Neurocomputing 357 (2019) 151–162

Contents lists available at ScienceDirect

Neurocomputing

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

TDPP-Net: Achieving three-dimensional path planning via a deep neural network architecture

Keyu Wu, Mahdi Abolfazli Esfahani, Shenghai Yuan, Han Wang*

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 4 May 2018
Revised 12 February 2019
Accepted 5 May 2019
Available online 9 May 2019

Communicated by Prof. Wei Feng

Keywords:
Deep neural network
Path planning
Deep learning
Real-time autonomous navigation
Autonomous unmanned vehicles

ABSTRACT

Path planning plays a significant role in autonomous navigation for robots in complex environments and hence has been extensively studied for decades. However, the computational time of most existing methods are dependent on the scale and complexity of environment, which leads to the compromise between time efficiency and path quality. To tackle this challenge, deep neural network based (DNN-based) planning methods have been actively explored. However, despite the success of DNN-based 2D planner, 3D path planning, which is a significant primitive for quite a few autonomous robots, is rarely handled by DNNs. In this paper, we propose a novel end-to-end neural network architecture named Three-Dimensional Path Planning Network (TDPP-Net) to realize DNN-based 3D path planning. Embedding the action decomposition and composition concept, our network predicts 3D actions merely through 2D convolutional neural networks (CNNs). Besides, the computational time of TDPP-Net is almost independent of environmental scale and complexity for each action prediction. The experimental results demonstrate that our approach exhibits remarkable performance for planning real-time paths in unseen 3D environments.

1. Introduction

Autonomous robots can be deployed in a wide range of applications, and hence various related researches have been actively conducted [1–4]. As an important prerequisite for autonomous robots, path planning has been one of the most extensively studied topics over the decades. Although representative path planning algorithms as well as their variants have been widely adopted in various applications [5,6], the computational time of most conventional approaches is dependent on the complexity and scale of environment. Moreover, these algorithms are difficult to be accelerated using GPUs because of their low potential for parallelization [7]. However, deep neural network architectures, such as CNNs, are promising to overcome the bottlenecks of conventional planning algorithms and learn to plan efficiently under various conditions.

In recent years, CNNs have achieved extraordinary success due to their impressive performance in various applications such as object recognition [8] and detection [9], image classification [10], semantic segmentation [11], and action recognition [12]. Moreover, CNNs nowadays have also been utilized to address a variety of planning problems [13–16], where a policy is characterized using a neural network. While performing these tasks, CNNs are typically adopted to extract features either during reinforcement learning [17,18] or imitation learning [19]. However, in spite of remarkable performance in feature extraction, reactive CNNs fail to generalize well to unseen maps as there lacks explicit planning procedure to recognize the nature of the tasks [20].

Based on the relation between planning algorithms and recurrent neural networks [21], the value iteration network (VIN) was proposed to approximate the Bellman update through fully differential CNNs [20]. With a planning computation module embedded in the feed-forward classification network, VIN demonstrates superior capability in learning 2D environments and thus can derive policies with improved generalization to new circumstances that have not been encountered previously. Besides VIN, similar networks or variants of VIN were also introduced. Instead of being model-free, the predictron proposed in [22] adopted an abstract model to scale more effectively in domains without the 2D encoding nature of state space. In addition, a generalized value iteration network was introduced in [23] to handle both regular and irregular graphs by adopting graph-based kernel functions. Also inspired by VIN, a path integral network was proposed in [24] to achieve optimal continuous control. In [25], the transition function was represented by a recurrent neural network and the QMDP RCNN proposed in [26] combined a Value Iteration RCNN and a Belief Propagation RCNN to achieve a more intuitive understanding of the

* Corresponding author.
E-mail addresses: wukeyu@ntu.edu.sg (K. Wu), MAHDI001@e.ntu.edu.sg (M. Abolfazli Esfahani), SYUAN003@e.ntu.edu.sg (S. Yuan), hw@ntu.edu.sg (H. Wang).

https://doi.org/10.1016/j.neucom.2019.05.001
0925-2312/

action choices. Besides, the visual navigation system developed in [27] utilized a hierarchical version of the value iteration network as its planning module while multiple VINs were interconnected in [28] for cooperative planning tasks.

As introduced above, DNN-based path planning is not newfangled in 2D space because of its high potential in performance improvement. However, for many autonomous robots, such as unmanned aerial vehicles (UAVs) [29] and autonomous underwater vehicles (AUVs) [30], 2D path planning is inadequate, which motivates the development of 3D planners. Nevertheless, DNN-based 3D path planners have rarely been developed to address the bottlenecks of the conventional algorithms. To resolve this problem, we propose an end-to-end neural network architecture named Three-Dimensional Path Planning Network (TDPP-Net) in this paper to plan 3D paths in real time through utilizing 2D CNNs.

Instead of using 3D CNNs to represent the value iteration algorithm in 3D space, our TDPP-Net revolves around the concept of action decomposition and composition within the network architecture. Specifically, TDPP-Net is trained to represent a policy which maps the observation of a system's state into a 3D action in the following way. First, multiple 2D maps encoding the surrounding obstacles and the goal information are generated based on the 3D observation. The value iteration modules are then applied to calculate the values for all the investigated 2D planes. Subsequently, the action is synthesized by taking all these 2D Q-values into account through a fully connected layer.

In our implementation, the policy is learned by mimicking the expert demonstrations through imitation learning and the effectiveness of TDPP-Net is demonstrated by generalizing the trained policy to new situations. Generally, TDPP-Net outperforms the 3D VIN in terms of training loss, training accuracy, success rate as well as path difference. Moreover, different from conventional 3D path planning methods, the computational time of TDPP-Net for each action prediction is almost constant under different circumstances. Also, each action prediction using TDPP-Net is independent due to the end-to-end nature. Hence, TDPP-Net is promising to deal with large-scale, complex, and dynamic environments more efficiently.

The main contributions of this paper can be summarized as follows:

- A novel end-to-end neural network architecture named TDPP-Net is proposed for planning real-time paths with high quality in 3D space.
- TDPP-Net utilizes 2D CNNs to tackle 3D path planning problems by embedding the concept of action decomposition and composition in the network architecture.
- TDPP-Net is capable of determining 3D actions with significantly improved performance in unseen environment by focusing on partial obstacle information.
- TDPP-Net is promising to plan paths in large-scale environments more efficiently benefiting from the scale-independent computational time for each action prediction.

Besides, a post-processing strategy is introduced to further improve the success rate of the proposed path planner though local adjustment during implementation. The remainder of the paper is organized as follows. The preliminaries are described in Section 2 while the TDPP-Net is introduced in Section 3. In Section 4, the experimental results are comprehensively presented and discussed. Finally, the paper is summarized and concluded in Section 5.

2. Preliminaries

Markov decision processes (MDPs) are state transition systems that are widely used for modeling sequential decision making scenarios and typically, planning problems can be formulated as instances of MDPs [31]. In general, an MDP consists of a set of states $s \in S$, a set of actions $a \in A$, a reward function $R(s, a, s')$, and a transition model $P(s'|s, a)$ which describes the probability of transitioning to a new state s' via taking an action a in current state s. In an MDP, a policy $\pi(a|s)$ specifies a mapping from states to actions and the optimal policy π^* is the one resulting in the maximum accumulated rewards in the long run [32].

The value iteration algorithm devised by Bellman [33] can be employed to determine the optimal policy. First, the value of a state s which follows a particular policy π is defined as the expected discounted cumulative reward as shown in Eq. (1), where γ is a fixed discount factor, and $R(s_t, a_t, s_{t+1})$ is the immediate reward received at each time step for transitioning to state s_{t+1} after taking action a_t in state s_t.

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s\right] \tag{1}$$

Similarly, the Q-value for a state-action pair as defined in Eq. (2) is prescribed as the expected discounted sum of rewards of doing an action a in state s, and thereafter following a policy π.

