International Journal of Information Management xxx (xxxx) xxxx

Contents lists available at ScienceDirect

International Journal of Information Management

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

On the training of a neural network for online path planning with offline path planning algorithms

Inkyung Sung*, Bongjun Choi, Peter Nielsen

Operations Research Group, Department of Materials and Production, Aalborg University, Fibigerstræde 16, 9220 Aalborg, Denmark

ARTICLE INFO

Keywords:
Path planning
Online path planning
Neural network
Offline path planning
Automated vehicle control
Data-driven control

ABSTRACT

One of the challenges in path planning for an automated vehicle is uncertainty in the operational environment of the vehicle, demanding a quick but sophisticated control of the vehicle online. To address this online path planning issue, neural networks, which can derive a heading for an operating vehicle in a given situation, have been actively studied, demonstrating their satisfactory performance. However, the study on the training path data, which specifies the desired output of a neural network and in turn influences the behavior of the neural network, has been neglected in the literature. Motivated by this fact, in this paper, we first generate different training path data sets applying two different offline path planning algorithms and evaluate the performance of a neural network as an online path planner depending on the training data under a simulation environment. We further investigate the properties of the training data that make a neural network more reliable for online path planning.

1. Introduction

Path planning is the process of finding a path between two points in an area while avoiding collisions with objects existing in the area. With the advance of robotics and control technologies and increased demands for automation in many industrial and public sectors, path planning has become a major topic in the domain of artificial intelligence, since it guides a vehicle to move safely from a point to another, determining the performance of a service performed by the vehicle (Khosiawan, Khalfay, & Nielsen, 2018). The application of path planning can be easily found in various contexts including even a virtual environment.

One of the critical characteristics of path planning for automated vehicles is the uncertainty in the operational environments of the vehicles. In particular, unknown or rarely detected objects in the operation environment of a vehicle demand continuous monitoring of the vehicle during operation. Therefore, online adaptation/re-computation of a path assigned to the vehicle is critical to guarantee the safety of the vehicle against newly detected and closely located objects. In the context of online or local path planning, given that a guide path is generated before deploying an automated vehicle, the position of the vehicle is continuously updated following the output of an online path planning algorithm until the vehicle reaches its goal position. This concept is described in Fig. 1.

Note that offline or global path planning, which often involves a shortest path problem, aims to find a complete path (i.e. a trajectory of points) in known environments. Furthermore, the algorithms for offline path planning are designed to explore as many paths as possible to minimize the travel distance of a path from a start point to a goal, making them impractical for online path planning. While there are efficient offline path planning algorithms, for example A* algorithm, they become intractable when the solution space is large, consuming huge memory resources during computation and implementation (Radmanesh, Kumar, Guentert, & Sarim, 2018).

To address online or local path planning, many heuristic path planning algorithms, which can quickly provide a segmented path or a heading of a vehicle in a given situation, have been proposed (Mac, Copot, Tran, & De Keyser, 2016). One of the actively applied solution approaches for online path planning is a *neural network*. A neural network is an adaptive system consisting of multiple inter-connected artificial neurons that models a complex relationship between the input and output of its target system. Due to its capability of modeling such complex relationships and linearity in the computation of an output given an input, many successful applications of the neural network to online path planning for automated vehicles have been reported. The application of neural networks is not limited to path planning. Neural networks are integral to various fields of research including computer vision and pattern recognition (Liu et al., 2017) and intelligence for

* Corresponding author.
E-mail addresses: inkyung_sung@mp.aau.dk (I. Sung), boc@m-tech.aau.dk (B. Choi), peter@mp.aau.dk (P. Nielsen).

https://doi.org/10.1016/j.ijinfomgt.2020.102142
Received 31 December 2019; Received in revised form 3 May 2020; Accepted 4 May 2020
0268-4012/ © 2020 Elsevier Ltd. All rights reserved.

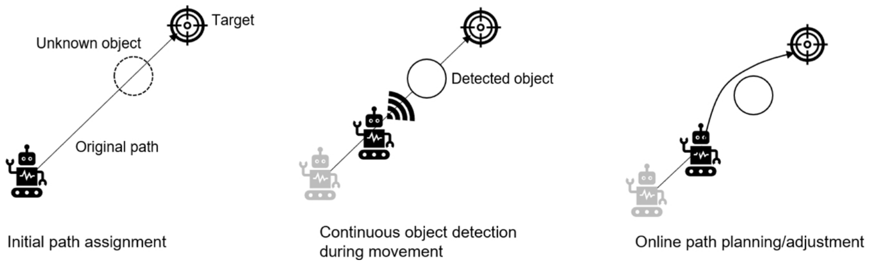
Target
Unknown object
Original path
Initial path assignment
Detected object
Continuous object detection during movement
Online path planning/adjustment

Fig. 1. Illustration of online path planning.

decision-making (Woźniak & Połap, 2020).

The concept of neural network-based online path planning is as follows. Based on the training data set representing desired paths of an automated vehicle, the connectivity between neurons of a neural network is updated such that the neural network models the relationship between the condition of a vehicle (input) and its desired path (output). Once the neural network is fully trained offline, it is embedded in an online path planning algorithm and derives a path online given a vehicle's operational condition (e.g. the current position of the vehicle).

It should be noted that, as explained above, the behavior and output of a neural network as an online path planner are obviously affected by its training path data. For example, when the training data does not contain some situations that a vehicle might encounter during its operation, the output quality of the corresponding neural network will be poor. However, training path data and its impact on the performance of neural networks have not received much attention.

Motivated by this fact, in this paper, we first implement two different offline path planning algorithms, the Bellman-Ford algorithm and a quadratic program, which derive the shortest and collision-free path between two points in a discretized and a continuous space, respectively. Many path planning problems sampled from a target environment are then solved by applying the offline path planning algorithms to prepare two different training data sets for a neural network. The neural network is trained based on the training data sets, separately, resulting in two different neural networks with the same structure, and the performance of the resulting neural networks is analyzed in a simulation environment, where unknown objects are suddenly detected by an automated vehicle. In the analysis, we further investigate the important properties of the training data that make the neural network more reliable as an online path planner.

The remainder of this paper is structured as follows. Section 2 reviews online path planning algorithms with a focus on the neural network approach. Section 3 provides preliminaries for the rest of the work in this study. The neural network-based online path planning system and its component algorithms including the two offline path planning algorithms are described in Section 4. The performance analysis of a neural network as an online path planner is conducted in Section 5, followed by a discussion of the implications, limitations and possible extensions of this study in Section 6. Finally, this paper is concluded in Section 7 with further remarks.

