International Journal of Computational Intelligence Systems (2021) 14:177
https://doi.org/10.1007/s44196-021-00031-y

RESEARCH ARTICLE

UAV Intelligent Coverage Navigation Based on DRL in Complex Geometrical Environments

Shuai Liu[1] · **Yuebin Bai**[1]

Received: 21 May 2020 / Accepted: 28 September 2021

Abstract
Unmanned aerial vehicle (UAV) is one of the preferred tools for coverage detection missions, because of its maneuverability and flexibility. It is challenging for the UAV to decide a track by itself in a complex geometrical environment. This paper presents a UAV intelligent navigation method based on deep reinforcement learning (DRL). We propose using geographic information systems (GIS) as the DRL training environment to overcome the inconsistency between the training environment and the test environment. We creatively save the flight path in the form of an image. The combination of the knowledge-based Monte Carlo tree search method and local search method can not only effectively avoid falling into local search, but also ensure learning the optimal search direction under the limitation of computing power. Experiments show that the trained UAV can find an excellent flight path by intelligent navigation, and able to make effective flight decisions in a complex geometrical environment.

Keywords UAV · Intelligent navigation · Coverage · Deep reinforcement learning · Monte Carlo tree search

1 Introduction

UAV has the advantages of small size, low cost, convenient use, minimal environmental requirements, and flexible aerial surveillance views over a wide area. Thus, it is employed in the field of surveillance, search, rescue, wildlife, border patrol, etc [1–4]. The UAV's sensor, such as the camera, is susceptible to interference in the complex geometry environment. The primary challenge in a coverage mission is planning the UAV path that effectively covers the given region [5]. These challenges include

Coverage sensing quality Many studies on UAV coverage problems assume that the given area has an ideal flat terrain [6]. In reality, most of the target terrain is rugged. The picture pixel quality obtained by the traditional method is inequality [5, 7]. Suppose UAV flies on a horizontal plane, as shown in Fig. 1. Photos' resolution taken by UAV camera sensors in red areas is higher, but photos' pixel taken in blue areas is low. UAV needs to adjust its altitude in real time to get good-quality terrain photos. All kinds of obstacles may occlude the cameras' view. To meet sensing resolution requirements, the UAV should be able to vary its flying height, which needs to optimize path planning in the 3D domain [8, 9].

Energy constraint and time constraint The UAV should travel through the waypoints with the time constraint [10, 11]. Searching for the optimal UAV path to meet the time and energy constraint is a non-deterministic polynomial-time hard (NP-hard) problem. Searching for a near-optimal path with comparable cost and much less search time is usually adopted in the existing UAV path planning algorithms. Finding an optimal flight path is factorial time complexity ($O(n!)$) [12]. n represents the number of alternative flight path points.

Intelligent real-time navigation UAV intelligent navigation means that UAV can make flight decisions itself based on the environment and coverage tasks. In recent years, deep reinforcement learning (DRL) methods are tried to solve intelligent path planning problems [13, 14]. By taking a depth image as the input and control commands as the output, the robot moves and tries to find a suitable path. However, most RGB-D cameras function in a limited range and cannot achieve satisfactory

✉ Yuebin Bai
yuebinbai@126.com

Shuai Liu
liushuai2017@buaa.edu.cn

[1] School of Computer Science and Engineering, Beihang University, College Road, Haidian District, No.37, Beijing 100191, China

Published online: 11 October 2021

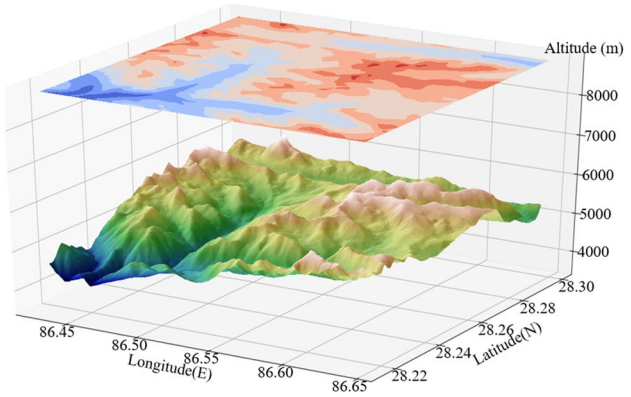


Fig. 1 UAV horizontally sample topography elevation mapping

navigation when used as the only sensor for long distances [15]. The DRL methods are not memorizing the maps at the testing stage but, instead, at the training stage [16]. In other words, finding a path in this way that requires flying repeatedly to learn a strategy in the real world is unrealistic.

This paper proposes a sensing quality-aware coverage and intelligent path planning solution for UAV monitoring of geometrically complex scenes with varying altitudes and occlusions. The goal is to provide overview images for a target area with satisfactory spatial and temporal resolutions under UAV energy limits. The main contributions of this paper are as follows: (1) we develop a DRL framework for UAV navigation in large-scale complex environments. We use terrain maps (for instance, Google terrain map, which is very close to the real UAV flying environment.) as models for DRL. The DRL framework makes it easier for us to find the global optimal flight path. After learning and training in the GIS, the UAV can make autonomous flight decisions. (2) The deep neural network (DNN) has a good understanding and analysis ability of images. We take the route UAV already passed by as the current agent observation. We creatively use the projection method to transform the 3D path into a 2D image. (3) We propose the terrain knowledge-based fast evolutionary MCTS (TK-MCTS) method. The TK-MCTS uses the terrain information to guide the UAV to search the unexplored area with a higher probability. (4) According to the terrain coverage task's characteristics, we combine local terrain exploration with global exploration. In this way, the UAV can avoid trapping at a local optimum. At the same time, it can also reduce the global estimation inaccuracy [17].

2 Relation Work

2.1 2D Coverage Path Planning

Early topographic coverage studies assume that target area terrains are ideal planes [18]. 2D path planning model and strategy are widely used, such as the spiral model, the spiral-like model, the Lawnmower model, the Zamboni model, the Dubins path model, and the modified Lawnmower/Zamboni path planning strategy [19]. However, most of the terrain is rugged. Obstacles affect the coverage of UAV camera sensors. If we only consider terrain coverage in 2D space, it is difficult to guarantee sensor quality.

2.2 Traditional 3D Coverage Path Planning

With the improvement of computing power, the latest research focuses on 3D space coverage path planning. Dai et al. [8] indicate that more images should be taken than in an ideal flat terrain case to achieve full coverage of all the spots inside a target area. During a coverage mission, [20] plan information-rich trajectories in continuous 3D space by building the terrain maps online to optimize initial solutions obtained. Scott et al. [21] propose an occlusion-aware UAV coverage technique by finding the best set of waypoints for taking pictures in a target area. The selected waypoints are then assigned to the UAV by solving a vehicle routing problem (VRP). Based on the matrix completion, [22] first select the dominator sampling points, and then select the virtual dominator sampling points. Finally, the optimal simulated annealing algorithm is used to plan the path of UAV based on the selected sampling points. In [23], the authors propose a coverage algorithm through the hexagonal tiling of a target region. But this method cannot be extended to 3D space. In [24], the authors develop a general preference-based multi-objective evolutionary algorithm to converge to preferred solutions, and preferences of a decision maker are elicited through reference point(s). However, this method cannot effectively deal with the environment where obstacles exist. Zhang et al. [25] exploit a newly defined individual cost matrix, which leads to an efficient multiple UAVs path planning algorithm. However, this algorithm is prone to failures in the relatively complex terrain environment. In [26], an integer linear programming formulation of the coverage path planning problem is shown to provide almost optimal strategies at a fraction of the computational cost of brute force methods. Bircher et al. [27] are capable of computing short inspection paths via an alternating two-step optimization algorithm. In this method, viewpoints are first found and then connected to form links. The path obtained by this method is often not the optimal path.