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a\right] \tag{2}$$

The value iteration starts with initializing the value function $V_0(s)$ arbitrarily for all states. And then, $V(s)$ and $Q(s)$ are refined according to Eqs. (3) and (4), respectively, in each iteration.

$$V_{n+1}(s) = \max_a Q_n(s, a) \tag{3}$$

$$Q_{n+1}(s, a) = \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V_n(s')) \tag{4}$$

In this manner, the value of each state is updated repeatedly until there is no significant change in the value function or the maximum number of iterations is reached. Subsequently, the optimal policy can be derived based on equation below.

$$\pi^* = \operatorname*{argmax}_a Q^*(s, a) \tag{5}$$

3. Methods

In our TDPP-Net, the policy is trained end-to-end through supervised imitation learning [34] and a large number of data with labeled optimal actions are generated using the Dijkstra's algorithm. The inputs of the network comprise the 3D grid map of obstacles, the current position of the agent, and the goal location. The output of the network is the one-hot representation of an action in 3D space. Since there are 26 achievable actions considering a 3D grid world, the 3D path planning problem can be regarded as a classification task with 26 classes in total. Therefore, the training of TDPP-Net involves the back-propagation to minimize the cross entropy loss between the outputs and the labeled actions over all training samples.

Aiming to yield optimal policies applicable to unseen scenarios, the TDPP-Net is required to figure out the goal-directed nature of path planning tasks so that its performance does not depend on specific maps. Therefore, our network embeds the value iteration modules which are capable to explicitly learn planning computations. However, instead of resolving the 3D path planning problem by emulating the value iteration processes through 3D CNNs, our approach adopts the idea of action decomposition and composition, which contributes to significantly improved performance using merely 2D CNNs.

In general, as illustrated in Fig. 1, our TDPP-Net is composed of three core modules, including a decomposition module, a value iteration module, and a composition module.

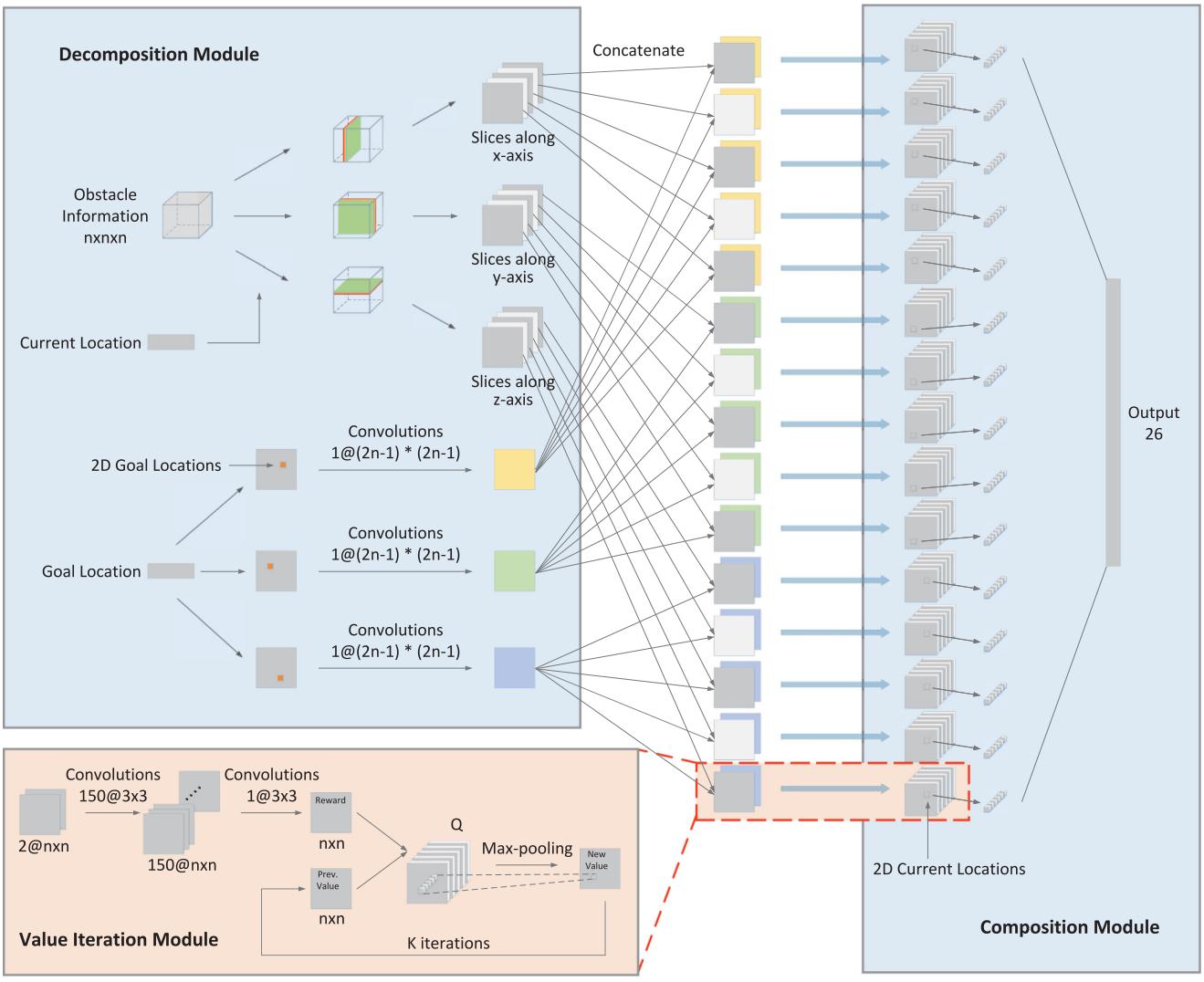


Fig. 1. Architecture of TDPP-Net. Inputs of the network include the current position of the agent, the goal location, and the 3D grid map of obstacles. Output of the network is the one-hot representation of a 3D action. The key components of TDPP-Net include the decomposition module, the value iteration module, and the composition module.

3.1. Decomposition module

The decomposition module is targeted to decompose the original 3D action and convert the problem into resolving actions on multiple 2D planes so that each 2D action can be considered separately. As demonstrated in Section 4, this will lead to much better performance compared to solving the 3D action as a whole.

In the first place, the range of motion is assumed to be restricted, which is reasonable in 3D grid worlds. Therefore, because of the local connectivity, the action in each direction depends only on a subspace of the 3D grid world. Besides, it is also intuitive that the optimal 3D action is related to the merits of the actions on surrounding 2D planes. That is, the optimal action should lead the agent towards the most desirable resultant direction. Therefore, in a particular state, the 3D action can be decided based on the Q-values of the 2D planes which are within the neighborhood of the agent's current location. As a result, our decomposition module, in a sense, refers to an attention mechanism [35] with merely relevant subspaces being attended to.

As depicted in Fig. 1, this form of attention is achieved in TDPP-Net by extracting concerned slices of the 3D obstacle map along all three axes and transforming the goal information from one 3D vector into three 2D goal maps. Specifically, as mentioned above, only the obstacles surrounding the current location of the agent is of attention and hence being extracted. Also, it is worth noting that instead of extracting exactly three slices along each axis, a more appropriate number of slices can be extracted to take more adequate information into account when predicting the desired action.

Specifically, in our implementation, the input 3D grid map of size $N \times N \times N$ contains all the obstacles which are randomly generated. Besides, in the meantime, the current and goal locations of the agent are also randomly determined while guaranteed to be collision-free. After the 3D grid map is fed into the network, five surrounding slices of the grid map are acquired along each axis. That is, if the current position of the agent in the 3D grid map is (x, y, z), then the $x-2$, $x-1$, x, $x+1$, and $x+2$-th slices along the x direction, the $y-2$, $y-1$, y, $y+1$, and $y+2$-th slices along the y direction, as well as the $z-2$, $z-1$, z, $z+1$, and $z+2$-th slices along the z direction will be extracted. As a result, a total of fifteen 2D obstacle maps of size $N \times N$ are generated.