2. Related work

There is a large volume of literature on path planning, reflecting the increasing interest in the use of autonomous vehicles and the central role of path planning in operating such vehicles. Please refer to Ferguson, Likhachev, and Stentz (2005), Mac et al. (2016), Paull, Saeedi, Seto, and Li (2013), Radmanesh et al. (2018), Zhao, Zheng, and Liu (2018) for comprehensive reviews of path planning algorithms. In this research, we review online path planning algorithms based upon heuristic approaches that are able to provide a complete or segmented path quickly, reacting to sudden changes or newly available information about operational environments.

2.1. *Classical online path planning*

A common and simple approach to online path planning is a bug algorithm in which two types of commands are available for a vehicle: (1) move toward a goal and (2) follow the contour of an obstacle if detected until the vehicle can head to the goal again (McGuire, de Croon, & Tuyls, 2019).

The potential (or vector force) field algorithm has also been addressed as a solution for online path planning (Hwang & Ahuja, 1992). In the algorithm, a potential field over an operational environment is derived, where a goal position has the lowest potential, whereas the neighborhood of an obstacle has positive potentials with repulsive forces. A vehicle at a position then moves along the potentials/forces defined at the position. An illustration of a potential field with the forces represented as arrows is presented in Fig. 2.

The roll-out policy is another heuristic approach to online path planning. Given the current state of a path planning process, a policy, commonly developed using greedy algorithms, derives a set of available paths and the quality and feasibility of the paths are evaluated regarding the possible next states (Tian, Bar-Shalom, & Pattipati, 2008). Obviously, a path can be accurately evaluated by looking ahead many next states, improving the quality of the path selected. However, since the number of scenarios required for the evaluation grows exponentially as the number of look-ahead states increases, the algorithm tends to approximately evaluate a path with a small number of look-ahead states.

It should be noted that the above-mentioned algorithms form the basis of the heuristic approaches to path planning due to their simplicity. However, they often fail to provide a feasible path when many obstacles exist in an environment or when an environment is dynamic (Mac et al., 2016). As an alternative, neural networks have been actively studied, demonstrating their potential as an online path planning algorithm.

2.2. *Neural network-based online path planning*

The neural network applications for online path planning can be categorized into two classes. In the first, a neural network is applied to represent the operational environment of a vehicle (Chen et al., 2020; Li, Yang, & Seto, 2009; Yang & Meng, 2000; Zhong, Shirinzadeh, & Tian, 2008). Specifically, a neural network is constructed such that its neural activity landscape has a peak and a valley at the positions of a goal and an obstacle in the environment, respectively. An illustration of

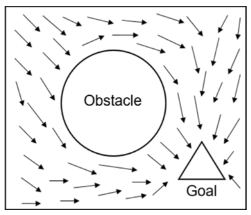

Fig. 2. A potential force map.

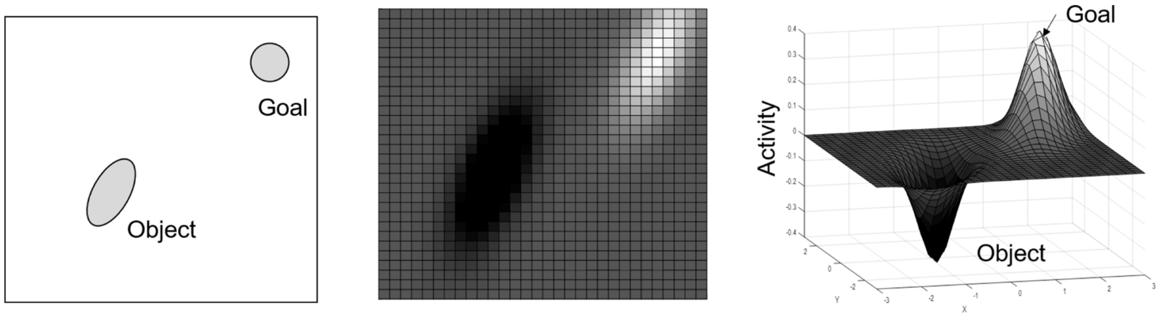

Fig. 3. Activity landscape of a neural network for path planning: an operational environment (left), projection of an activity landscape for the environment (middle), and an activity landscape for the environment (right).

the neural activity landscape for an environment is presented in Fig. 3.

With the activity landscape modeled by a neural network, a vehicle at a position is moved to one of the neighboring positions, where the corresponding neural activity value is maximized. With a shunting equation rooted in a model for a patch of membrane in a biological neural system, the activity value of a neuron in a neural network can also be updated to reflect changes in the operational environment.

It should be noted that such an application of a neural network is aligned with the concept of the potential field algorithm in the sense that a neural network represents an operational environment like a potential field. One of the limitations of this environment-representation approach is that additional control logic for a vehicle is necessary for the approach to be a complete path planning algorithm. Imagine a situation where neural activity values of the neighborhood positions of a vehicle are all equal. In this case, a rule or an action that resolves this degeneracy is essential for path planning.

The second class of neural network applications to online path planning is the neural network used as a controller (Capi, Kaneko, & Hua, 2015; Dezfoulian, Wu, & Ahmad, 2013; Singh & Parhi, 2011; Singh & Thongam, 2019). In this case, the situational conditions of a vehicle, for example, distance to an obstacle and bearing angle between the current position of the vehicle and a goal position, are used as inputs of a neural network. The neural network is then trained to convert the input to a heading of the vehicle that guarantees a collision-free movement of the vehicle.

Singh and Parhi (2011) designed a four-layer neural network for navigation of a mobile robot. The inputs of the neural network are the distances between the robot and the obstacles located at the left, the right, and the front of the robot, respectively, and the bearing angle between the robot and its goal position. Given the inputs, the neural network derives the heading of the robot. In a simulation environment, the proposed approach showed better performance than a fuzzy logic-based controller.

Capi et al. (2015) designed neural networks for a robot navigation with more detailed input. In the work of Capi et al. (2015), 17 neurons are used to represent the sensory data observed by a laser sensor with a 140° angle of view and a compass sensor, and the difference between the robot and its goal position. The proposed neural network demonstrated its performance as an online path planner in a real system.

Singh and Thongam (2019) also proposed a neural network for mobile robot navigation. Five inputs are used to represent robot's sight toward a goal divided into five segments. Depending on the situation in a segment, the input for a neuron corresponding to the segment has a value ranging from 0 to 1.5. A value of 0 means the segment is occupied by an object, whereas 1.5 means the segment is not blocked by any object. The outputs of the neural network are one of the five segments to move and the moving speed of a robot.

Note that the neural network-based online path planners are often designed for a specific robot and sensors. To generalize a neural network for other types of robots with different sensors, Dezfoulian et al. (2013) proposed an architecture, in which the input sensor data is first interpreted to decide which action to take (object avoidance or wall following) and the corresponding neural network to the action is activated to derive a heading of the robot. For the object-avoidance action, the neural network proposed by Singh and Parhi (2011) was implemented.