2.3 Intelligent 3D Path Planning

DNN has excellent learning ability and memory ability. Recent improvements in DRL have allowed solving problems in many 2D domains, such as Atari games [28, 29]. Wang et al. [30], based on the deep Q-Network framework, the raw depth image is taken as the only input to estimate the

Q values corresponding to all moving commands. Combining deep learning (DL) with reinforcement learning (RL), complex 3D terrain path planning has achieved better results than before. In recent years, DRL methods have been used in navigation [31, 32] and path planning [33–35] applications. In [36], DNN is used for real-time path planning. The study focuses on avoiding collisions with obstacles. Planning for autonomous unmanned ground vehicles, [37] proposes constrained shortest path search with graph convolutional neural networks (CNN).

3 DRL GIS Training Environment

DRL algorithms have demonstrated progress in learning to find a goal in challenging environments. The experiments in [16] show that the DRL algorithms are not memorizing the maps at the testing stage, but, rather, at the training stage. We propose using GIS as the DRL training environment to overcome the inconsistency between the training environment and the test environment [38]. UAV is trained in GIS and has the ability to make intelligent decisions so that the UAV can make accurate decisions in the real world. The steps to establish the GIS terrain training environment include terrain sampling, waypoint generation, and visibility analysis.

3.1 Complex GIS Terrain Sampling

We transform the terrain into a discrete set of geographic coordinate points $B = \{b_1, ..., b_n\}$, $b_i = (x_i, y_i, z_i)$. x_i denotes longitude, y_i denotes latitude, and z_i denotes altitude. Along with *longitude* and *latitude* on the ground plane, we sample the 2D coordinates on the ground plane with a step size of B_s. We can obtain the terrain elevation of each sampling point.

3.2 UAV Waypoint Generation and Visibility Analysis

It is possible to generate waypoints (photograph location) on the top of each terrain sampling point with different altitudes [8]. The collection of optional waypoint set is $R = \{p_1, ..., p_m\}$. Our goal is to find an ordered set of waypoints $W_k = [p_k^1, ..., p_k^j]$ from R, $W_k \subseteq R$. A covering flight path l is formed when the UAV flies along with this ordered set of geographical coordinate points, $l = p_k^1 \rightarrow \cdots \rightarrow p_k^j$. p_k^1 is the starting position. p_k^j is the termination position.

The problem of measuring the visible range of a point on GIS can be transformed into two points visual judgment problem. We use the interpolation visibility analysis method to judge whether two points are visible.

Algorithm 1 Interpolation visibility analysis method

Require: Initialize geographical points a, b.

1: Calculate a straight line l through a, b.
2: Insert $(lat_1, long_1, h_1)...(lat_i, long_i, h_t)$ evenly on l between a and b.
3: Get the elevation coordinates $(lat_1, long_1, e_1)$ ... $(lat_n, long_n, e_n)$.
4: **if** $(\forall t_i, h_{t_i} < e_{t_i}) || (\forall t_i, h_{t_i} > e_{t_i})$ **then**
5: $visibility = true$.
6: **else**
7: $visibility = false$.

The detail of the interpolation visibility analysis method shows as Algorithm 1. GIS can calculate the elevation $(lat_1, long_1, e_1)...(lat_n, long_n, e_n)$ according to latitude and longitude. e_{t_i} represents elevation data obtained by GIS computation. If $\exists\, t_i, t_j$, $h_{t_i} \geq e_{t_i}$ and $h_{t_j} \leq e_{t_j}$; a, b invisible. When the interpolation density reaches a specific number, we can effectively judge whether a and b are visible in the line of sight. We can see in Fig. 2. In the horizontal direction, interpolation is performed every 5 m along the line segment l. From A to B, $h_{A\rightarrow B} > e_{A\rightarrow B}$. Therefore, we can see B from A. From A to C, $h_{A\rightarrow C'} > e_{A\rightarrow C'}$, $h_{C'\rightarrow C''} < e_{C'\rightarrow C''}$, $h_{C''\rightarrow C} > e_{C''\rightarrow C}$. Therefore, we cannot see C from A.

4 UAV Intelligent Coverage Navigation Based on DRL

We present a UAV intelligent navigation method based on DRL. A UAV is an agent in the DRL structure. We use a four-rotor UAV to perform the coverage mission. The UAV continuously improves its strategy to maximize the covering tasks cumulative