In addition, the original 3D goal vector is converted into three 2D locations by projecting the 3D vector directly onto the three 2D planes. For instance, if a 3D goal is located at (i, j, k), the corresponding 2D goals generated on the xy-plane, xz-plane and yz-plane are (i, j), (i, k) and (j, k), respectively. The 2D locations are then transformed into 2D goal maps accordingly by one-hot

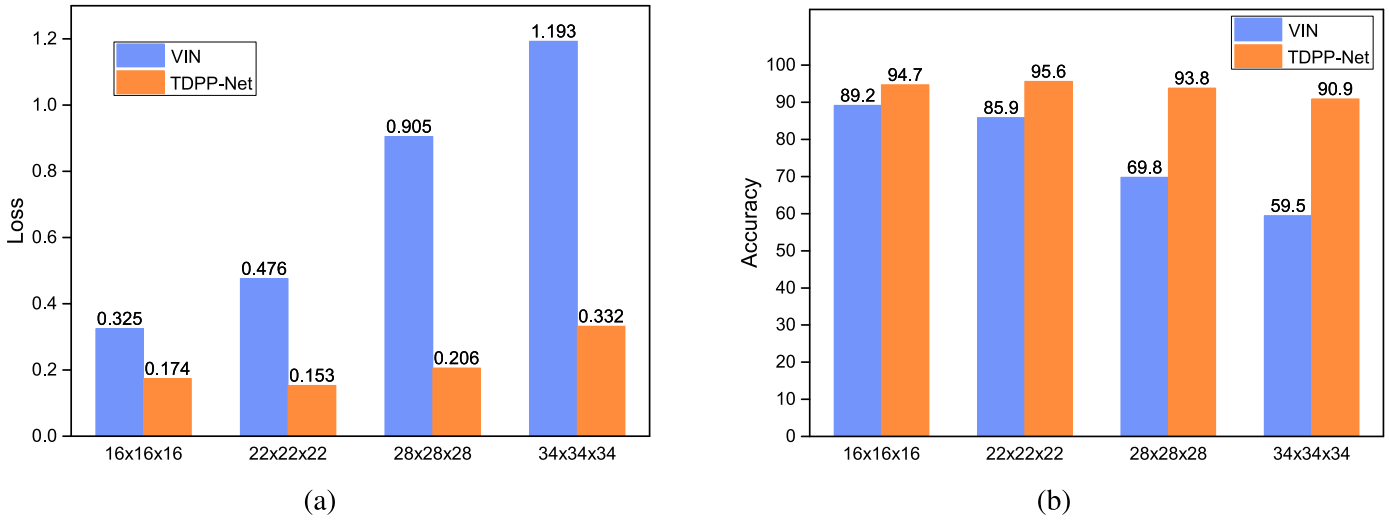


Fig. 2. Training loss and accuracy of VIN and TDPP-Net.

encoding. Next, a convolutional layer is applied to each 2D goal map for spreading information of the goal to other states in the grid map. Since the goal information is conveyed to each grid, less iteration is required for the value iteration module.

Finally, the 2D obstacle and goal maps are concatenated together correspondingly as the inputs of the value iteration modules.

3.2. Value iteration module

The value iteration modules of TDPP-Net emulate the processes of value iteration in all the queried 2D planes.

The value iteration module embeds an explicit planning computation by approximating the Bellman update through CNNs. Referring to VIN proposed in [20], a state in our value iteration module denotes the agent's position on a 2D plane mapped from its original 3D location. In addition to the state, an observation of the state also includes one slice extracted from the 3D obstacle map to indicate the accessibility domain in the 2D plane, and the corresponding 2D feature map which encodes the information of the goal. Then, both the rewards $R(s, a, s')$ and the transition probabilities $P(s'|s, a)$ are related to the observations $\phi(s)$ via CNNs. Firstly, a CNN layer with 150 kernels of size 3×3 is applied to the input, followed by a second CNN layer with one kernel of size 3×3 to create a reward layer which is expected to distinguish among obstacle areas, collision-free areas as well as goal regions. The transition probabilities are then also represented using CNNs with 3×3 filters since the transitions in grid worlds are local. In this way, the reward and the transition probabilities for a state can both be estimated from its immediate surroundings.

Imitating the process of value iteration, the obtained reward layer and a initialized value layer are first fed into a convolutional layer with each channel corresponding to the Q-value for a particular action and kernel parameters corresponding to the transition probabilities. Subsequently, the value layer is updated through a max-pooling layer, which conforms to the value update step specified in Eq. (3). The updated value layer is then stacked with the reward and fed back into the convolutional layer in the next iteration to calculate the Q-values as stated in Eq. (4). The convolution and max-pooling operations are performed repeatedly for K iterations to obtain the final values. Originally, the number of iterations should be decided in consideration of the map size so that the information from the goal state is propagated to all other states. However, in TDPP-Net, the iteration number can be less dependent on the map size since the goal information has already been spread in the decomposition module, which makes it more efficient and competitive in large-scale environments.

Ultimately, the outputs of the value iteration modules are passed to the composition module.

3.3. Composition module

In general, the composition module is designed to mimic the action composition process for generating the resultant action in 3D space. The inputs to this module contain graphs of the Q-values resulted from the value iteration modules, and the current 2D states obtained as follows. Supposing the original 3D location of the agent is (x, y, z), then the 2D states corresponding to the value graphs generated for the slices along x axis are (y, z). Similarly, those corresponding to the slices along y and z axes are (x, z) and (x, y), respectively.

Next, considering the attention mechanism, our composition module extracts the Q-values that correspond to the current states from all the Q-value graphs. And these queried values are then fed into a fully connected softmax output layer. In this way, the composition module can be interpreted implicitly as the exploration of a 3D action through the combination of the desirable actions on all the investigated planes indicated by the corresponding Q-values.

Furthermore, it is worth noting that although the MDP learnt in TDPP-Net is not the true MDP, its optimal plan contains useful knowledge to update the true one iteratively. As a result, the learnt MDP approximates the original MDP to a large degree and the resultant policy is capable to achieve similar performance as the true optimal policy. Therefore, once sufficiently trained, the proposed deep neural network architecture is competent to plan collision-free paths towards the goal locations in unseen 3D environments.

To evaluate its performance, extensive experiments have been conducted and the results are presented in the following section.

4. Experiment results

In this section, we evaluate the proposed DNN-based 3D path planner through experiments conducted based on Tensorflow in four different scenarios. In the first scenario, grid worlds of size $16 \times 16 \times 16$ are generated with 30–50 random obstacles placed in each world. The generated obstacles are allowed to overlap and the maximum edge length of an obstacle is set to five. Similarly, the second, third, and forth scenarios include grid worlds

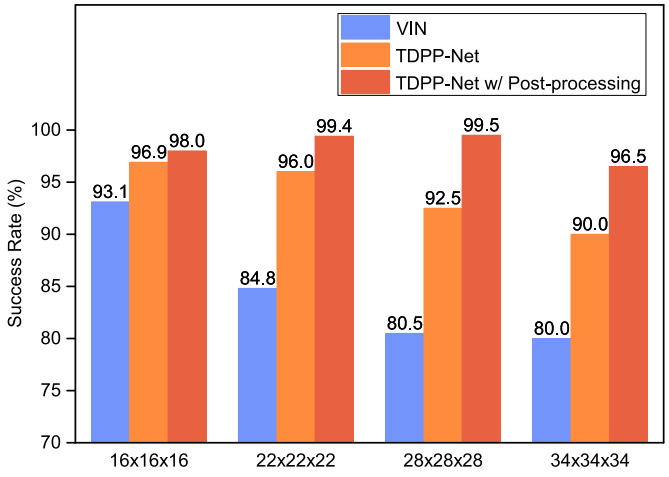


Fig. 3. Success rate of VIN, TDPP-Net, and TDPP-Net w/Post-processing.

of size $22 \times 22 \times 22$, $28 \times 28 \times 28$, and $34 \times 34 \times 34$ with 60–80, 80–100, and 100–150 random obstacles, respectively. The environments generated for training and testing data are different in order to evaluate the performance of TDPP-Net in unseen situations.

Since 3D path planning problems are rarely tackled with DNNs, there lacks report on the state-of-art performance in planning 3D paths via DNNs. In fact, to the best of our knowledge, TDPP-Net is the first deep neural network architecture designed for 3D path planner to date. Therefore, we extend VIN to deal with 3D path planning problems by replacing its 2D CNNs by 3D ones. The sizes of all the convolutional kernels in VIN are $3 \times 3 \times 3$ and the number of channels in the Q-value layer is set to 40.

During evaluation of DNN-based planners, the next state is determined by taking the action predicted by a DNN at the current state and in turn, the predicted state is regarded as the input to the network for searching the next state. As indicated in Fig. 2, TDPP-Net outperforms VIN in all circumstances in terms of both training loss and accuracy. And the superiority of TDPP-Net becomes more noticeable with increased map size, denoting that TDPP-Net exhibits better generalization capability to large-scale environments than VIN.

