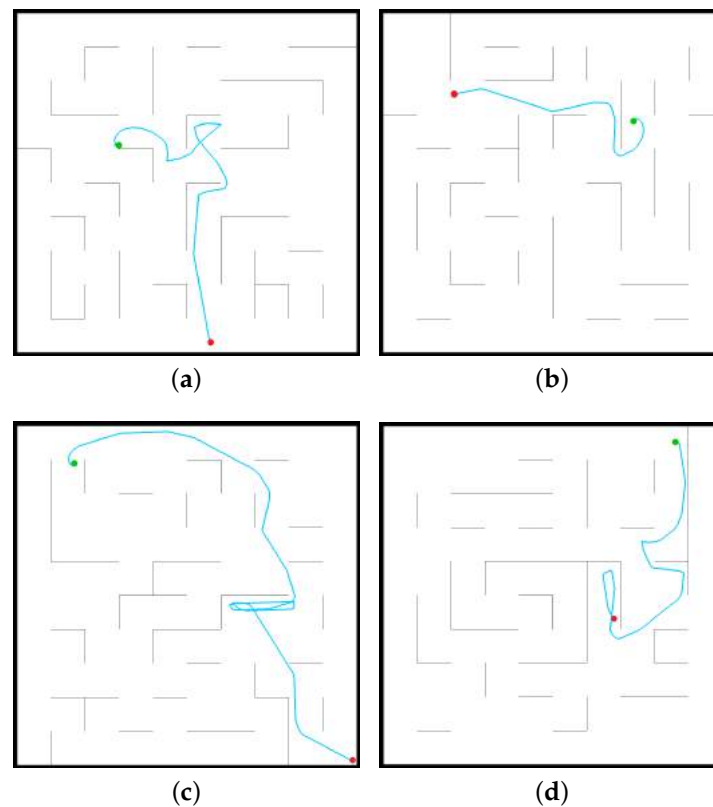


Figure 8. The paths of the mobile robot while performing navigation tasks using the trained neural network model in the simulator. (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4.

The paths of the mobile robot in some exemplary cases are shown in Figure 8. The results demonstrate the appealing properties of the neural network based navigation method proposed in this paper, in that the mobile robot could arrive at the destination without colliding with any obstacles. The navigation performance in other cases was similar to those in Figure 8. Note that all the robot knows at a certain position is sensor information, so that it has no ability to make the optimal decision as in other global planning methods. However, compared with other approaches, the proposed method did not consume high computing resources to obtain the global environment information and was capable of responding to dynamic changes in time. As shown in Figure 8a,c,d, although the path is not the shortest, the robot safely moved from the starting point to the destination position through its own adjustment ability.

The neural network was employed to replicate human behavior, but the planned paths shown in Figure 8 seemed not to conform to human thinking in theory. This is mainly due to two reasons: on the one hand, people usually combine the entire scene map to judge the integral path from a global perspective. However, when the information available to humans is reduced to a local area, the decision made is similar to the neural network. On the other hand, humans utilize the brain's memory to assist in decision making, which is unavailable to the neural network proposed in this paper. This limitation may cause the robot to spin around in a certain area. Note that the spinning of the robot as shown in Figure 8c is a local optimum caused by the proposed method. However, since the choice of action conforms to the probability distribution output by the neural network, the robot may choose another action at the same location, which helps it out of the current predicament.

5.4. Discussion

To show the effectiveness of the proposed algorithm in autonomous mobile robot navigation, it was compared with the artificial potential field (APF) method commonly used in a 2D environment, which usually forms a shorter path. As shown in Figure 9a,b,

the path that is formed by Algorithm 1 proposed in this paper was only slightly longer than the path obtained by the artificial potential field method in the second half without knowing the global environment information, but the generated path was smoother than that of the APF method. Furthermore, the APF method has the probability to fall into the local minimum solution as shown in Figure 9c, while the method based on the neural network is capable of bypassing obstacles of various shapes ahead, as shown in Figure 9d.