The study reported here belongs to the second class of the neural network applications and can be distinguished from the literature by the following contributions. First, we analyze the performance of a neural network depending on its training path data. For this, we prepare two different training data sets, derived under different assumptions and approaches. Based on the analysis, we further investigate the properties of the training data that make a neural network more reliable in finding a conflict-free path online.

3. Preliminaries: neural networks

A neural network is an inter-connected group of mathematical functions represented as neurons. A neural network is often composed of multiple neuron layers, where a layer is a collection of neurons with the same depth in a neural network. Fig. 4 illustrates a neural network with multiple layers. The input layer collects the initial data for the neural network and the output layer returns a classification result to which the input data is mapped. In the hidden layers the input data is transformed into the input for the output layer by mathematical functions.

Neurons denoted as nodes in a neural network are inter-connected with internal parameters called *weights*. Specifically, in a neural network, a neuron calculates the sum of products of its input and their corresponding weights, and adds a bias to the weighted sum. The output of the neuron is finally determined by an activation function of the neuron, which performs the nonlinear transformation of the net input into the output. The activation function introduces the non-linearity characteristic to a neural network, making the neural network capable of representing complex relationships as multiple-layered

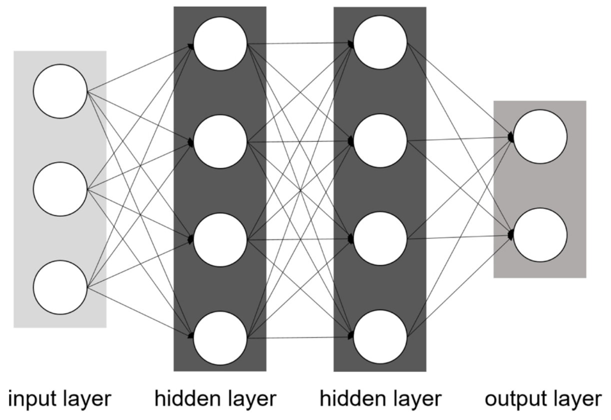

Fig. 4. A neural network structure.

combinations of such computations. The computation of a neuron can be written as follows:

$$y = f\left(\sum_{j=1}^{I} w_j x_j + b\right), \tag{1}$$

where y is the output of a neuron with I inputs and w is the weight between the input and output, b is a bias term, and f denotes a certain activation function. By having a fixed input $x_0 = 1$, which makes $w_0 = b$, the expression above can be simplified further as

$$y = f\left(\sum_{j=0}^{I} w_j x_j\right). \tag{2}$$

Based on this expression, a structure between layers in a neural network can be represented using matrix notation as follows:

$$\boldsymbol{y}^{(i)} = f(\boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)}), \tag{3}$$

where $\boldsymbol{y}^{(i)}$ is the output of ith neural network layer and $\boldsymbol{W}^{(i)}$ is the weight between the $i-1$th and the ith layers. Then for a multi-layer neural network, its input can be transformed into its output following Eq. (3), called forward propagation.

In the structure, a neural network is trained to model the relationship between input and its desired output by setting proper weights of the neural network. The goodness of the weight is often measured as a loss function, and one of the most used loss functions in the neural network context is the mean-squared error,

$$\begin{aligned} E(\text{error}) &= E((d(\boldsymbol{x}) - F(\boldsymbol{x}|\boldsymbol{W}))^2) \\ &= \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} (d(\boldsymbol{x}) - F(\boldsymbol{x}|\boldsymbol{W}))^2, \end{aligned} \tag{4}$$

where $\mathcal{D}$ is the entire training data set of $(\boldsymbol{x}, d(\boldsymbol{x}))$ pairs. This function aims to minimize the average squared error between the output of a neural network $F(\boldsymbol{x}|\boldsymbol{W})$ with weight $\boldsymbol{W}$ and the desired or actually observed value $d(\boldsymbol{x})$ given input $\boldsymbol{x}$.

Then we can use the gradient descent algorithm, which iteratively updates weight $\boldsymbol{W}$ based on the gradient of the loss function with respect to $\boldsymbol{W}$, to minimize the loss function. In an iteration of the algorithm, weight $\boldsymbol{W}$ is updated according to the following equations:

$$\boldsymbol{W}_{\text{new}} = \boldsymbol{W}_{\text{old}} - \alpha\cdot\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \nabla_W (d(\boldsymbol{x}) - F(\boldsymbol{x}|\boldsymbol{W}))^2, \tag{5}$$

where α is a parameter called learning rate. Note that calculating the gradient in Eq. (5) using data set $\mathcal{D}$ is impractical as the size of the set is huge in general. Instead, the gradient can be approximated by replacing the training data set $\mathcal{D}$ with a manageable batch data set sampled from $\mathcal{D}$, following the concept of the stochastic gradient descent algorithm.

Note that the output of a neural network with multiple layers, given weight $\boldsymbol{W}$ and input $\boldsymbol{x}$, is calculated by sequential transformations of input $\boldsymbol{x}$ through the multiple layers of the neural network following Eq. (3). Therefore, the gradient of the error term with respect to weight $\boldsymbol{W}$ can be calculated using the chain rule in differential calculus. Please refer to Goodfellow, Bengio, and Courville (2016) for more details about the gradient calculation.

4. Neural network-based online path planning

4.1. Problem description

The online path planning problem in this paper is formulated as follows. Given a two-dimensional operational environment $\mathcal{E}$, where multiple obstacles indexed by $o \in O$ exist, a vehicle should find a feasible path between start s and goal g. A path is defined as a sequence of points in $\mathcal{E}$ and is considered feasible when it does not cause a collision of the vehicle with any obstacles. An obstacle, represented as a circle, does not move but its position is not known a priori and will be detected as the vehicle approaches it.

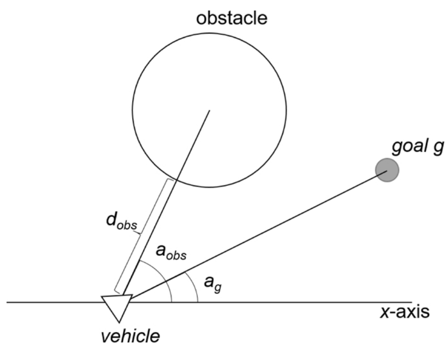

Fig. 5. The operational situation of a vehicle, which will be used as an input to a neural network.