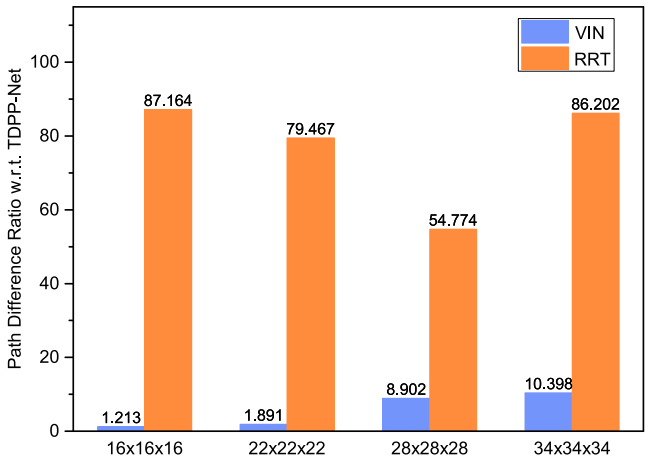
This conclusion is further confirmed considering two additional metrics, including success rate and path difference, as shown in Fig. 3 and Table 1, respectively. The success rate indicates the probability of reaching the goal state successfully from the start state without colliding with any obstacle by following the predicted actions. And for the successful planned paths, the path length is defined as the sum of distances between consecutive waypoints and the path difference is defined as the difference in path length between the predicted and the ground truth paths. As shown in Fig. 3, the success rate of VIN exceeds 90% only in the $16 \times 16 \times 16$ worlds and drops dramatically in larger-scale environments. However, Although the success rate of TDPP-Net also decreases a bit with increased environmental scale, it is above 90% in all scenarios.

Table 1
Comparison of different 3D path planners in terms of average path difference in various scenarios.

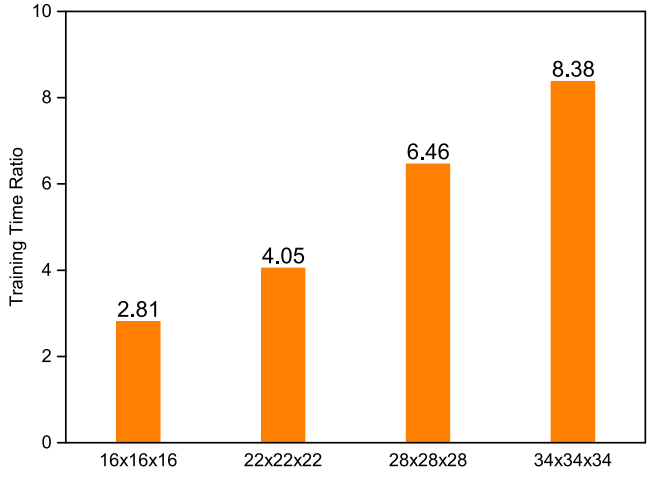
Method	Average path difference			
	$16 \times 16 \times 16$	$22 \times 22 \times 22$	$28 \times 28 \times 28$	$34 \times 34 \times 34$
RRT	5.230	7.311	8.983	9.741
VIN	0.074	0.174	1.460	1.175
TDPP-Net	0.060	0.092	0.164	0.113
TDPP-Net w/ Post-processing	0.069	0.120	0.210	0.128

Additionally, according to Table 1 and Fig. 4(a), the path difference of VIN is also larger compared to TDPP-Net, which reveals that the paths planned by TDPP-Net is more optimal with less detours. According to Table 1, the path quality of TDPP-Net is close to that resulted from the optimal path planning approaches, such as A* and Dijkstra's algorithm, because of its small training loss and high training accuracy. However, compared to TDPP-Net, the path difference of VIN is larger and this disadvantage becomes more apparent in larger-scale environments according to Fig. 4(a). With higher success rate and smaller path difference, TDPP-Net can be better generalized to large-scale environments compared to VIN, which is consistent with the conclusion from comparing training loss and accuracy. Besides, as depicted in Fig. 4(b), the TDPP-Net is also trained in much less time compared to VIN.

It is worth mentioning that an additional efficient post-processing strategy is also utilized and evaluated. Because real-world applications impose a high requirement on success rate, the post-processing scheme is aimed to further increase the success rate of the proposed TDPP-Net based 3D path planner through local adjustment during implementation for engineering

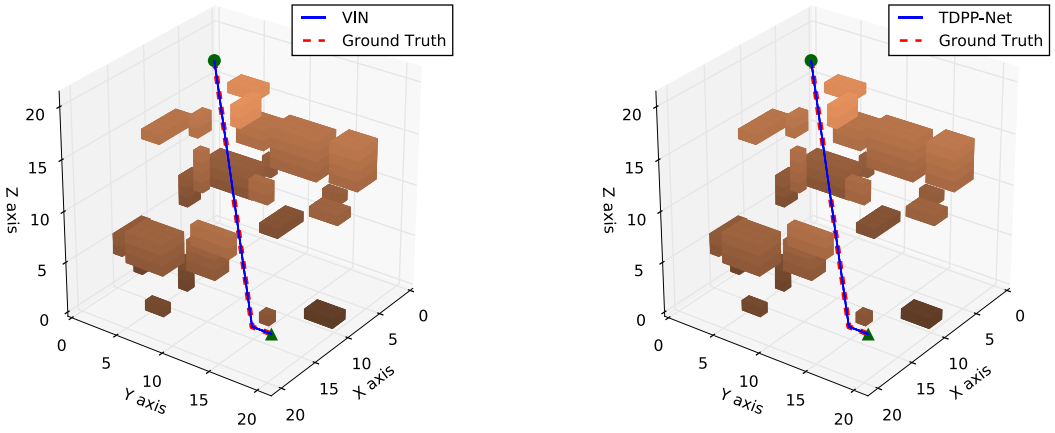


(a) Path difference ratio with respect to TDPP-Net

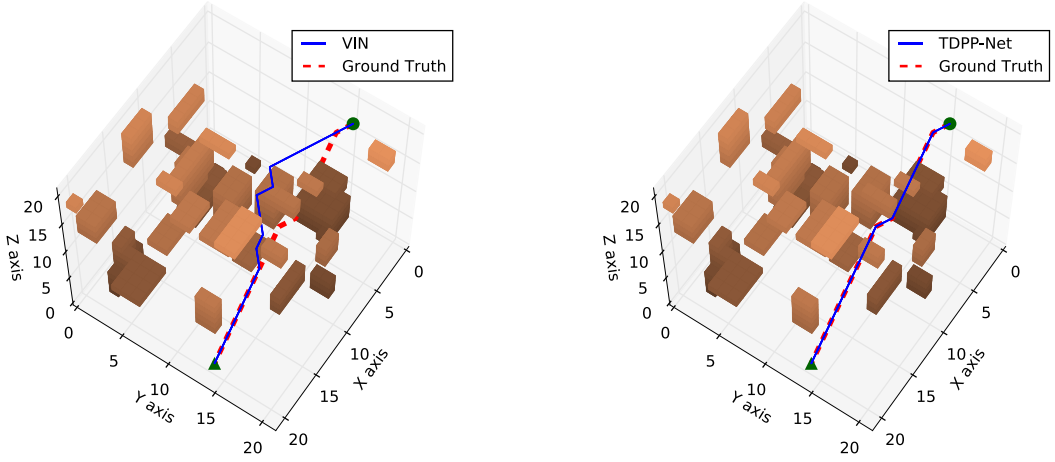


(b) Training time ratio of VIN with respect to TDPP-Net

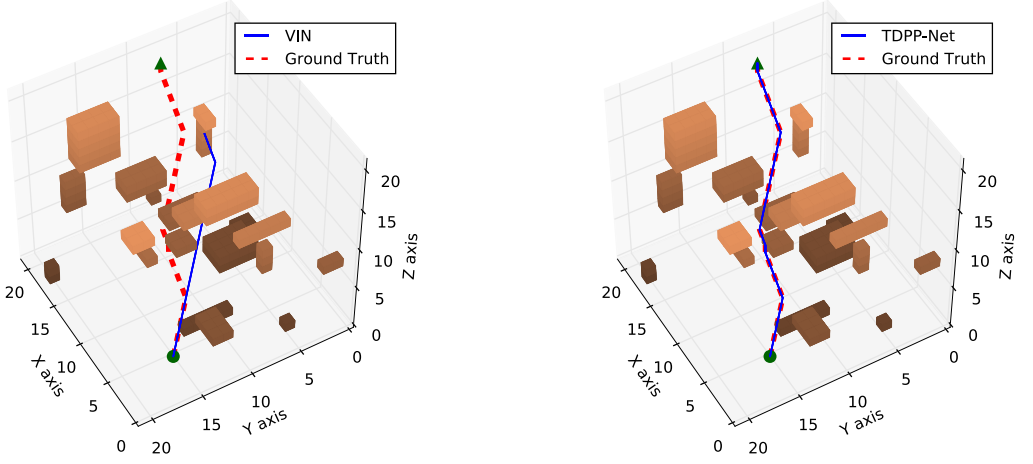
Fig. 4. Path difference ratio and training time ratio with respect to TDPP-Net.