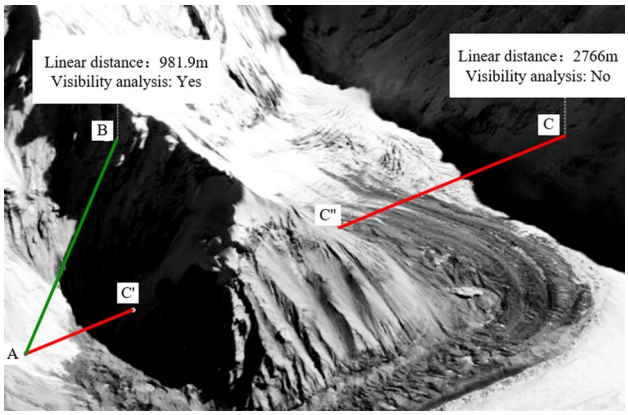


Fig. 2 Interpolation visibility analysis

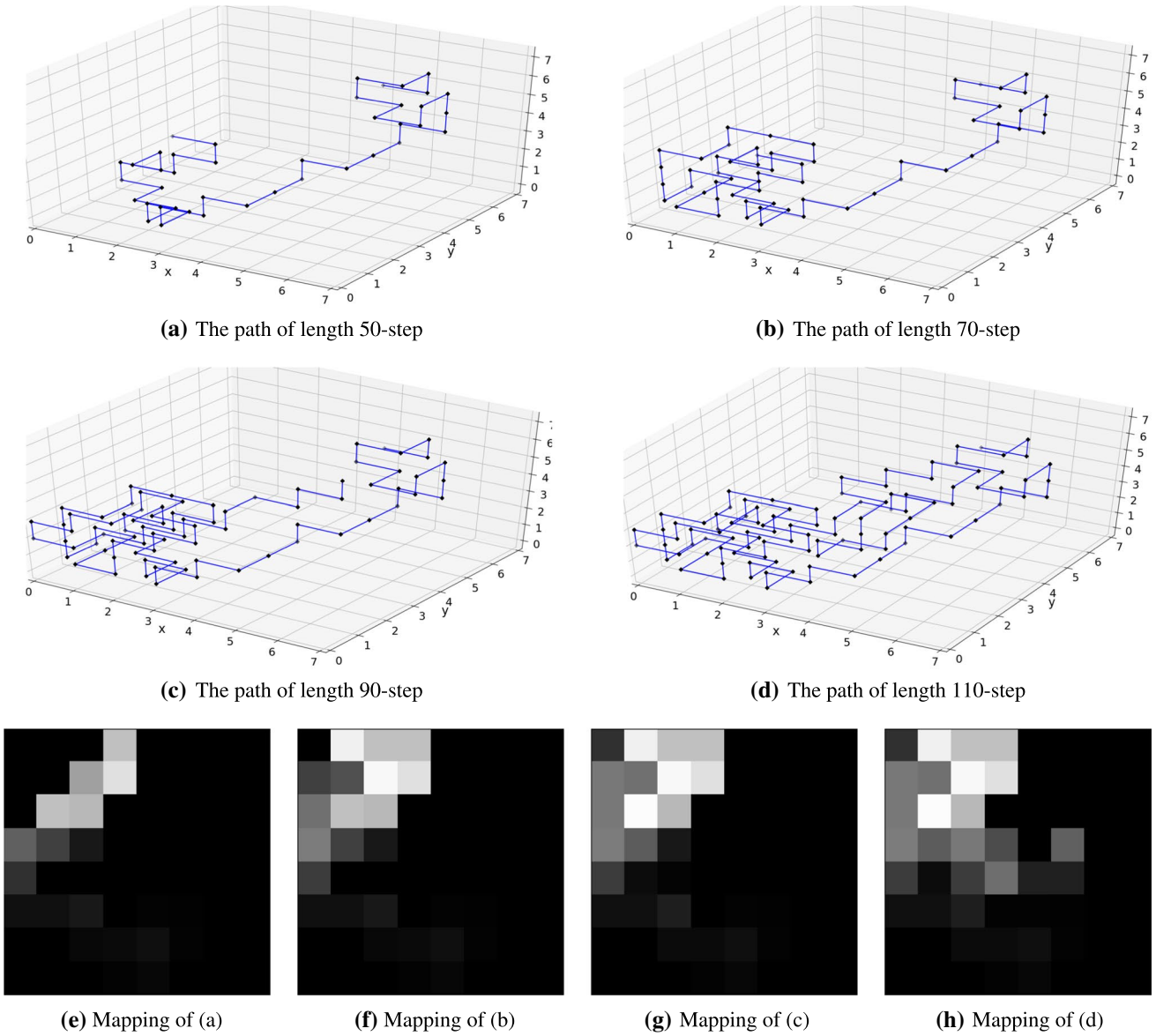
rewards by collecting samples (states, actions, and rewards) from its interactions with complex terrain environments. We use a neural network to simulate the $E(l)$ function. Input the state into the neural network and output the UAV flight action value. At each discrete time step, the UAV selects an action from the action space (up, down, latitude+, latitude−, longitude+ and longitude−). We creatively use the UAV flown path as the current UAV state, which is the input neural network state.

4.1 UAV Coverage Navigation State

We creatively save the flight path in the form of an image and use the image as the input of DNN. The DNN has an excellent ability to analyze and understand data in image form. If the UAV path information is converted into the image form, the DNN will be better for feature extraction and classification. We propose the *elevation compression method* to convert 3D GPS path data into a 2D image.

To illustrate the algorithm, we take Fig. Convert3DGPSpathdatainto2Dimage as an example. Suppose Fig. 3a is the initial state. With the elevation compression method, we get a two-dimensional map of the path, Fig. 3e. Starting from Fig. 3a, after 20 actions, the status of the UAV is shown in Fig. 3b. Figure 3(f) is the two-dimensional map of Fig. 3b. Figure 3c, d is the UAV path at intervals of 20 actions. Figure 3g, h is the mapping of 3c, d. We see that as the path increases, the total of gray rectangle decreases. The DNN can effectively recognize the change of image grayscale and distinguish the path.

The *elevation compression method* is shown in Algorithm 2. The step 2 stores waypoints in the 3D matrix M. We convert



(a) The path of length 50-step

(b) The path of length 70-step

(c) The path of length 90-step

(d) The path of length 110-step

(e) Mapping of (a)

(f) Mapping of (b)

(g) Mapping of (c)

(h) Mapping of (d)

Fig. 3 Convert 3D GPS path data into 2D image

the z-axis data on the same xy axis into a binary sequence. Each binary data are stored in decimal form in the matrix V. Convert matrix V to gray image. For example, we take the z column where the $X-axis$ is i and the $Y-axis$ is j. Assume that $M[i][j] = [0, 1, 0, 1, 1]$. Let us view [0, 1, 0, 1, 1] as binary number 01011. Converts the binary number 01011 to the decimal number, 11. We store 11 in a two-dimensional matrix $V[i][j] = 11$. Calculate all z columns using Algorithm 2 step 3–10. A two-dimensional matrix V is obtained to store image data.

Algorithm 2 Elevation Compression Method

Require: Store the waypoints by shape in a 3D matrix M; Initialize the matrix M as 0; Initialize the GPS path as p; Initialize the 2D-matrix V as 0;

1: **for** $i = 0$ to $len(p) - 1$ **do**
2: $M[p[i]] = 1$.
3: **for** $x := 0$ to $M - x - axis$ **do**
4: $z = NULL$.
5: **for** $y := 0$ to $M - y - axis$ **do**
6: Convert M[x][y] to string.
7: $z = z + M[x][y]$.
8: Convert z from string to binary d.
9: Convert z from binary to decimal d.
10: $V[x][y] = d$.
11: Map V to grayscale image.

4.2 UAV Coverage Navigation Reward Based on TK-MCTS

Flying over previously uncovered areas as much as possible can yield more simulated flight results in a limited time. Based on the path coverage performance comparison, we determine the action reward value. The search space for UAV waypoints is vast, and we need to optimize our search. Depending on the terrain and the coverage state, drones flying in the no coverage direction can more effectively cover the entire mission. We propose the *terrain knowledge – based Monte Carlo tree search* (TK-MCTS) algorithm. The combination of the TK-MCTS method and local search method can not only effectively avoid falling into local search but also ensure learning the optimal search direction under the limitation of computing power. The TK-MCTS is much better than traditional MCTS in exploitation. The TK-MCTS algorithm uses map knowledge to guide the UAV to fly as far as possible to the previously uncovered area. Different from the traditional MCTS algorithm, *position guide points* (PGP) are used to guide the UAV flight direction. PGP is a random subset of the optional flight points set R. $PGP = \{g_0, ..., g_l\}, PGP \subseteq R$. The visible terrain area of g_i is t_i. $PGPT = \{t_0, ..., t_l\}$. We calculate the probability of flying in different directions. Use the formula 1 to calculate the scores in each direction:

$$score = (n_i + 1) * rand(0, 1) * \rho \tag{1}$$

where n_i is the number times of i has been simulated. $\rho \in (0, 1]$. If the vector $s \to i$ is going in the same direction as $s \to g_i$ on x, y or z axis, $\rho \in (0, 1)$. Otherwise, $\rho = 1$. Choose the minimum score in different directions as the simulation node.

Algorithm 3 TK-MCTS

Require: Initialization waypoint set R; Initialization calculators $times$; Initialization time threshold T; Initialization the terrain that has been covered is C_k; Start at the current simulation point S; Initialization simulation counter p_{step};

1: **while** $times < T$ **do**
2: Initialization num.
3: Randomly select l nodes from R, $PGP = \{g_1, ..., g_l\}$.
4: Calculate g_i's coverage area, c_i.
5: **if** $c_j \subseteq C_k$ **then**
6: Remove g_j from the set of PGP.
7: **if** $PGP! = NULL$ **then**
8: Randomly selection guiding point g_m from PGP, as guiding point.
9: Use Eq.1 to calculate the scores in each direction.
10: Choose the minimum score.
11: $num = num + 1$.
12: **if** g_m or $num > p_{step}$ **then**
13: Go to $step2$.
14: **else**
15: Go to $step4$.
16: Evaluation, get reward.

The specific algorithm is Algorithm 3. According to the results of flight simulation in different directions, we set the reward value of the direction with the best simulation results to be 1, the other direction is 0.