The operational situation of a vehicle moving toward the goal is represented as a 3-tuple $\psi = \langle d_{obs}, a_{obs}, a_g\rangle$, where:

- d_{obs} is the distance to the closest obstacle from the current position of the vehicle;
- a_{obs} is the bearing angle between the vehicle and the closest obstacle to the vehicle with respect to the horizontal axis; and,
- a_g is the bearing angle between the vehicle and goal g with respect to the horizontal axis.

Fig. 5 illustrates the state of a vehicle, which will be used as input for a neural network.

4.2. Neural network-based online path planning system

To solve the online path planning problem, we designed a solution framework based on a neural network, where trajectories of feasible paths for various path planning problems are generated by applying offline path planning algorithms. A neural network trained based on the trajectories is then used to provide a control action θ for a vehicle (i.e. a heading of the vehicle) given the state of the vehicle ψ online. The structure of the framework is illustrated in Fig. 6.

As described, the neural network-based path planning system consists of two parts, the offline neural network training and the neural network-based online path planning. For the offline neural network training, we first generated training problems considering the operational environment of a vehicle. The generated problems were then solved by applying the offline path planning algorithms, which will be explained in the following sub-section. The solutions from the algorithms were set as the desired output of a neural network and the weights of the neural network were updated accordingly based on the process explained in Section 3.

Online path planning with the trained neural network is conducted as follows. Initially, a vehicle heads to its goal position and moves while updating its heading according to the output of the neural network. Given the current state of the vehicle at time step t, ψ_t, the neural network provides an action for the vehicle at that time, θ_t. The state of the vehicle is updated in the environment according to the action and the updated state at time $t+1$, ψ_{t+1}, is transmitted to the neural network for the next action. Such steps are repeated until the vehicle reaches its goal position.

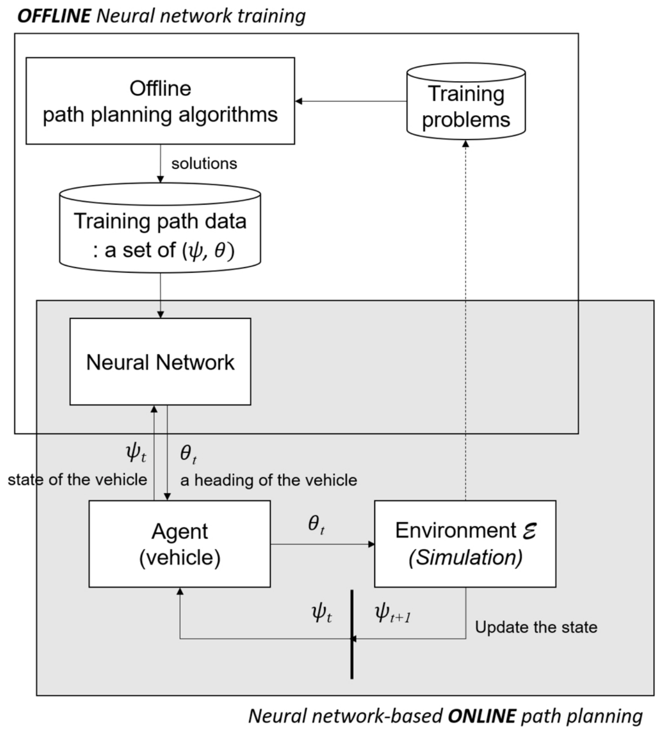
OFFLINE Neural network training
Offline path planning algorithms
Training problems
solutions
Training path data : a set of (ψ, θ)
Neural Network
ψ_t
θ_t
state of the vehicle
a heading of the vehicle
Agent (vehicle)
θ_t
Environment ℰ (Simulation)
ψ_t
ψ_t+1
Update the state
Neural network-based ONLINE path planning

Fig. 6. A solution framework for online path planning using a neural network.

4.3. offline neural network training

4.3.1. Training data set generation: offline path planning algorithms

To generate the desired paths, which will be used to train a neural network, we implement two offline path planning algorithms. Recall that offline path planning is designed to provide a complete path of a vehicle with a focus on minimizing the distance of the path. While the algorithms for offline path planning take a relatively long time to generate the final paths, making them impractical for online path planning, the shortness of the paths is guaranteed.

The offline path planning algorithms in literature can be classified into two types, depending on their solution spaces. A solution space in path planning specifies possible positions for a vehicle. The solution space is often discretized as a graph, which has finite positions for vehicle placement. In contrast, a vehicle can be placed anywhere in a continuous solution space.

Considering the fundamental differences between the paths derived under the two different solution spaces (discrete or continuous) and the expected impacts of the differences on the performance of the proposed approach, we implemented two different offline path planning algorithms: the Bellman–Ford algorithm working with a discrete solution space and a quadratic program working with a continuous solution space.

Let us first explain the Bellman-Ford algorithm, which is one of the most common approaches to find the shortest path in a graph. When using the Bellman–Ford algorithm, an operational area is first decomposed into a set of regular-sized grids and a grid-based graph connecting the corners of the grid is constructed. In the graph, a vehicle can move at most in eight directions. Fig. 7 illustrates the structure of the graph.

With a square grid-based graph $G = (V, A)$ where V and A are the set of nodes and arcs in the graph, respectively, the Bellman–Ford algorithm with the First-in-First-Out (FIFO) node selection priority rule that continuously updates the label (cost) of node $i \in V$, L_i, is implemented as described in Algorithm 1.

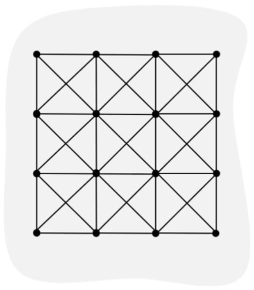

Fig. 7. A graph structure for the Bellman–Ford algorithm, where the shaded area represents the original space for path planning, the black circles represent possible positions for a vehicle and the lines specify the movement between positions.

Algorithm 1. FIFO label setting algorithm

```
$L_i \longleftarrow \infty$ , $\forall i \in N \setminus \{o\}$, $L_o \longleftarrow 0$
$LIST \longleftarrow \{o\}$
while $LIST \neq \emptyset$ do
  Choose the first node in $L$ and remove the node from $L$
  for all arc $(i, j) \in A$ do
    if $(i, j)$ does not overlap with any obstacles in $\mathcal{E}$ do
      $L_j^{\text{temp}} \longleftarrow L_i + c_{ij}$
      if $L_j^{\text{temp}} < L_j$ then
        $L_j \longleftarrow L_j^{\text{temp}}$
        Append $j$ to the end of $LIST$
      end if
    end if
  end for
end while
```

As described, once the label of a node is updated, its neighborhood nodes are added to the scanning node list *LIST*, resulting in the computational complexity $O(|V||A|)$. Compared with Dijkstra's algorithm, whose computational complexity is $O(|V|^2)$ in its original form, the Bellman–Ford algorithm tends to work well in a sparse graph like the one used for path planning and transportation.