(a) In the first scenario, both VIN and TDPP-Net succeed in predicting the optimal path

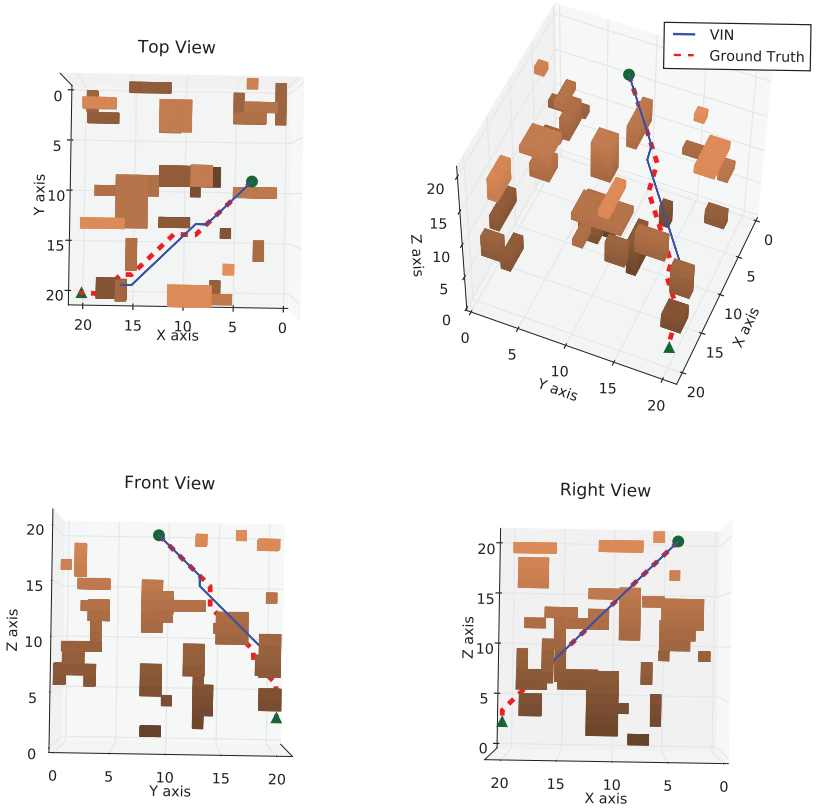


(b) In the second scenario, TDPP-Net yields the optimal path while VIN fails to find the shortest path

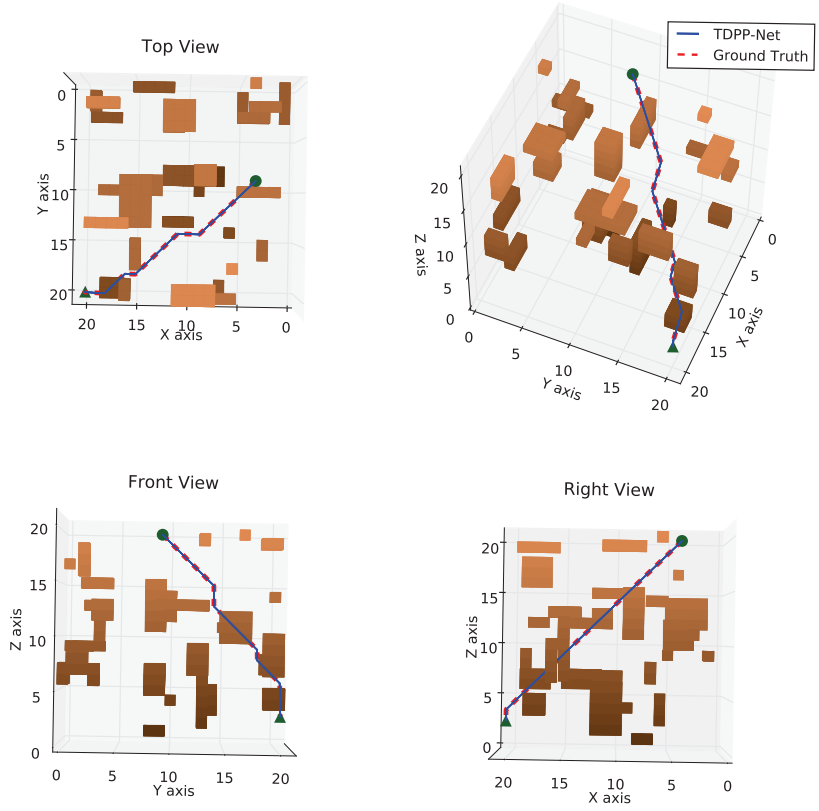


(c) In the third scenario, TDPP-Net finds the shortest path while VIN fails to predict a valid path

Fig. 5. Examples of paths predicted by VIN and TDPP-Net in three unseen 3D environments. The obstacles are represented by solid voxels with their darkness indicating the heights of the obstacles. The start and goal locations are highlighted using green circles and triangles, respectively. The ground truth shortest paths are marked in red dashed lines while the paths predicted by the DNNs are shown in blue solid lines.

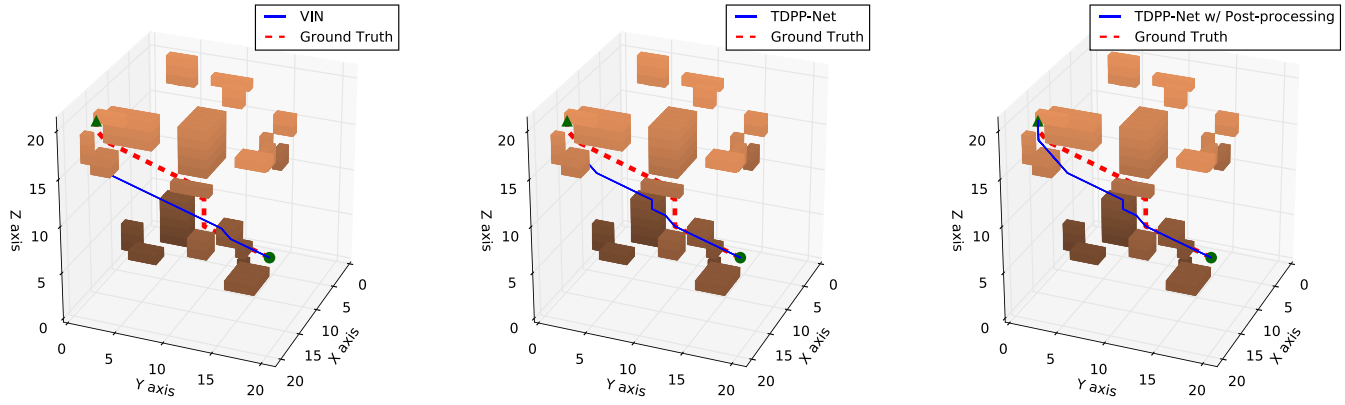


(a) Perspective, top, front and right views of the path predicted by VIN

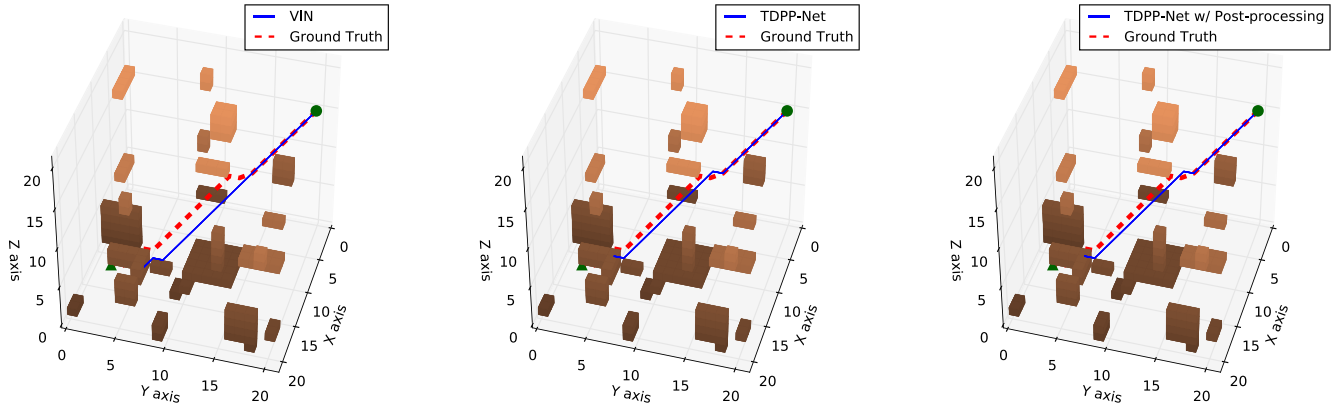


(b) Perspective, top, front and right views of the path predicted by TDPP-Net

Fig. 6. The 3D perspective views and corresponding 2D orthographic projections of paths predicted by VIN and TDPP-Net, respectively, in the same unseen 3D environment. The ground truth shortest paths are marked in red dashed lines while the paths predicted by the DNNs are shown in blue solid lines.