4.3 UAV Coverage Navigation DRL Implementation

The core of reinforcement learning is the discovery of the optimal action-value function $Q^*(s, a)$ by maximizing the expected return, starting from state s taking action a:

$$Q^*(s,a) = \max_\pi E[R_t|s_t = s, a_t = a, \pi] \tag{2}$$

The total future reward until the termination is R_t. The π represents DRL policy. With future reward discounted factor is γ, the total future estimated reward is

$$R_t = \sum_{t=0}^{T} \gamma^t r \tag{3}$$

The essential assumption in DRL is the Bellman equation, which transfers the target to maximize the value of $r + \gamma Q^*(s', a')$ as

$$Q^*(s,a) = E_{s'\sim e}[r + \gamma \max_{a'} Q^*(s', a')|s, a] \tag{4}$$

where s' is the next state. DRL estimates the action-value equation by convolutional neural networks with weights θ, so that $Q(s, a, \theta) \approx Q^*(s, a)$. Set the training batch size to be b, the loss function is

$$L(\theta_i) = \frac{1}{b} \sum_{i=1}^{b} (y_k - Q(s_k, a_k; \theta_i))^2 \tag{5}$$

where y_k is the target output evaluation network. It is calculated by future expectation estimated. If the sampled transition is not a fly way sample, the evaluation for this (s_k, a_k) pair is set as termination reward r_{ter}.

In this paper, the shape of the input terrain image is $10 \times 10 \times 3$. The structure is depicted in Fig. 4. We use two 3D convolutional layers to extract the path image features. We apply a $10 \times 10 \times 4$ (the number of channels = 4) and $10 \times 10 \times 64$ (the number of channels = 64) convolution filter on images. The role of the convolution layer is local perception. That is, each feature in the picture is first perceived locally, and then the local comprehensive operation is carried out at a higher level, to obtain global information. Each convolution layers calculation results are input to the pooling layer. The main role of the pooling layer is to reduce the feature dimension. In Fig. 4, long short-term memory (LSTM) cannot only solve that RNN cannot deal with long-distance dependence but also solve that gradient explosion or gradient disappearance. We use an additional two fully connected layers for exploration policy learning. Finally, the neural network outputs the UAV action values. Each *Conv* or *Fullyconnected layer* is followed by a *Rectified Linear Unit* (ReLU) activation function layer to increase the non-linearity. The number under each layer is the output data channels of the cubes.

Algorithm 4 shows the workflow of our revised DRL process. We set the number of iterative rounds *episode* to M. As shown in steps 4–16, perform an action a_t in state s_t, get the next state t_{t+1} and the current reward r_t. Adds the four tuples (s_t, a_t, r_t, s_{t+1}) to the experience pool D. Take m random samples (s_i, a_i, r_i, s_{i+1}) from the empirical pool D, where $i = 1, 2, 3...., m$, calculate the target value y_i. Step 20 uses the mean square error loss function $\frac{1}{m}\sum_{i=1}^{m}\left(y_i - Q\left(s_i, a_i, \omega\right)\right)^2$ to update the Q network parameters. We use the memory replay method and the $\varepsilon - greedy$ training strategy to control the dynamic distribution of training samples. At the beginning of every repeated exploration loop, the UAV is set to a random start point. It extends the randomization of the UAV locations from the whole simulation world and keeps the diversity of the data distribution saved in memory replay for training.

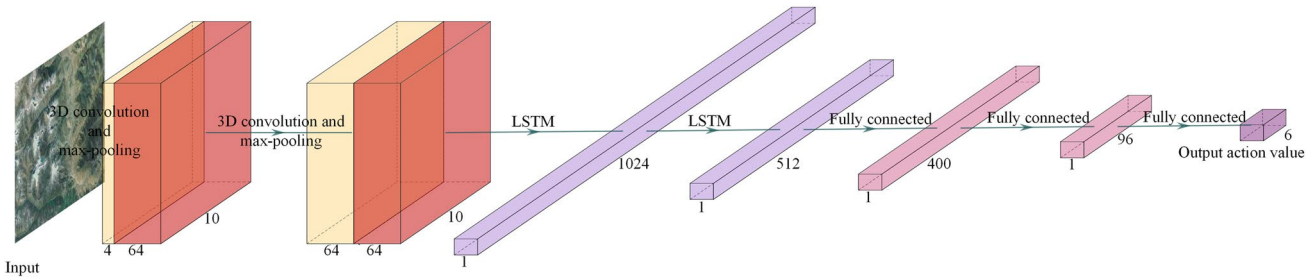


Fig. 4 UAV coverage navigation DRL

Algorithm 4 Deep Reinforcement Learning Algorithm

Require: Initialize the weights of evaluation networks as w; Initialize the bias of evaluation networks as b; Initialize the memory D to store experience replay; Set the visual distance threshold l; Number of batch samples m;

for $episode = 1$ to M **do**
  Reset the starting position of the UAV.
  while True **do**
    if Random(0,1) $> \varepsilon$ **then**
      Select a random action a_t.
    else
      $a_t = \arg\max_a Q(s_t, a; w)$.
    if $a_t == Invalidaction$ **then**
      $r_t = r_{ter}$.
      $s_{t+1} = None$.
      break.
    else
      Execute a_t.
      Get new fly 3D path.
      Convert 3D path data into 2D image s_{t+1} by Algorithm 2.
      Get reward r_t by TK-MCTS.
    Store the transition (s_t, a_t, r_t, s_{t+1}) in D.
    Select a batch m randomly from D.
    Calculates the current target Q, y_i. $y_i = r_i + \gamma \max_{a'} Q(s_{i+1}, a'; w)$.
    Update w through a gradient descent procedure on the batch of $(y_k - Q(\varphi_k, a_k; w))^2$.

The space complexity of Algorithm 4 is closely related to the neural network layers. Therefore, we need to analyze the space complexity layer by layer. Assume that the convolution kernel size is $H \times W$, the input channel is I and the out channel is O. The total number of filters in the convolutional layer is $H \times W \times I$. Each filter will be mapped to O new channels. Plus a bias for each filter's calculation. Therefore, the total number of parameters is $(H \times W \times I + 1) \times O$. Pooling Layer is a fixed operation with no weighting factor. A fully connected layer is an n_{input},m_{output} dimensional input and output with $(n+1) \times m$ parameters. The LSTM will maintain a total of 4 sets of parameters, corresponding to input gates, output gates, oblivion gates and candidate states. Therefore, the total number of parameters is $4 \times (n_{\text{hidden}} m_{\text{input}} + n^2_{\text{hidden}} + n_{\text{hidden}})$. n_{hidden} is the hidden size and m_{input} is the input size.

The time complexity of training the neural network is $O(E \times D/B \times T)$, where E is episode, D is the memory size, B is the batch size, and T is the time complexity of a single iter. T can continue to be decomposed into $O(T) \approx O\left(L \times n_{\text{layer}}\right)$, L is the average time complexity of each layer and n_{layer} is the number of layers. The L can be further decomposed into $O(L) \approx O\left(M_{\text{input}} N_{\text{output}} H_{map} W_{map} K\right)$, where the time to compute the convolutional layers is assumed to be the average time per layer. M_{input} and N_{output} are the number of input and output channels, respectively, K is the size of the convolutional kernel, and H_{map} and W_{map} are the spatial dimensions of the output feature map, respectively.