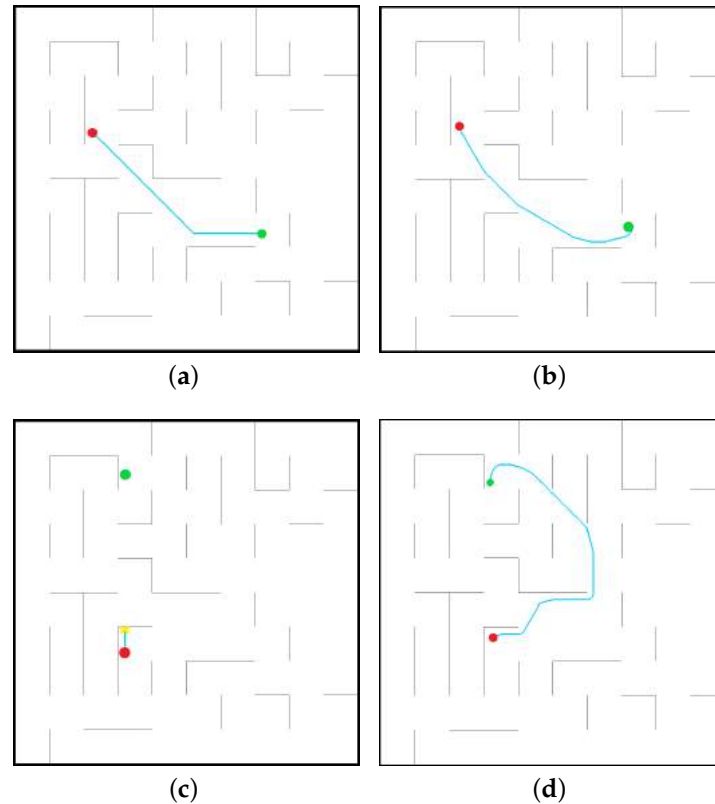


Figure 9. The paths of the mobile robot while performing navigation tasks using the trained neural network model in the simulator. (a) APF; (b) proposed; (c) APF; (d) proposed.

Meanwhile, the dynamic window approach (DWA) is a classic local path planning algorithm. When the robot reaches a certain position, DWA predicts the trajectory of each set of speeds in a limited dynamic window and gives a speed evaluation, and the set of speeds with the highest evaluation is selected as the current speed for execution. Finally, the path planning is transformed into the problem of maximizing the objective function, but real-time computing consumes high computing resources, around 100 times more than the proposed algorithm. Compared to the DWA, the advantage of the method proposed in this paper is that the computation required for neural network training is offloaded to the computer to reduce the requirement on the robotic hardware. When the network is transplanted to the robot, it only needs to perform a few matrix operations to obtain the corresponding action.

6. Conclusions and Future Work

This paper employed a neural network model to address the class of autonomous mobile robot navigation problems in mobile robot systems with the existence of obstacles. It focused on the scenario where the mobile robot only had a finite number of motion types and no global environmental information. Human-made decisions on these motion types can be collected together with the corresponding environmental information on various occasions including random generated obstacles. In this sense, the supervised learning method was employed to yield an algorithm to train a neural network model to replicate the

human decision process using recorded data. To validate the performance of the proposed algorithm and the trained model, a mobile robot simulator was developed to collect the data from the human decision process as well as perform a representative navigation task in a 2D plane. The simulation results illustrated that a high estimation accuracy rate of nearly 90% with respect to the human decision-making process was obtained using the proposed algorithm, and the trained model successfully completed the given tasks. Compared with the APF method, the neural network approach was able to handle the navigation task with obstacles without the local minimum problem, which proves the effectiveness of neural network decision.

For future work, the trained model will be applied to an experimental test platform to further examine its practical performance, and the iterative learning control method in [35–37] can be implemented to guarantee the practical performance of the repetitive motion types. It is undeniable that the neural network only considers the current state of the robot for decision making, which leads to the certainty of global judgment. A navigation algorithm with a short memory will be considered to be applied to further improve its performance to handle the circular motion problem as shown in Figure 8c. Meanwhile, reinforcement learning methods can be employed to let the robot train itself by gaining a reward when approaching the destination or receiving punishment when colliding with an obstacle. Last but not least, the proposed method can be extended to a distributed version to handle the navigation task consisting of a number of mobile robots.