(a) In the first environment, both VIN and TDPP-Net fail to find an admissible path while a near optimal path is predicted by adding the post-processing strategy to TDPP-Net



(b) In the second environment, all three planners fail to yield a valid path

Fig. 7. Examples of paths predicted by VIN, TDPP-Net, and TDPP-Net with post-processing in two unseen 3D environments. The obstacles are represented by solid voxels with their darkness indicating the heights of the obstacles. The start and goal locations are highlighted using green circles and triangles, respectively. The ground truth shortest paths are marked in red dashed lines while the paths predicted by the DNNs are shown in blue solid lines.

Table 2
Comparison of different 3D path planners in terms of average computational time in various scenarios.

Method	Average computational time per path (s)			
	$16 \times 16 \times 16$	$22 \times 22 \times 22$	$28 \times 28 \times 28$	$34 \times 34 \times 34$
A*	0.019 ± 0.007	0.069 ± 0.026	0.216 ± 0.104	0.401 ± 0.201
RRT	0.008 ± 0.007	0.019 ± 0.012	0.026 ± 0.018	0.035 ± 0.089
VIN (GPU)	0.021 ± 0.003	0.075 ± 0.004	0.197 ± 0.025	0.386 ± 0.054
VIN (CPU)	0.587 ± 0.053	2.669 ± 0.138	8.621 ± 1.091	18.000 ± 2.537
TDPP-Net (GPU)	0.060 ± 0.006	0.091 ± 0.005	0.120 ± 0.011	0.136 ± 0.016
TDPP-Net (CPU)	0.090 ± 0.007	0.237 ± 0.015	0.517 ± 0.043	0.926 ± 0.110
TDPP-Net w/ Post-processing (GPU)	0.060 ± 0.005	0.098 ± 0.006	0.120 ± 0.009	0.138 ± 0.017
TDPP-Net w/ Post-processing (CPU)	0.090 ± 0.009	0.254 ± 0.019	0.543 ± 0.056	0.933 ± 0.109

applications. Generally, if the post-processing strategy is performed, the TDPP-Net based 3D path planner will attempt to output an updated action if the predicted one causes the agent to bump into an obstacle.

Specifically, also referring to the action decomposition and composition mechanism, the rationale of post-processing strategy is to investigate the components of an action separately. First, one of the three action components is neglected to check if the collision can be avoided. However, if the agent will still end up colliding with an obstacle by adopting the updated actions, two action components will then be ignored if all three components are nonzero. In this way, there is a high chance of solving a challenging collision problem while preserving the direction of the predicted action to the maximum extent. By employing the post-processing, the success rate of the proposed 3D path planner is increased and maintained high even in large-scale environments as shown in Fig. 3, which makes the planner more practical. Moreover, the reduction of success rate in larger environments is almost eliminated after introducing the post-processing scheme so that no significant influence is exerted on the performance of our proposed path planner by increasing the map size.

Table 3
Comparison between VIN and TDPP-Net in terms of average computational time per action prediction.

Method	Average computational time per action prediction (s)			
	$16 \times 16 \times 16$	$22 \times 22 \times 22$	$28 \times 28 \times 28$	$34 \times 34 \times 34$
VIN	0.0018	0.0043	0.0083	0.0154
TDPP-Net	0.0053	0.0056	0.0057	0.0057

However, it is also observed from Table 1 that while the success rate is improved due to addition of the post-processing strategy, the average difference in path length is also slightly increased because the post-processing is aimed to resolve the dilemmas instead of searching the optimal states. However, the minor increase in path length is acceptable and the average path difference is still considerably smaller than VIN. Therefore, the post-processing scheme improves the overall performance of the proposed TDPP-Net based 3D path planner considering the apparently improved success rate and slightly increased path difference.

To present in a more intuitive way, VIN and TDPP-Net are utilized to plan paths in three different unseen circumstances and the results are provided in Fig. 5. For the ease of visualization, the paths predicted by the two networks are demonstrated separately. In Fig. 5, the blue lines indicate the paths predicted by the two networks and the red dashed lines show the ground truth paths planned by the Dijkstra's algorithm. The obstacles are represented using solid voxels with their darkness indicating the altitudes of the obstacles. The start and goal locations are highlighted by green circles and green triangles, respectively. Each row represents the results in one $22 \times 22 \times 22$ environment. In the first scenario, both DNN-based path planners succeed in finding the optimal path. However, in the second scenario, VIN fails to find the shortest path and in the third environment, VIN fails to predict a valid path since an predicted action causes the agent to bump into an obstacle. On the contrary, TDPP-Net based 3D planner succeed in finding the shortest paths in both these two cases.

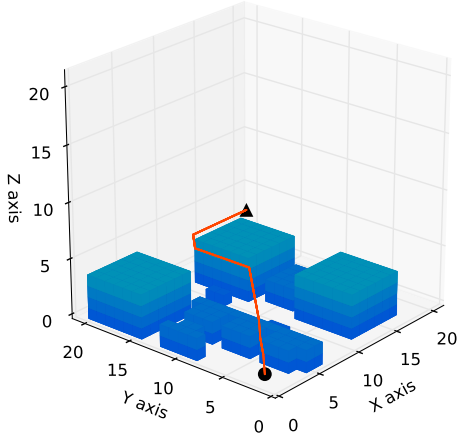
Besides 3D views, 2D orthographic projections of the paths predicted by VIN and TDPP-Net are also provided in Fig. 6, respectively. Similar to the environment demonstrated in Fig. 5(c), the VIN fails to find an admissible path in the new 3D environment presented in Fig. 6. However, TDPP-Net succeed in find the optimal shortest path. It can be observed that in all the failure cases of VIN, the actions predicted by the network tend to drive the agent to the target position through an alternative path. Since the optimal path is not unique for many cases, an alternative shortest path is possible to be predicted by the networks after they have learned the nature of planning. Nevertheless, there will exist a higher chance for a planning network to fail if its planned paths divert from the ground truth paths more frequently because of the large training loss.

Despite the much lower training loss of TDPP-Net, in the environments demonstrated in Fig. 7, both VIN and TDPP-Net fail to find the valid paths because of the deviation from the ground truth path. However, by introducing the post-processing strategy, the problematic action can be modified locally to discover a valid path towards the goal with a higher probability. As shown in Fig. 7(a), although TDPP-Net fails to predict an admissible path in this environment, after adding the post-processing scheme, the TDPP-Net based planner is capable of finding an accessible path under the same condition with a path difference equaling to 0.0964. Nonetheless, although with the addition of post-processing, TDPP-Net based 3D planner becomes superior to deal with most of the tricky situations as depicted in Fig. 3, it still fails to predict a collision-free path in the last scenario shown in Fig. 7(b). In this situation, the goal is encompassed by a larger amount of obstacles, which makes the task even more difficult. A possible solution to this challenge is to include more cases with complex obstacles around the start and goal locations in the training dataset.

Regarding the computational time, comparison among five different 3D planners, including the widely adopted A* and Rapidly-exploring Random Trees (RRT) algorithms, the VIN, the TDPP-Net, and the TDPP-Net with additional post-processing scheme, is conducted and presented in Table 2. Both A* and RRT are representative works with the former being complete and optimal [36] while the later being probabilistically complete and well-known for its speed [37]. It is worth noting that since the classical methods have low potential for parallelization [7], they are only implemented using CPU during the experiments. However, the DNN-based planners are executed both with and without the acceleration of GPU.