5 Experiment and Result

5.1 Terrain Data Sampling and Visibility Analysis

On the Cesium platform [39], we chose the geographic location N86.8° − 87.0°, E27.5° − 28.02° as the area for the terrain coverage task. We sample the 2D coordinates along $longitude - axes$ and $latitude - axes$ on the ground plane with a step size of 0.628 km. We obtain each terrain sampling point elevation corresponding by Cesium. Each optional waypoint does visibility analysis. On the Cesium, we use Algorithm 1 (interpolation visibility analysis method) to obtain visibility set v_i of each optional waypoint p_i. We randomly pick 10 waypoints, as shown in Fig. 5. The average visual distance of optional waypoints to the whole terrain sampling points is more than 2000 m. We set the effective visibility distance threshold value as 1500 m. Each optional waypoint can only see a few numbers of terrain sample points.

5.2 The TK-MCTS Performance

We use different algorithms in the same flight terrain space to verify the advantages of the TK-MCTS algorithm. We compare the total number of searches and the number of valid searches with the exhaustive method and traditional MCTS algorithm [40, 41]. As we can see from Fig. 6(a), the Exhaustion method performs a broader range of searches in the same time frame. However, the successful full coverage terrain path searches account for only 0.0388% of the total searches. The traditional MCTS search has the smallest search range in the same period. The traditional MCTS method successfully searches the full coverage terrain path more times than the Exhaustion method in the same time frame. We see that although the TK-MCTS method does not have the most searches in the same time frame, it is the most efficient way to find the full coverage path. The number of successful full coverage terrain path searches accounts for 47.29% of the total searches.

5.3 A Combination of Local and Global Search

The spatial complexity of the coverage path planning search is approximately O(n!). Although the TK-MCTS approach significantly improves the effectiveness of the simulation, it is not possible to conduct extensive searches in nearly infinite search spaces. On a mediocre computer, the average number of valid searches is 326.89 when the search time is 10 s. Finding the full coverage path is 153.12 times. The limited number of simulation samples will bring high errors to evaluate the effectiveness of the current motion direction.

Starting from the current waypoint, we calculate the time and coverage to view all paths within n steps using an exhaustive method.

Figure 7a shows the impact of the different steps on coverage. Figure 7a shows that merely increasing the number of exhaustive searches does not improve the simulation. The $n-step$ exhaustive method can only determine whether it is a locally optimal solution. We cannot simulate the optimal global solution by increasing n. As $n-step$ increases the exhaustion time index increases as well.

For the above problems, we design a method combining local search and global search to determine the simulation reward value. Through the TK-MCTS method, we can roughly estimate the optimal global direction. By $n-step$ exhaustion, we determine the direction of the optimal local solution. The simulated reward design is shown in Table 1.

Negative no waypoint can fly to in the current state;

TK − MCTS positive only through TK-MCTS gets a positive reward;

Exhaustive positive only through Exhaustive gets a positive reward;

Exhaustive and TK − MCTS positive through Exhaustive and the TK-MCTS Positive get a positive reward;

Other no simulation results are obtained. Through Exhaustive and the TK-MCTS does not obtain a valid path coverage method

5.4 DRL Intelligent Path Planning

We initialize each layer weights from a normal distribution (mean 0 and variance 0.3), and the biases are set as 0.1. The training parameters are shown in Table 2. All models are trained and tested with TensorFlow on a single NVIDIA GeForce GTX 1050ti. The impact of batch size on training is as follows: if it is too small, it will lead to great gradient change, loss oscillation, and network difficult convergence; if it is too large, the gradient is very accurate, loss oscillation is small, and it is easy to fall into local optimum. Plenty of practice shows that the best training results are always achieved when the batch size is between 2 and 32. The effect of learning rate on training is as follows: too small means it takes longer to converge; too large may not converge or the loss may explode. The initial learning rate is usually set at 0.01–0.001. Replay memory breaks the correlation between data by storage-sampling. There is no particular requirement for the replay memory size. $discountfactor = 0.9$, which balances the current and future rewards.

We have analyzed the validity of DRL intelligent path planning through experiments. We evaluate the proposed work in terms of coverage quality and path planning quality in Cesium. With varying step sizes, we compare the coverage percentage of our proposed algorithm with the exhaustion algorithm and MCTS algorithm. The computation results help to determine the appropriate step size to ensure terrain full coverage with the generated waypoints.

5.4.1 Varying Step Size Coverage Performance

The distance between the terrain sample point and waypoint directly affects the resolution of the terrain image obtained by the UAV. We set the visual range threshold from a waypoint to the terrain sampling point as 1000 m. The total UAV step number can approximately represent the UAV carry out task time. The coverage results for the target area are shown in Fig. 8. The vertical axe presents the percentage of areas that can be covered with the given image resolution requirement. The step size in the horizontal axes represents the number of blocks. We step through the environment in each iteration while searching for waypoints, and step size values ranging from 1000 to 1900 are tested. In Fig. 8, the UAV starts from different positions to verify the effectiveness of the DRL method in training UAV intelligent navigation. DRL algorithm, Exhaustion algorithm and MCTS algorithm are adopted to calculate the average coverage of different steps. The search time of Exhaustion and MCTS is 300 s. The UAV performs multiple searches of a given step length within the 300 s.

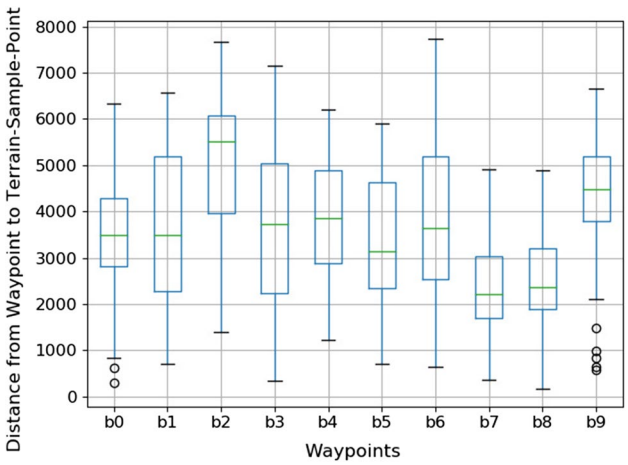
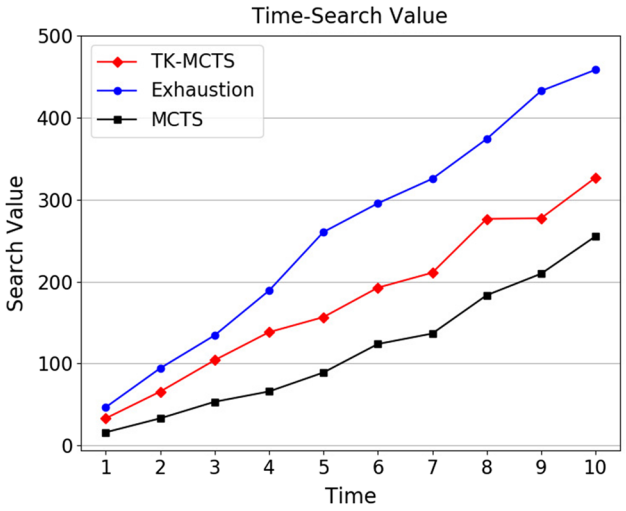
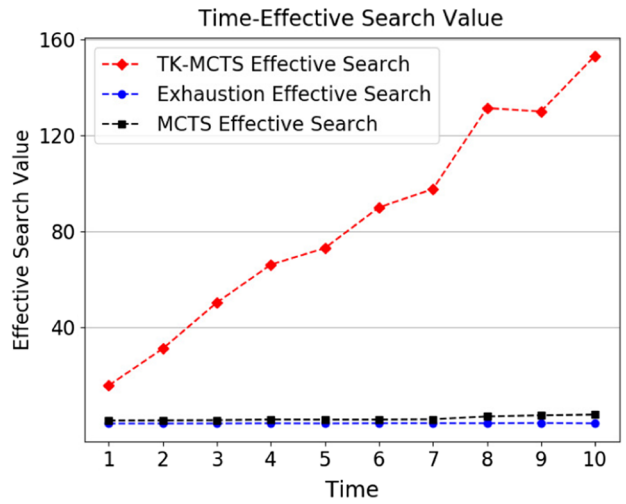