Next, we implement a quadratic program (QP) for path planning modified from the model proposed in Turnbull, Lawry, Lowenberg, and Richards (2016). The QP is implemented to minimize the distance between the position of a vehicle $\boldsymbol{P}_t$ at time step t and the goal position $\boldsymbol{P}_g$ over the planning time horizon T, subject to avoiding collisions with obstacles. In the QP, the position of a vehicle is updated by the decision variables – the velocity of the vehicle at time step t, $\boldsymbol{V}_t$, and the acceleration of the vehicle at time step t, $\boldsymbol{A}_t$.

Note that the constraint, which ensures that the distance between the position of the vehicle at time step t, $\boldsymbol{P}_t$, and the center point of obstacle o $\boldsymbol{P}_o$ is greater than or equal to the radius of the obstacle R_o, written as

$$||\boldsymbol{P}_t - \boldsymbol{P}_o||_2 \geq R_o \quad \forall\ o \in O,\ \ 0 < t \leq T, \tag{6}$$

leads to the non-convexity of the original path planning problem. To handle this, following the work in Turnbull et al. (2016), we approximate the non-convexity constraint as a linear form while representing an obstacle as a polygon with $n_c = 6$ sides. The QP with the approximation is written as follows:

$$\min \quad \sum_{t=1}^{T} ||\boldsymbol{P}_t - \boldsymbol{P}_g||_2 \tag{7}$$

$$\text{subject to} \quad \boldsymbol{P}_0 = \boldsymbol{P}_s \tag{8}$$

$$\boldsymbol{V}_0 = \boldsymbol{v}_0 \tag{9}$$

$$\boldsymbol{A}_0 = \boldsymbol{0} \tag{10}$$

$$\boldsymbol{V}_{t+1} = \boldsymbol{V}_t + \boldsymbol{A}_t \Delta t \quad \forall\ 0 \le t < T \tag{11}$$

$$\boldsymbol{P}_{t+1} = \boldsymbol{P}_t + \boldsymbol{V}_t + \frac{1}{2}\boldsymbol{A}_t \Delta t^2 \quad \forall\ 0 \le t < T \tag{12}$$

$$\|\boldsymbol{V}_t\|_2 \le v_{\max} \quad \forall\ 1 \le t \le T \tag{13}$$

$$\|\boldsymbol{A}_t\|_2 \le a_{\max} \quad \forall\ 1 \le t \le T \tag{14}$$

$$\boldsymbol{d}_c(\boldsymbol{P}_t - \boldsymbol{P}_o) \ge R_o - M\cdot b_{oct} \quad \forall\ o \in O,\ c \in N_c,\ 1 \le t \le T \tag{15}$$

$$\sum_{c \in N_c} b_{oct} \le n_c - 1 \quad \forall\ o \in O,\ 1 \le t \le T \tag{16}$$

$$\boldsymbol{P}_t, \boldsymbol{V}_t, \boldsymbol{A}_t \in \mathbb{R}^2 \quad \forall\ 1 \le t \le T \tag{17}$$

$$b_{oct} \in \{0, 1\} \quad \forall\ o \in O,\ c \in N_c,\ 1 \le t \le T \tag{18}$$

The objective (7) is to minimize the sum of the distance between a vehicle and the goal position, over the planning horizon T. Note that the positions of the vehicle and the goal are represented as two-dimensional coordinates. Constraints (8)–(10) set the initial position, velocity and acceleration of the vehicle, respectively. Given the time interval between time steps, Δt, the velocity and position of the vehicle are updated by constraints (11) and (12), respectively. Constraints (13) and (14) limit the maximum velocity and acceleration of the vehicle. The linear approximation to Eq. (6) is done by constraints (15) and (16) as proposed in Turnbull et al. (2016). The matrix $\boldsymbol{d}_c$ is set as $\left[\cos\frac{2\pi c}{n_c}\ \ \sin\frac{2\pi c}{n_c}\right]$. Constraint (16) selects which side of an obstacle, $c \in N_c = \{1, \cdots, n_c\}$, is used to avoid the obstacle. M is a large positive scalar. Constraints (17) and (18) are real number and binary constraints for the decision variables.

It should be clearly noted that the two offline path planning algorithms are different in terms of their solution spaces. The Bellman-Ford algorithm gives a finite set of positions for a vehicle through the discretization of a continuous operational area (see Fig. 7), whereas the QP (7)–(18) deals with path planning in a continuous space, providing more freedom in path generation.

4.3.2. Neural network implementation

We designed a six-layered neural network for online path planning. The input layer had three neurons based on the setup for the state of a vehicle, $\psi = \langle d_{obs}, a_{obs}, a_g \rangle$. The neural network had four hidden layers with 50 nodes on each layer. The output layer had a single node for $\theta \in \{-\pi, \pi\}$. We used the rectified linear unit (ReLU) defined as $y = \max(0, x)$ for the activation function, which is one of the most commonly used activation functions for neural networks. The structure of the neural network in this study is illustrated in Fig. 8.

Note that the given state ψ, desired output $\theta^*(\psi)$ was set by the offline path planning algorithm. The neural network was then trained to minimize the error between $\theta^*(\psi)$ and its output $\theta(\psi)$ by updating its weight as explained in Section 3.

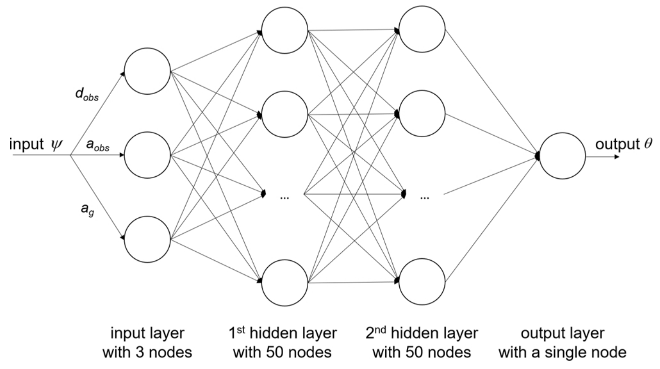
d_{obs}
input ψ
a_{obs}
a_g
output θ
...
...
input layer with 3 nodes
1st hidden layer with 50 nodes
2nd hidden layer with 50 nodes
output layer with a single node

Fig. 8. The structure of the neural network implemented.

Table 1
The feasibility of the neural networks for the test problems.