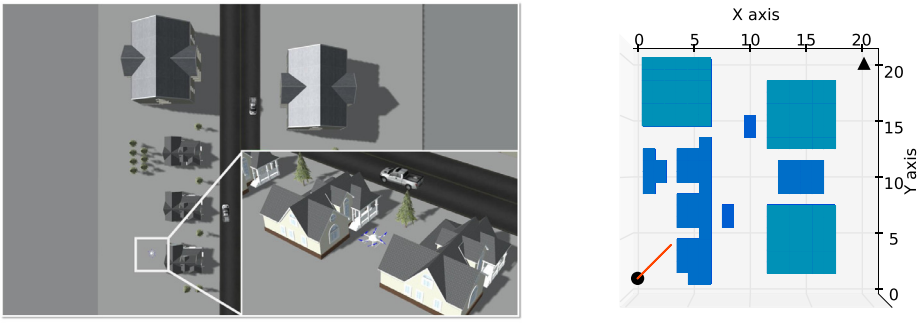
First, the average length of the ground truth paths in the $16 \times 16 \times 16$ testing worlds is 11.16 and those of the ground truth paths in $22 \times 22 \times 22$, $28 \times 28 \times 28$, and $34 \times 34 \times 34$ testing worlds are 17.15, 22.12, and 24.71, respectively. It can be noticed that the computational times of the GPU-accelerated TDPP-Net in different scenarios are almost proportional to the average path lengths,

(a) Simulation scenario in Gazebo

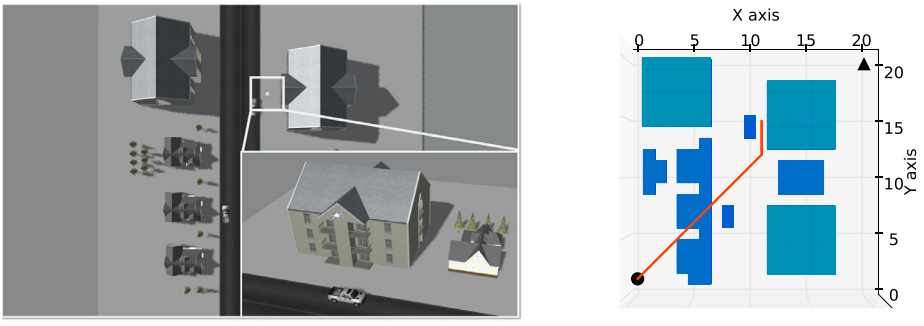


(b) Path predicted by TDPP-Net in the 3D grid world

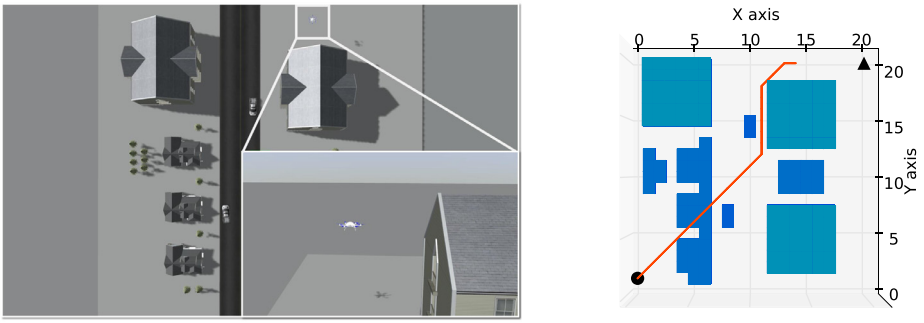
Fig. 8. Simulation of planning 3D paths for an autonomous quadrotor using TDPP-Net.



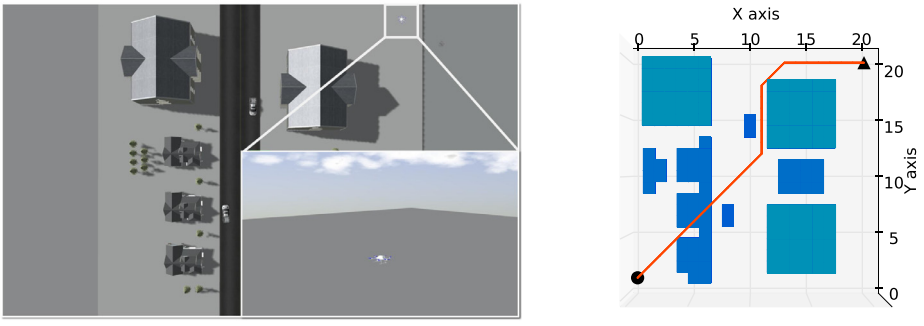
(a) Intermediate step 1



(b) Intermediate step 2



(c) Intermediate step 3



(d) Final step

Fig. 9. Intermediate steps of the simulation in Fig. 8. In the simulation, the autonomous quadrotor is expected to follow the waypoints planned by the TDPP-Net. The screenshots of three interim steps and the final step are shown on the left side. The corresponding top views of the planning results in the 3D grid world are demonstrated on the right side.

revealing that the computational time of TDPP-Net for each action prediction is steady regardless of the increased environmental size. This is because the iteration number of its value iteration module is kept constant during the tests. Therefore, although TDPP-Net requires longer time than A* and VIN to figure out the whole path in smaller environments, it takes much less time in larger worlds. Besides, when GPU is not used, VIN will cost much longer time for computation and becomes incapable to realize real-time performance. For instances, compared to TDPP-Net, VIN takes more than 16 times longer in the $28 \times 28 \times 28$ world and more than 19 times longer when evaluated in the $34 \times 34 \times 34$ world.

Furthermore, it is also noted that due to the simplicity of the post-processing scheme, no significant burden in computation is detected after introducing it during the implementation of TDPP-Net. Therefore, the post-processing strategy is expected to improve the success rate almost without increasing the computational time in practical applications. In addition to average computational time per path, it is also noteworthy that the standard deviation of the computational time is smaller when adopting the GPU-accelerated TDPP-Net in comparison to both A* and RRT algorithms, which implies that the computational time of TDPP-Net is less dependent on the complexity of environment. Last but not least, although the RRT algorithm is faster than the other algorithms, it generally ends up with paths with larger path difference as depicted in Fig. 4(a) and Table 1. Therefore, by considering the success rate, the path difference, the average computational time as well as the standard deviation of computational time, TDPP-Net with addition of the post-processing strategy is more promising and competitive for 3D path planning tasks, especially in large-scale environments.

As mentioned above, the computational time per action prediction of TDPP-Net is almost constant. This is further verified by calculating the average computational times of the networks for one action prediction as presented in Table 3. In contrast, the computational time per action prediction of VIN increases with increased map size, indicating that TDPP-Net becomes more preferable in larger-scale environments also in terms of computational time per action prediction.

More importantly, for conventional algorithms, additional computation is required for re-planning in dynamic environments. For example, the costs of affected nodes need to be recalculated in grid based methods and the trees need to be pruned and regrown in sampling based approaches. Nevertheless, in contrast to conventional algorithms, each prediction resulted from TDPP-Net is independent and costs approximately invariant computational time due to the end-to-end nature. Therefore, TDPP-Net is more talented to planning 3D paths in dynamic environments with no additional effort required for re-planning. In addition, TDPP-Net is capable to predict credible actions for an agent to proceed even if the goal location is temporarily inaccessible while the conventional methods will fail in such cases, which further presents the significance of TDPP-Net.

Finally, the proposed network is applied to plan paths for a simulated autonomous quadrotor during navigation in a cluttered 3D environment as shown in Fig. 8. The simulation scenario is indicated in Fig. 8(a) and the 3D path predicted by TDPP-Net is illustrated in Fig. 8(b). Specifically, in Fig. 8(b), the predicted 3D path is highlighted in red and the blue solid voxels represent the obstacles with their darkness indicating the heights of the obstacles. Besides, three interim steps and the final state of the simulation are also demonstrated in Fig. 9. Specifically, the simulated quadrotor is expected to ascend while maneuver towards the goal at the beginning to avoid collision with houses. Also, it is required to detour during the latter half of the navigation to avoid collision with the apartment. In Fig. 9, the screenshots captured in Gazebo are shown on the left side while the top views of the planning results in the 3D grid world are also provided on the right side. Generally, the simulation presents the potential of TDPP-Net in real-world applications.