Fig. 5 An example of optional waypoint visibility analysis

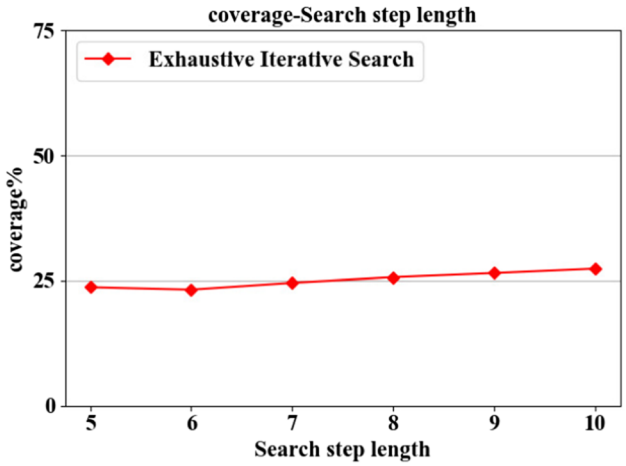


(a) Total number of searches

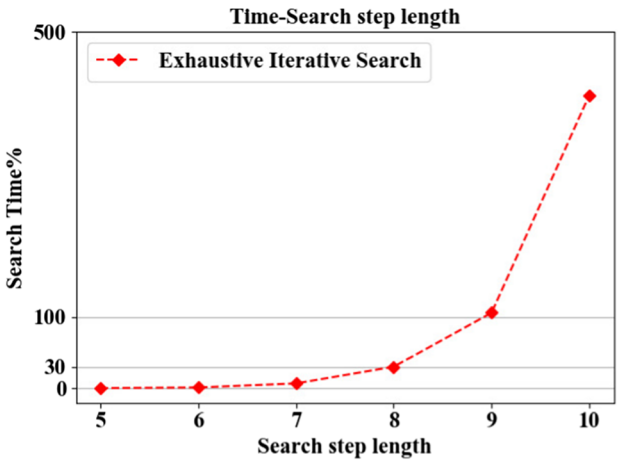


(b) The number of effective searches

Fig. 6 Performance comparison of TK-MCTS, MCTS, and Exhaustion



(a) Coverage-Search step length



(b) Time-Search step length

Fig. 7 Exhaustive iterative search time and coverage

We can see that when step = 1700, the terrain coverage obtained by the DRL method can reach 100%. From 1000-step to 1900-step, DRL can always achieve better coverage than the Exhaustion algorithm and MCTS algorithm. Moreover, from 1000-step to 1900-step, the MCTS algorithm can obtain a better coverage effect than the Exhaustion algorithm. During the initial training process, DRL selects the action calculated by the DNN with a 50% probability for each step, and randomly selects action with a 50% probability. The method allows a full exploration of UAV navigation in various situations. Through experiments, we prove that DRL could learn excellent results with the TK-MCTS simulation.

Table 1 UAV state and reward design

State	Reward
Negative	−20.0
TK-MCTS positive	0.2
Exhaustive positive	0.1
Exhaustive and TK-MCTS Positive	0.4
Other	0

5.4.2 Terrain Average Coverage

In coverage missions, we are concerned about the UAV effective coverage. All area's effective coverage requires high resolution of every terrain data collected by UAV. We have heat maps to visually describe the performance of the DRL algorithm in improving UAV terrain coverage quality. Figure 9a–c is the heat maps of the DRL algorithm, MCTS

Fig. 9 Performance comparison in terrain effective coverage resolution distance. x0=27.9596°N, x1=27.9660°N, x2=27.9723°N, x3=27.9787°N, x4=27.9851°N, x5=27.9914°N, x6=27.9978°N, x7=28.0042°N, x8=28.0105°N, x9=28.0169°N, y0=86.8965°E, y1=86.9029°E, y2=86.9092°E, y3=86.9156°E, y4=86.9220°E, y5=86.9283°E, y6=86.9347°E, y7=86.9410°E, y8=86.9474°E, y9=86.9538°E

algorithm and Exhaustive method, respectively. We use shades of color to indicate distance. Darker the color, the larger the distance is.

The horizontal axis represents the latitude, and the vertical axis represents longitude. First, we see that the DRL heat map Fig. 9a is generally lighter in color than the other two, and the MCTS heat map Fig. 9b is lighter in color than Fig. 9c. We can see that the maximum distance from the waypoint to the terrain sampling point can be less than 200 m with the coverage effect of navigation by the DRL algorithm. With the coverage effect of navigation by the MCTS algorithm, the maximum distance from the waypoint to the terrain sampling point is 600–700 m. The maximum distance from the waypoint to the terrain sampling point will be larger when it uses the Exhaustive method navigation. Second, we see that the coverage achieved by the DRL method is very even for each piece of terrain. Since some parts are light and some parts are dark, the coverage effect of each piece of terrain by other methods is uneven. Especially the

Table 2 The training parameters

Parameter	Value
Batch size	32
Replay memory size	1000
Discount factor γ	0.9
Learning rate	0.01
Gradient momentum	0.9

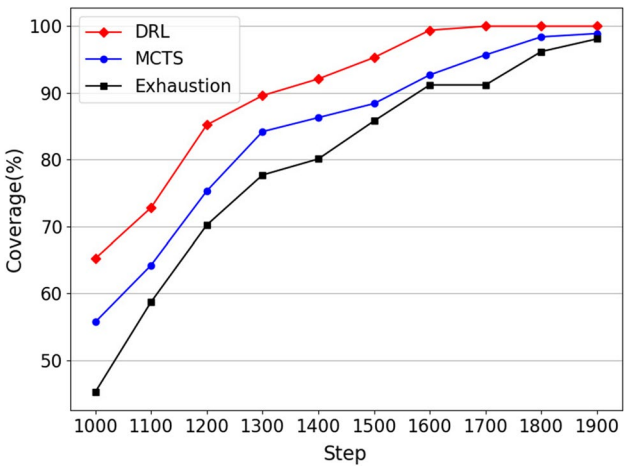
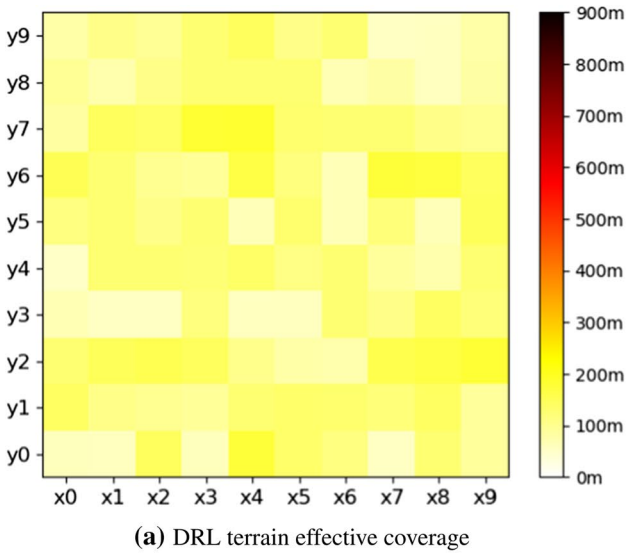
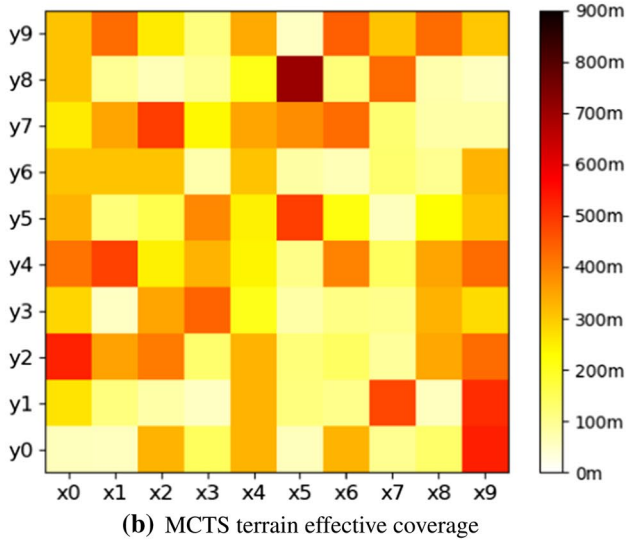