Author Contributions: Conceptualization, Y.C., Y.Z. and X.L.; methodology, Y.C. and C.C.; software, Y.C. and C.C.; validation, Y.C. and C.C.; formal analysis, Y.C., Y.Z. and X.L.; investigation, C.C.; resources, Y.C., Y.Z. and X.L.; data curation, Y.C.; writing—original draft preparation, Y.C.; writing—review and editing, Y.Z. and X.L.; visualization, C.C.; supervision, L.S.; project administration, Y.C.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 62103293, the Natural Science Foundation of Jiangsu Province grant number BK20210709, the Suzhou Municipal Science and Technology Bureau grant number SYG202138, and the Entrepreneurship and Innovation Plan of Jiangsu Province grant number JSSCBS20210641.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Abbreviations	Description
CUDA	Compute unified device architecture
GPU	Graphics processing unit
t-SNE	t-Distributed Stochastic Neighbor Embedding
Symbols	Description
\mathbb{R}^n	is an n-dimensional Euclidean vector space
$\mathbb{R}^{m \times n}$	is an $m \times n$ real matrix space
\mathcal{M}_i	is the i -th motion type of a mobile robot
$\mathcal{F}_i(\cdot)$	is the i -th state change function of a mobile robot
$\mathcal{L}(\cdot)$	is the cross entropy loss function
t	is the sample decision-making time index
Λ_{train}	is the training data set
Λ_{test}	is the testing data set

References

1. Gong, Y.; Zhang, L.; Liu, R.P.; Yu, K.; Srivastava, G. Nonlinear MIMO for Industrial Internet of Things in Cyber-Physical Systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5533–5541. [[CrossRef](#)]
2. Schulman, J.; Duan, Y.; Ho, J.; Lee, A.; Awwal, I.; Bradlow, H.; Pan, J.; Patil, S.; Goldberg, K.; Abbeel, P. Motion planning with sequential convex optimization and convex collision checking. *Int. J. Robot. Res.* **2014**, *33*, 1251–1270. [[CrossRef](#)]
3. Rezaee, H.; Abdollahi, F. A Decentralized Cooperative Control Scheme with Obstacle Avoidance for a Team of Mobile Robots. *IEEE Trans. Ind. Electron.* **2014**, *61*, 347–354. [[CrossRef](#)]
4. Ji, J.; Khajepour, A.; Melek, W.W.; Huang, Y. Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints. *IEEE Trans. Veh. Technol.* **2017**, *66*, 952–964. [[CrossRef](#)]
5. Cheng, C.; Chen, Y. A neural network based mobile robot navigation approach using reinforcement learning parameter tuning mechanism. In Proceedings of the IEEE 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021.
6. Goerzen, C.; Kong, Z.; Mettler, B. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *J. Intell. Robot. Syst.* **2010**, *57*, 65–100. [[CrossRef](#)]
7. Campbell, S.; Naeem, W.; Irwin, G.W. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annu. Rev. Control* **2012**, *36*, 267–283. [[CrossRef](#)]
8. Galceran, E.; Carreras, M. A survey on coverage path planning for robotics. *Robot. Auton. Syst.* **2013**, *61*, 1258–1276. [[CrossRef](#)]
9. Meng, D.; Xiao, Y.; Guo, Z.; Jolfaei, A.; Qin, L.; Lu, X.; Xiang, Q. A Data-driven Intelligent Planning Model for UAVs Routing Networks in Mobile Internet of Things. *Comput. Commun.* **2021**, *179*, 231–241. [[CrossRef](#)]
10. Khansari-Zadeh, S.M.; Billard, A. A dynamical system approach to realtime obstacle avoidance. *Auton. Robot.* **2012**, *32*, 433–454. [[CrossRef](#)]
11. Chu, K.; Lee, M.; Sunwoo, M. Local Path Planning for Off-Road Autonomous Driving with Avoidance of Static Obstacles. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 1599–1616. [[CrossRef](#)]
12. Alonso-Mora, J.; Naegeli, T.; Siegwart, R.; Beardsley, P. Collision avoidance for aerial vehicles in multi-agent scenarios. *Auton. Robot.* **2015**, *39*, 101–121. [[CrossRef](#)]
13. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
14. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*; Springer: Berlin, Germany, 2001.
15. Nasrabadi, N.M. Pattern recognition and machine learning. *J. Electron. Imaging* **2007**, *16*, 049901.
16. Sun, Y.; Liu, J.; Yu, K.; Alazab, M.; Lin, K. PMRSS: Privacy-preserving Medical Record Searching Scheme for Intelligent Diagnosis in IoT Healthcare. *IEEE Trans. Ind. Inform.* **2021**, *18*, 1. [[CrossRef](#)]
17. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 2012 International Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
18. Weng, G.; Dong, B.; Lei, Y. A level set method based on additive bias correction for image segmentation. *Expert Syst. Appl.* **2021**, *185*, 115633. [[CrossRef](#)]
19. Ge, P.; Chen, Y.; Wang, G.; Weng, G. A hybrid active contour model based on pre-fitting energy and adaptive functions for fast image segmentation. *Pattern Recognit. Lett.* **2022**, *158*, 71–79. [[CrossRef](#)]
20. Zhang, X.; Yang, L.; Ding, Z.; Song, J.; Zhai, Y.; Zhang, D. Sparse Vector Coding-based Multi-Carrier NOMA for In-Home Health Networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 325–337. [[CrossRef](#)]
21. Chen, Y.; Zhou, Y. Machine learning based decision making for time varying systems: Parameter estimation and performance optimization. *Knowl.-Based Syst.* **2020**, *190*, 105479. [[CrossRef](#)]
22. Chen, Y.; Zhou, Y.; Zhang, Y. Machine Learning-Based Model Predictive Control for Collaborative Production Planning Problem with Unknown Information. *Electronics* **2021**, *10*, 1818. [[CrossRef](#)]
23. Chen, Y.; Jiang, W.; Charalambous, T. Machine learning based iterative learning control for non-repetitive time-varying systems. *Int. J. Robust Nonlinear Control* **2022**, *Early View*. [[CrossRef](#)]
24. Tao, H.; Wang, P.; Chen, Y.; Stojanovic, V.; Yang, H. An unsupervised fault diagnosis method for rolling bearing using STFT and generative neural networks. *J. Franklin Inst.* **2020**, *357*, 7286–7307. [[CrossRef](#)]
25. Jaradat, M.A.K.; Al-Rousan, M.; Quadan, L. Reinforcement based mobile robot navigation in dynamic environment. *Robot. Comput.-Integr. Manuf.* **2011**, *27*, 135–149. [[CrossRef](#)]
26. Duguleana, M.; Mogan, G. Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Syst. Appl.* **2016**, *62*, 104–115. [[CrossRef](#)]
27. Zhu, K.; Zhang, T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Sci. Technol.* **2021**, *26*, 674–691. [[CrossRef](#)]
28. Li, H.; Zhang, Q.; Zhao, D. Deep Reinforcement Learning-Based Automatic Exploration for Navigation in Unknown Environment. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 2064–2076. [[CrossRef](#)] [[PubMed](#)]
29. Xiao, W.; Yuan, L.; He, L.; Ran, T.; Zhang, J.; Cui, J. Multigoal Visual Navigation with Collision Avoidance via Deep Reinforcement Learning. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–9. [[CrossRef](#)]
30. Ni, Z.; He, H.; Wen, J.; Xu, X. Goal Representation Heuristic Dynamic Programming on Maze Navigation. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 2038–2050. [[PubMed](#)]