	Failure to		Success	Total
	Find a goal	Avoid collisions		
$\mathcal{N}_{BF}$	0	0	30	30
$\mathcal{N}_{QP}$	4	1	25	30

5. Impacts of the training data on the neural networks

In this section, we present our investigation of the performance of the neural network as an online path planning algorithm depending on the training data sets generated by the two offline path planning algorithms. We first generated a path planning problem set to be solved by the two offline path planning algorithms. We generated 10,000 path planning problem instances on a 50 × 50-unit distance-sized map, randomly allocating 10 obstacles and the start and the goal of a vehicle per problem instance. No obstacle overlapped with any other obstacle and its radius had a value between two- and five-unit distances.

The problems were then solved by the offline path planning algorithms, separately, resulting in two training data sets, $\mathcal{D}_{BF}$ and $\mathcal{D}_{QP}$. Note that a complete path is a sequence of positions of a vehicle. Thus, we extracted the pairs of (ψ, θ) from the path. For 10,000 path planning problems, we gathered 229,385 and 273,892 pairs of (ψ, θ) for $\mathcal{D}_{BF}$ and $\mathcal{D}_{QP}$, respectively.

Based on the two training data sets, we trained two neural networks, $\mathcal{N}_{BF}$ and $\mathcal{N}_{QP}$ and compared them in a simulation environment. For comparison, we generated 30 test problem instances with the same setting for the training problem instance creation. In the simulation, a vehicle's velocity is constant and its position, initially set as its starting position, is updated following the heading derived by the trained neural networks, the $\mathcal{N}_{BF}$ and the $\mathcal{N}_{QP}$, at every time step. When the vehicle is close enough to a goal position (i.e. within a half-unit distance), the vehicle is considered arrived and finishes its journey.

Let us first present the feasibility of the paths found by the neural networks for the 30 problem instances. Table 1 shows the feasibility of the neural networks with the two different training sets.

It can be first observed that the $\mathcal{N}_{BF}$ shows its feasibility as an online path planner deriving feasible paths, i.e. a path from a start to a goal without any collisions with obstacles, for the test problem instances. On the other hand, the $\mathcal{N}_{QP}$ does not show its feasibility as it often fails to reach a goal position or to avoid an obstacle.

To understand the reason for the performance difference, we took a close look at the paths generated by the two offline path planning algorithms. In the investigation, we found that the behavior in avoiding an obstacle differed depending on the algorithm; this is clearly presented in Fig. 9.

As presented in the figure, given the same setting for path planning, the QP (solid line) finds the path that smoothly avoids an obstacle, giving a shorter path length than the path given by the Bellman–Ford

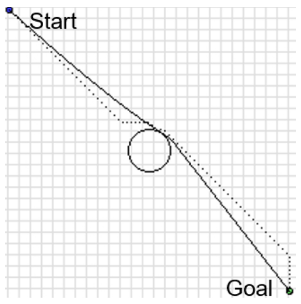

Fig. 9. Paths derived from the Bellman-Ford algorithm (dotted) and the QP (solid).

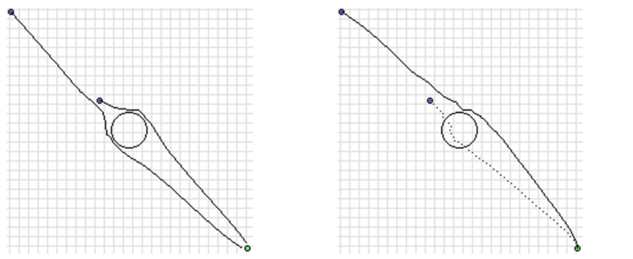

Fig. 10. Paths derived from the $\mathcal{N}_{BF}$ (left) and the $\mathcal{N}_{QP}$ (right) with different starting points.

algorithm (dotted line). This is because the QP derives a path over a continuous space, whereas the Bellman–Ford algorithm derives a path in a discretized space with at most eight directions to move in. Interestingly, this difference, which might make the QP more suitable for offline path planning than the Bellman–Ford algorithm, makes the $\mathcal{N}_{QP}$ unreliable for online path planning.

Specifically, in the training data generated by the QP, there is a lack of training data representing a situation where an obstacle is relatively close to a vehicle. As presented in Fig. 9, the QP tends to adjust the heading of a vehicle in an early stage of the journey, minimizing the total distance of the path.

Consequently, when a nearby obstacle is detected, which did not occur in the training data, the $\mathcal{N}_{QP}$ trained by the data tended to fail to provide a desired action. Fig. 10 explicitly shows this situation. In the figure, while the $\mathcal{N}_{BF}$ found feasible paths regardless of the distance to an obstacle, the $\mathcal{N}_{QP}$ failed to avoid an obstacle when the obstacle was located close to a vehicle (the dotted line in the figure).

We further tested the performance of the neural networks in large-scale path planning problems with densely located obstacles. We generated 30 problem instances on a 100 × 100-unit distance-sized map, randomly placing 80 obstacles per problem instance (two times as dense as the first test problem setting). Other conditions for the test problem generation were the same as in the first test problem set. The feasibility analysis is summarized in Table 2.

First, the table shows that the $\mathcal{N}_{BF}$ failed 5 times in finding collision-free paths for the large-scale problem instances. The failures occurred when multiple obstacles were located close to each other. Since the designed neural network only considered the closest obstacle from the current position of a vehicle, the action derived by the neural network could cause a collision with one of the other obstacles. This issue will be discussed in detail in the following section. It is also observed that the feasibility of the $\mathcal{N}_{BF}$ became worse in the large-scale problem instances, as was expected.

However, when we consider the cases where the $\mathcal{N}_{BF}$ found feasible paths to the goals, the $\mathcal{N}_{BF}$ showed competitive performance in terms of path length. We compared the length of the paths generated by the $\mathcal{N}_{BF}$ with the paths generated by the Bellman-Ford algorithm for the feasibility-guaranteed cases. Table 3 summarizes the comparison.

In the table, we present the *optimality gap*, the relative path length increment introduced by the $\mathcal{N}_{BF}$ with respect to the path from the Bellman–Ford algorithm. Interestingly, the paths generated by the $\mathcal{N}_{BF}$ tended to be shorter than those generated by the Bellman-Ford algorithm (a negative optimality gap on average). As the obstacles are more densely located in the environment, possibly reducing the available

Table 2
The feasibility of the neural networks for the large-scale test problems.

	Failure to		Success	Total
	Find a goal	Avoid collisions		
$\mathcal{N}_{BF}$	0	5	25	30
$\mathcal{N}_{QP}$	14	7	9	30

Table 3
Optimality of the paths generated by the $\mathcal{N}_{BF}$ in terms of length.

	Optimality gap (%)[a]	
	50 × 50 with 10 obstacles	100 × 100 with 80 obstacles
Avg.	−0.39	−1.11
Std.	0.03	0.04
Max	7.69	4.79
95%	4.11	4.26

[a] Optimality gap: 100 (%) × (Path length by the proposed approach − Path length by the Bellman–Ford algorithm)/Path length by the Bellman–Ford algorithm.