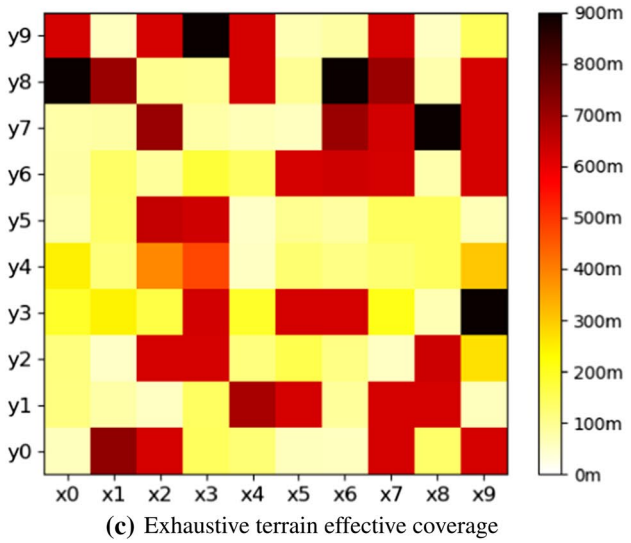
Fig. 8 Coverage performance with varying step sizes



(a) DRL terrain effective coverage



(b) MCTS terrain effective coverage



(c) Exhaustive terrain effective coverage

coverage resolution by Exhaustive method is quite different. In the training stage, DRL uses the TK-MCTS method to calculate the optimal simulation results of the next step. DRL can fully consider local best and global best.

6 Conclusion

In this work, we develop a DRL framework for UAV navigation in large-scale complex environments. We can use GIS with accurate and rich data as the training environment by converting GPS path data into image data. This method can effectively overcome the huge errors caused by the inconsistency between the training environment and using environment. The combination of the TK-MCTS search method and local search method cannot only effectively avoid falling into local search, but also ensure to learn the optimal search direction under the limitation of effective computing force. The results show that CNNs can learn important features from GPS path information of the 3D environment, and learn a navigation policy from the TK-MCTS simulation. The integration of visual navigation and GPS navigation is the research direction to improve navigation accuracy. For the unknown terrain environment, the combination of online 3D terrain generation and GPS navigation is also an important issue to be studied in the future.

Acknowledgements This work is supported by the National Natural Science Foundation of China (NSFC) [No.61502019, No.61572062 and No.61732002]; National Key Research and Development Program of China [No.2016YFB1000503]. We thank everyone for their support.

Declarations

Conflict of Interest The authors declare that they have no competing interests.

References

1. Mozaffari, M., Saad, W., Bennis, M., Nam, Y.-H., Debbah, M.: A tutorial on UAVs for wireless networks: applications, challenges, and open problems. IEEE Commun. Surveys Tutorials **21**(3), 2334–2360 (2019). https://doi.org/10.1109/COMST.2019.2902862
2. Liu, Y., Dai, H.-N., Wang, Q., Shukla, M.K., Imran, M.: Unmanned aerial vehicle for internet of everything: opportunities and challenges. Comput. Commun. **155**, 66–83 (2020). https://doi.org/10.1016/j.comcom.2020.03.017
3. Altan, A., Hacioglu, R.: Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking under external disturbances. Mech. Syst. Signal Proc. (2019). https://doi.org/10.1016/j.ymssp.2019.106548
4. Mohamed, N., Al-Jaroodi, J., Jawhar, I., Idries, A., Mohammed, F.: Unmanned aerial vehicles applications in future smart cities. Technol. Forecasting Social Change (2020). https://doi.org/10.1016/j.techfore.2018.05.004
5. Huang, H., Savkin, A.V.: An algorithm of reactive collision free 3-D deployment of networked unmanned aerial vehicles for surveillance and monitoring. IEEE Trans. Ind. Inf. **16**(1), 132–140 (2020). https://doi.org/10.1109/TII.2019.2913683
6. Samir, M., Sharafeddine, S., Assi, C.M., Nguyen, T.M., Ghrayeb, A.: UAV trajectory planning for data collection from time-constrained IoT devices. IEEE Trans. Wireless Commun. **19**(1), 34–46 (2020). https://doi.org/10.1109/TWC.2019.2940447
7. Pham, H.X., La, H.M., Feil-Seifer, D., Deans, M.C.: A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking. IEEE Trans. Syst. Man Cybernetics-Syst. **50**(4), 1537–1548 (2020). https://doi.org/10.1109/TSMC.2018.2815988
8. Dai, R., Fotedar, S., Radmanesh, M., Kumar, M., Quality-aware, U.A.V.: coverage and path planning in geometrically complex environments. Ad Hoc Netw. **73**, 95–105 (2018). https://doi.org/10.1016/j.adhoc.2018.02.008
9. Lin, Y., Saripalli, S.: Sampling-based path planning for UAV collision avoidance. IEEE Trans. Intell. Transp. Syst. **18**(11), 3179–3192 (2017). https://doi.org/10.1109/TITS.2017.2673778
10. Hua, M., Wang, Y., Zhang, Z., Li, C., Huang, Y., Yang, L.: Power-efficient communication in UAV-aided wireless sensor networks. IEEE Commun. Lett. **22**(6), 1264–1267 (2018). https://doi.org/10.1109/LCOMM.2018.2822700
11. Zeng, Y., Zhang, R.: Energy-efficient UAV communication with trajectory optimization. IEEE Trans. Wireless Commun. **16**(6), 3747–3760 (2017). https://doi.org/10.1109/TWC.2017.2688328
12. Caillouet, C., Giroire, F., Razafindralambo, T.: Optimization of mobile sensor coverage with UAVs, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, IEEE. https://doi.org/10.1109/infcomw.2018.8406980, 2018
13. Lu, Y., Xue, Z., Xia, G.-S., Zhang, L.: A survey on vision-based UAV navigation. Geo-spatial Inf. Sci. **21**(1), 21–32 (2018). https://doi.org/10.1080/10095020.2017.1420509
14. Wang, C., Wang, J., Shen, Y., Zhang, X.: Autonomous navigation of UAVs in large-scale complex environments: a deep reinforcement learning approach. IEEE Trans. Vehicular Technol. **68**(3), 2124–2136 (2019). https://doi.org/10.1109/TVT.2018.2890773
15. Tai, L., Paolo, G., Liu, M.: Virtual-to-real deep reinforcement learning: continuous control of mobile robots for Mapless Navigation, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), ISBN 978-1-5386-2682-5, ISSN 2153-0858, 31–36, 2017ISBN
16. Banerjee, S., Dhiman, V., Griffin, B., Corso, J. J.: Do deep reinforcement learning algorithms really learn to navigate?, https://openreview.net/forum?id=BkiIkBJ0b, 2018
17. Rout, M., Roy, R.: Self-deployment of mobile sensors to achieve target coverage in the presence of obstacles. IEEE Sensors J. **16**(14), 5837–5842 (2016). https://doi.org/10.1109/JSEN.2016.2571064
18. Zorbas, D., Pugliese, L.D.P., Razafindralambo, T., Guerriero, F.: Optimal drone placement and cost-efficient target coverage. J.