5. Conclusion

A novel end-to-end neural network architecture named TDPP-Net has been introduced in this paper for 3D path planning. Embedding the concept of action decomposition and composition, the proposed path planner is capable of predicting 3D actions through focusing on partial obstacle information and implementing value iterations in 2D planes. Moreover, an additional post-processing scheme has been presented to further improve the success rate of the proposed DNN-based path planner.

The experimental results have demonstrated that our network exhibits significantly improved performance compared to VIN when generalized to new situations in terms of training loss, training accuracy, success rate and path difference. Besides, the proposed DNN-based path planner also addresses the bottlenecks of conventional path planning algorithms. In contrast to the classical path planners, the computational time of TDPP-Net is steady in regardless of the increased environmental scale and complexity. Moreover, since each prediction is independent and plausible actions can be provided even if the goal is temporarily inaccessible, TDPP-Net is more promising than existing planners in dynamic environments without additional computational efforts.

In the future, we plan to decrease the computational time while further increase the success rate of the proposed DNN-based path planner by modifying the network architecture. Also, we desire to develop DNN-based path planners to deal with local 3D path planning tasks in dynamic environments.

Conflict of interest

None.

Acknowledgment

This work is supported by the ST Engineering-NTU Corporate Lab.

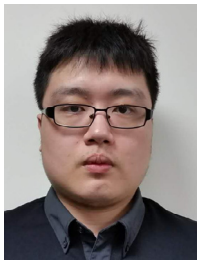
References

[1] X. Yang, H. Luo, Y. Wu, Y. Gao, C. Liao, K.-T. Cheng, Reactive obstacle avoidance of monocular quadrotors with online adapted depth prediction network, Neurocomputing 325 (2019) 142–158.
[2] F.-P. Tian, W. Feng, Q. Zhang, X. Wang, J.-Z. Sun, V. Loia, Z.-Q. Liu, Active camera relocalization from a single reference image without hand-eye calibration, IEEE Trans. Pattern Anal. Mach. Intell. (2018). 1–1
[3] D. Miao, F.-P. Tian, W. Feng, Active camera relocalization with rgbd camera from a single 2d image, in: Proceedings of International Conference on Acoustics, Speech, and Signal Processing, ICASSP, 2018.
[4] K. Wu, M. Abolfazli Esfahani, S. Yuan, H. Wang, Learn to steer through deep reinforcement learning, Sensors 18 (11) (2018) 3650.
[5] F. Yan, Y.-S. Liu, J.-Z. Xiao, Path planning in complex 3d environments using a probabilistic roadmap method, Int. J. Autom. Comput. 10 (6) (2013) 525–533.
[6] M. Likhachev, D. Ferguson, Planning long dynamically feasible maneuvers for autonomous vehicles, Int. J. Robot. Res. 28 (8) (2009) 933–945.
[7] A. Kanezaki, J. Nitta, Y. Sasaki, Goselo: goal-directed obstacle and self-location map for robot navigation using reactive neural networks, IEEE Robot. Autom. Lett. 3 (2) (2018) 696–703.
[8] M. Liang, X. Hu, Recurrent convolutional neural network for object recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3367–3375.
[9] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, A convolutional neural network cascade for face detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5325–5334.
[10] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[11] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
[12] S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 221–231.

[13] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, R. Vaughan, Learned map prediction for enhanced mobile robot exploration, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2019.
[14] Y. Lu, S. Yi, Y. Liu, Y. Ji, A novel path planning method for biomimetic robot based on deep learning, Assem Autom. 36 (2) (2016) 186–191.
[15] T. Weber, S. Racanière, D.P. Reichert, L. Buesing, A. Guez, D.J. Rezende, A.P. Badia, O. Vinyals, N. Heess, Y. Li, et al., Imagination-augmented agents for deep reinforcement learning, in: Proceedings of Advances in Neural Information Processing Systems, NIPS, 2017, pp. 5690–5701.
[16] E. Rehder, F. Wirth, M. Lauer, C. Stiller, Pedestrian prediction by planning using deep neural networks, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 1–5.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[18] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al., Learning to navigate in complex environments, in: Proceedings of International Conference on Learning Representations, ICLR, 2017.
[19] A. Giusti, J. Guzzi, D.C. Cireşan, F.-L. He, J.P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al., A machine learning approach to visual perception of forest trails for mobile robots, IEEE Robot. Autom. Lett. 1 (2) (2016) 661–667.
[20] A. Tamar, Y. Wu, G. Thomas, S. Levine, P. Abbeel, Value iteration networks, in: Proceedings of Advances in Neural Information Processing Systems, 2016, pp. 2154–2162.
[21] R. Ilin, R. Kozma, P.J. Werbos, Efficient learning in cellular simultaneous recurrent neural networks-the case of maze navigation problem, in: Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL, IEEE, 2007, pp. 324–329.
[22] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, et al., The Predictron: end–to-end learning and planning, in: Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, pp. 3191–3199.
[23] S. Niu, S. Chen, H. Guo, C. Targonski, M.C. Smith, J. Kovačević, Generalized value iteration networks: life beyond lattices, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI, 2018.
[24] M. Okada, L. Rigazio, T. Aoshima, Path integral networks: end-to-end differentiable optimal control, 2017 arXiv preprint arXiv: 1706.09597.
[25] J. Oh, S. Singh, H. Lee, Value prediction network, in: Proceedings of Advances in Neural Information Processing Systems, 2017, pp. 6120–6130.
[26] T. Shankar, S.K. Dwivedy, P. Guha, Reinforcement learning via recurrent convolutional neural networks, in: Proceedings of 2016 23rd International Conference on Pattern Recognition, ICPR, IEEE, 2016, pp. 2592–2597.
[27] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, Cognitive mapping and planning for visual navigation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 2616–2625.
[28] E. Rehder, M. Naumann, N.O. Salscheider, C. Stiller, Cooperative motion planning for non-holonomic agents with value iteration networks, 2017 arXiv:1709.05273.
[29] S. Mittal, K. Deb, Three-dimensional offline path planning for uavs using multiobjective evolutionary algorithms, in: Proceedings of IEEE Congress on Evolutionary Computation, CEC, IEEE, 2007, pp. 3195–3202.
[30] M. Ataei, A. Yousefi-Koma, Three-dimensional optimal path planning for waypoint guidance of an autonomous underwater vehicle, Robot. Auton. Syst. 67 (2015) 23–32.
[31] D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, Dynamic Programming and Optimal Control, 1, Athena Scientific, Belmont, MA, 1995.
[32] A. Kolobov, Planning with Markov Decision Processes: An AI Perspective, Synthesis Lectures on Artificial Intelligence and Machine Learning, 6, Morgan & Claypool Publishers, 2012, pp. 1–210.
[33] R. Bellman, Dynamic Programming, Courier Corporation, 2013.
[34] S. Ross, G.J. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in: Proceedings of International Conference on Artificial Intelligence and Statistics, 2011, pp. 627–635.
[35] V. Mnih, N. Heess, A. Graves, et al., Recurrent models of visual attention, in: Proceedings of Advances in Neural Information Processing Systems, 2014, pp. 2204–2212.
[36] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.
[37] S.M. LaValle, J.J. Kuffner Jr, Randomized kinodynamic planning, Int. J. Robot. Res. 20 (5) (2001) 378–400.

Keyu Wu received the B.Eng. degree in Bioengineering from National University of Singapore, Singapore, in 2013. She is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Her current research interests include deep reinforcement learning, deep learning, path planning, trajectory generation, and autonomous robots.

Mahdi Abolfazli Esfahani received his B.Eng. and M.Eng. in Computer Engineering, and Artificial Intelligence and Robotics from Ferdowsi University of Mashhad, Iran, respectively. He was the leader of Nexus RoboCup soccer simulation team and achieved more than 20 national and international awards. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. His research interests include deep learning, Visual-Inertial Odometry, and SLAM.



Shenghai Yuan is currently pursuing the Ph.D. degree with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. His research interests include Visual Odometry, Stereovision, and autonomous robotics system.

Han Wang is currently an Associate Professor with the School of Electrical and Electronics Engineering, Nanyang Technological University. He received his Bachelor degree in Computer Science from Northeast Heavy Machinery Institute (China), and Ph.D. degree from the University of Leeds (UK), respectively. His research interests include computer vision, AI and robotics. He has published over 120 top quality international conference and journal papers. Dr. Wang is a senior member of IEEE.