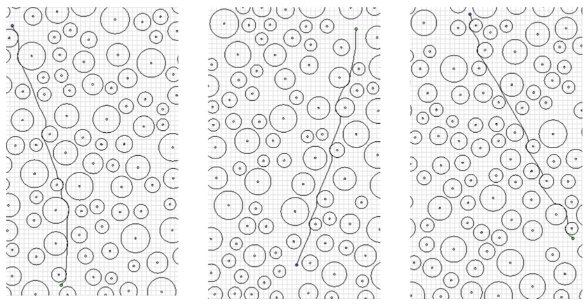

Fig. 11. Feasible paths generated by the $\mathcal{N}_{BF}$.

positions of a vehicle in a graph for the Bellman–Ford algorithm, reduction in path length even becomes slightly increased. This is because the $\mathcal{N}_{BF}$ finds a path in a continuous space, giving more chances to avoid an obstacle with minimum path increments. Example feasible paths generated by the $\mathcal{N}_{BF}$ are presented in Fig. 11.

In brief, based on the analysis presented above, the findings can be noted as follows:

- having the training data for a situation where an obstacle is close to a vehicle's position is critical to train a neural network in online path planning, especially in uncertain and dynamic environments; and
- a neural network-based online path planning shows its potential in finding the shortest and collision-free path when provided with a proper training data set.

6. Discussion

The main keyword that describes the synthesis of our study with existing research in relevant fields is automation. Recall that the proposed approach automatically generates the training data set by applying offline path planning algorithms, and the neural network trained by the data guides a vehicle to reach a goal online. As described, the human intervention in the proposed approach is minimized. This type of approach has an impact on large-scale automated services such as smart cities and factories (Manfreda, Ljubi, & Groznik, 2019), where large numbers of entities including automated vehicles are operating interactively and control of all the entities with human intervention is impossible.

In relation to automation, the contributions of the proposed approach to online path planning and relevant applications can be noted as follows. First, gathering a large enough and effective enough training data set for the neural network-based online path planning can be done automatically, which is not straightforward to achieve in many applications of artificial intelligence (Dwivedi et al., 2019). This can increase the automation level of automated vehicle deployments. Next, we showed that a neural network can capture patterns existing on the shortest and collision-free paths generated by offline path planning algorithms, which is difficult to be formally stated by human operators. Last, due to the computational ease and low hardware requirements for

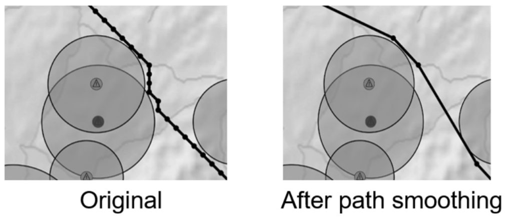

Fig. 12. The concept of path smoothing.

running a neural network-based control system, implementation of such a system to a vehicle with limited onboard computing capacity is feasible. This will greatly increase the speed with which automated units can move safely through complicated environments as the actual use of the instantiated neural network has a negligible computation time.

6.1. Practical implications

First and foremost, the main implication for practical applications is as stated above that the approach is straight forward to apply to any reasonable automated environment where independent automated units are traveling. The proposed solution framework for online path planning can be adapted by incorporating application-specific constraints into the offline path planning algorithms, while retaining the proposed structure of the approach.

A further clear implication is seen through the experimental results. Here it was conclusively demonstrated that the training data has a significant impact on the performance of the neural network for online path planning, leading to the following practical remarks on training data construction:

- offline path planning algorithms working with known environments can be used to set the desired output of a neural network as an online path planner, which lessens the burden of generating desired paths for the neural network;
- the patterns in the desired paths derived by the algorithms should be reviewed with particular attention to their applicability to an uncertain and dynamic environment; and
- for this purpose, potential exceptional situations which a vehicle would face during operation can be included in the training data set for a neural network with desired actions corresponding to the situations.

Combined with the finding in this study that avoiding an obstacle with a proper training data set can be relatively easily achieved by training a simple structure neural network, the remarks also suggest that given a limited time budget for implementing a neural network-based online path planner, preparing a training data including the offline path planning algorithm design deserves a priority effort.

6.2. Theoretical contributions

In this research, we demonstrated that acquisition of a desired path set is a rather open-ended process of training the neural network as an online path planner. There are specific properties desirable to train such a neural network, e.g. containing many situations where a vehicle avoids an obstacle suddenly. Thus theoretical and empirical studies on the properties could further guide the practice of neural network applications in online path planning.

Furthermore, we see the potential of a neural network for path smoothing in graph-based offline path planning algorithms. As observed from our experimental results, a path derived from a graph (a discrete space) tends to be rough due to the discretization of a continuous operational area, incurring unnecessary additional travel distances. To minimize the quality loss by the discretization, a neural network trained based on the behavior of a target offline path planning algorithm can be applied to the path derived by the algorithm to make the path smoother. This is clearly demonstrated to reduce its length and increasing operability of the path. The concept of path smoothing is illustrated in Fig. 12, adopted from the work in Danancier, Ruvio, Sung, and Nielsen (2019).

6.3. Limitations and future research directions

As presented in the previous section, we observed the potentials of a neural network in finding a feasible path with the minimum length in online path planning. At the same time, the limitations of the current neural network design were identified especially when obstacles were closely located. This could be because the neural network in this study considered the closest obstacle only to derive the action for a vehicle. This issue can also be found in path planning with the control logic that considers a single obstacle (Danancier et al., 2019; Turnbull et al., 2016).

This weakness becomes more obvious when multiple obstacles overlapped, as shown in Fig. 13. As presented, control to avoid an obstacle could cause a conflict with other nearby obstacles.

To overcome the weakness of the proposed approach, the state ψ can be extended to represent more obstacles in an operational environment (e.g. the obstacles within a certain distance of a vehicle), so that the neural network can derive an action, which considers these obstacles. A convolutional neural network, which is designed to handle spatial-oriented image data as its input, can be an alternative to this problem. The curse of dimensionality and the size of the trained data introduced by the extension will be the main issue in the use of a neural network for online path planning.

The approach proposed in this paper can also be extended to address a three-dimensional environment, introducing additional dimensions to the input and output of a neural network. The main challenge here is that this dramatically increases the amount of training data needed to provide similar performance as achieved in the two-dimensional environment in this paper. Moreover, it is computationally more demanding to generate optimal paths in a three-dimensional space using off-line path planning algorithms.