Netw. Comput. Appl. **75**, 16–31 (2016). https://doi.org/10.1016/j.jnca.2016.08.009
19. Cabreira, T.M., Di Franco, C., Ferreira, P.R., Jr., Buttazzo, G.C.: Energy-aware spiral coverage path planning for UAV photogrammetric applications. IEEE Robotics Automation Lett. **3**(4), 3662–3668 (2018). https://doi.org/10.1109/LRA.2018.2854967
20. Popovic, M., Vidal-Calleja, T., Hitz, G., Chung, J.J., Sa, I., Siegwart, R., Nieto, J.: An informative path planning framework for UAV-based terrain monitoring. Autonomous Robots **44**(6), 889–911 (2020). https://doi.org/10.1007/s10514-020-09903-2
21. Scott, K., Dai, R., Kumar, M.: Occlusion-aware coverage for efficient visual sensing in unmanned aerial vehicle networks, in: 2016 IEEE Global Communications Conference (GLOBECOM), https://doi.org/10.1109/GLOCOM.2016.7842033, 2016
22. Liu, X., Liu, Y., Zhang, N., Wu, W., Liu, A.: Optimizing trajectory of unmanned aerial vehicles for efficient data acquisition: a matrix completion approach. IEEE Internet Things J. **6**(2), 1829–1840 (2019). https://doi.org/10.1109/JIOT.2019.2894257
23. Kadioglu, E., Urtis, C., Papanikolopoulos, N.: UAV coverage using hexagonal tessellation, in: 2019 27th Mediterranean Conference on Control and Automation (MED), ISBN 978-1-7281-2803-0, ISSN 2325-369X, 37–42, 2019
24. Dasdemir, E., Koksalan, M., Ozturk, D.T.: A flexible reference point-based multi-objective evolutionary algorithm: an application to the UAV route planning problem. Comput. Oper. Res. (2020). https://doi.org/10.1016/j.cor.2019.104811
25. Zhang, B., Liu, W., Mao, Z., Liu, J., Shen, L.: Cooperative and geometric learning algorithm (CGLA) for path planning of UAVs with limited information. Automatica **50**(3), 809–820 (2014). https://doi.org/10.1016/j.automatica.2013.12.035
26. Li, M., Richards, A., M. S.: Reliability-Aware Multi-UAV Coverage Path Planning Using Integer Linear Programming, in: UKRAS20 Conference: "Robots into the real world" Proceedings, EPSRC UK-RAS Network, https://doi.org/10.31256/cy5ej9k, 2020
27. Bircher, A., Kamel, M., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., Siegwart, R.: Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots. Autonomous Robots **40**(6), 1059–1078 (2016). https://doi.org/10.1007/s10514-015-9517-1
28. Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Hasan, M., Essen, B.C.V., Awwal, A.A.S., Asari, V.K.: A state-of-the-art survey on deep learning theory and architectures. Electronics **8**(3), 292 (2019). https://doi.org/10.3390/electronics8030292
29. Chandrasekaran, G., Periyasamy, S., Rajamanickam, K.P.: Minimization of test time in system on chip using artificial intelligence-based test scheduling techniques. Neural Comput. Appl. **32**(9), 5303–5312 (2020). https://doi.org/10.1007/s00521-019-04039-6
30. Wang, J., Gou, L., Shen, H.-W., Yang, H.: DQNViz: a visual analytics approach to understand deep Q-networks. IEEE Trans. Vis. Comput. Graphics **25**(1), 288–298 (2019). https://doi.org/10.1109/TVCG.2018.2864504
31. Bouhamed, O., Ghazzai, H., Besbes, H., Massoud, Y., Autonomous, U.A.V., Navigation: A DDPG-Based Deep Reinforcement Learning Approach, in, : IEEE International Symposium on Circuits and Systems (ISCAS). IEEE **2020**, (2020). https://doi.org/10.1109/iscas45731.2020.9181245
32. Patle, B.K., Babu, G.L., Pandey, A., Parhi, D.R.K., Jagadeesh, A.: A review: on path planning strategies for navigation of mobile robot. Defence Technol. **15**(4), 582–606 (2019). https://doi.org/10.1016/j.dt.2019.04.011
33. Bayerlein, H., Theile, M., Caccamo, M., Gesbert, D.: UAV Path Planning for Wireless Data Harvesting: a Deep Reinforcement Learning Approach, in: GLOBECOM 2020—2020 IEEE Global Communications Conference, IEEE, https://doi.org/10.1109/globecom42002.2020.9322234, 2020
34. Aggarwal, S., Kumar, N.: Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. Comput. Commun. **149**, 270–299 (2020). https://doi.org/10.1016/j.comcom.2019.10.014
35. Chandrasekaran, G., Karthikeyan, P.R., Kumar, N.S., Kumarasamy, V.: Test scheduling of System-on-Chip using Dragonfly and Ant Lion optimization algorithms. Neural Comput. Appl. **40**(3), 4905–4917 (2021). https://doi.org/10.3233/JIFS-201691
36. Carrio, A., Sampedro, C., Rodriguez-Ramos, A., Campoy, P.: A review of deep learning methods and applications for unmanned aerial vehicles. J. Sensors (2017). https://doi.org/10.1155/2017/3296874
37. Osanlou, K., Bursuc, A., Guettier, C., Cazenave, T., Jacopin, E.: Optimal Solving of Constrained Path-Planning Problems with Graph Convolutional Networks and Optimized Tree Search, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, https://doi.org/10.1109/iros40897.2019.8968113, 2019
38. Liu, R., Wang, J., Zhang, B.: High definition map for automated driving: overview and analysis. J. Navig. **73**(2), 324–341 (2019b). https://doi.org/10.1017/s0373463319000638
39. AGI, An open-source JavaScript library for world-class 3D globes and maps, http://cesium.com, 2019
40. Nunes, C., De Craene, M., Langet, H., Camara, O., Jonsson, A.: Learning decision trees through Monte Carlo tree search: an empirical evaluation, Wiley Interdisciplinary Reviews-data Mining and Knowledge Discovery https://doi.org/10.1002/widm.1348, 2020
41. Li, Y., Fu, M., Xu, J.: Monte Carlo tree search with optimal computing budget allocation, in: 2019 IEEE 58th Conference on Decision and Control (CDC), IEEE, https://doi.org/10.1109/cdc40024.2019.9030099, 2019