31. Kim, C.J.; Chwa, D. Obstacle Avoidance Method for Wheeled Mobile Robots Using Interval Type-2 Fuzzy Neural Network. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 677–687. [[CrossRef](#)]
32. Talbot, B.; Dayoub, F.; Corke, P.; Wyeth, G. Robot Navigation in Unseen Spaces Using an Abstract Map. *IEEE Trans. Cogn. Dev. Syst.* **2021**, *13*, 791–805. [[CrossRef](#)]
33. Yuan, W.; Li, Z.; Su, C.Y. Multisensor-Based Navigation and Control of a Mobile Service Robot. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 2624–2634. [[CrossRef](#)]
34. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: Berlin, Germany, 2013.
35. Chen, Y.; Chu, B.; Freeman, C.T. Generalized Iterative Learning Control using Successive Projection: Algorithm, Convergence and Experimental Verification. *IEEE Trans. Control Syst. Technol.* **2020**, *28*, 2079–2091. [[CrossRef](#)]
36. Chen, Y.; Chu, B.; Freeman, C.T. Iterative Learning Control for Path-Following Tasks with Performance Optimization. *IEEE Trans. Control Syst. Technol.* **2022**, *30*, 234–246. [[CrossRef](#)]
37. Chen, Y.; Chu, B.; Freeman, C.T. Iterative Learning Control for Robotic Path Following with Trial-Varying Motion Profiles. *IEEE/ASME Trans. Mechatron.* **2022**, *Unpublished work*. [[CrossRef](#)]