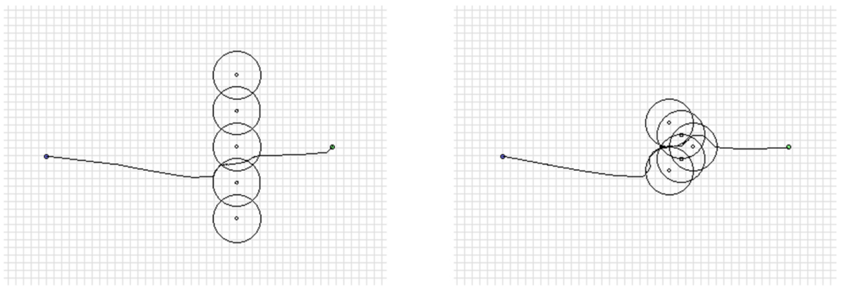

Fig. 13. Path planning using the proposed approach in cases with overlapping obstacles.

7. Conclusion

Following the promising demonstrated/potential benefits of deploying autonomous vehicles in various applications and the key role of online path planning in the applications, in this paper, we addressed an online path planning algorithm applying a neural network. With offline and exact algorithms for path planning, which provide two different training data sets for the neural network, we trained two neural networks and tested their performances with respect to providing collision-free paths.

Let us conclude this paper by discussing the expected barriers regarding acceptance by human decision makers. First, the interpretability of the output derived by a neural network can be low. This is critical in many contexts such as military operations or safety critical situations, where it is important for users or stakeholders to understand the rationale of the path planning decision. Neural networks are generally constructed with many parameters to represent complex relationships with high accuracy, making it more difficult to understand how the neural networks work. To solve this issue, techniques that reduce the ambiguity of neurons (the possibility of activating a neuron via various unrelated factors) can be implemented in a neural network for path planning.

Next, considering the practical difficulty of acquiring perfect situational awareness in an operational environment during online path planning, additional control logic that guides a vehicle when the neural network fails to give a feasible path can be integrated into a neural network-based path planning system. In this sense, the proposed approach would be used as a component of a decision support tool for human operators or a vehicle control system rather than a fully automated vehicle control system, which is aligned with the role of artificial intelligence in decision making proposed in Duan, Edwards, and Dwivedi (2019).

Authors' contribution

Inkyung Sung: conceptualization, methodology, software, validation, writing – original draft, review and editing. Bongjun Choi: methodology, software, validation, investigation. Peter Nielsen: conceptualization, writing – review and editing.

References

Capi, G., Kaneko, S., & Hua, B. (2015). Neural network based guide robot navigation: An evolutionary approach. *Procedia Computer Science, 76*, 74–79.

Chen, Y., Liang, J., Wang, Y., Pan, Q., Tan, J., & Mao, J. (2020). Autonomous mobile robot path planning in unknown dynamic environments using neural dynamics. *Soft Computing,* 1–17.

Danancier, K., Ruvio, D., Sung, I., & Nielsen, P. (2019). Comparison of path planning algorithms for an unmanned aerial vehicle deployment under threats. *IFAC-PapersOnLine, 52*, 1978–1983.

Dezfoulian, S. H., Wu, D., & Ahmad, I. S. (2013). A generalized neural network approach to mobile robot navigation and obstacle avoidance. *Intelligent autonomous systems 12,* 25–42.

Duan, Y., Edwards, J. S., & Dwivedi, Y. K. (2019). Artificial intelligence for decision making in the era of big data-evolution, challenges and research agenda. *International Journal of Information Management, 48*, 63–71.

Dwivedi, Y. K., Hughes, L., Ismagilova, E., Aarts, G., Coombs, C., Crick, T., et al. (2019). Artificial intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management*.

Ferguson, D., Likhachev, M., & Stentz, A. (2005). A guide to heuristic-based path planning. *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS),* 9–18.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Presshttp://www.deeplearningbook.org.

Hwang, Y. K., Ahuja, N., et al. (1992). A potential field approach to path planning. *IEEE Transactions on Robotics and Automation, 8*, 23–32.

Khosiawan, Y., Khalfay, A., & Nielsen, I. (2018). Scheduling unmanned aerial vehicle and automated guided vehicle operations in an indoor manufacturing environment using differential evolution-fused particle swarm optimization. *International Journal of Advanced Robotic Systems,* 15.

Li, H., Yang, S. X., & Seto, M. L. (2009). Neural-network-based path planning for a multirobot system with moving obstacles. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 39*, 410–419.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing, 234*, 11–26.

Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems, 86*, 13–28.

Manfreda, A., Ljubi, K., & Groznik, A. (2019). Autonomous vehicles in the smart city era: An empirical study of adoption factors important for millennials. *International Journal of Information Management*.

McGuire, K., de Croon, G., & Tuyls, K. (2019). A comparative study of bug algorithms for robot navigation. *Robotics and Autonomous Systems, 121*, 103261.

Paull, L., Saeedi, S., Seto, M., & Li, H. (2013). AUV navigation and localization: A review. *IEEE Journal of Oceanic Engineering, 39*, 131–149.

Radmanesh, M., Kumar, M., Guentert, P. H., & Sarim, M. (2018). Overview of path-planning and obstacle avoidance algorithms for UAVs: A comparative study. *Unmanned Systems, 6*, 95–118.

Singh, M. K., & Parhi, D. R. (2011). Path optimisation of a mobile robot using an artificial neural network controller. *International Journal of Systems Science, 42*, 107–120.

Singh, N. H., & Thongam, K. (2019). Neural network-based approaches for mobile robot navigation in static and moving obstacles environments. *Intelligent Service Robotics, 12*, 55–67.

Tian, X., Bar-Shalom, Y., & Pattipati, K. R. (2008). Multi-step look-ahead policy for autonomous cooperative surveillance by UAVs in hostile environments. *2008 47th IEEE Conference on Decision and Control,* 2438–2443.

Turnbull, O., Lawry, J., Lowenberg, M., & Richards, A. (2016). A cloned linguistic decision tree controller for real-time path planning in hostile environments. *Fuzzy Sets and Systems, 293*, 1–29.

Woźniak, M., & Połap, D. (2020). Intelligent home systems for ubiquitous user support by using neural networks and rule-based approach. *IEEE Transactions on Industrial Informatics, 16*, 2651–2658.

Yang, S. X., & Meng, M. (2000). An efficient neural network method for real-time motion planning with safety consideration. *Robotics and Autonomous Systems, 32*, 115–128.

Zhao, Y., Zheng, Z., & Liu, Y. (2018). Survey on computational-intelligence-based UAV path planning. *Knowledge-Based Systems, 158*, 54–64.

Zhong, Y., Shirinzadeh, B., & Tian, Y. (2008). A new neural network for robot path planning. *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics,* 1361–1366.