International Journal of Research and Technology in Electrical Industry

journal homepage: ijrtei.sbu.ac.ir

Shahid Beheshti University

Quadrotor Low-Noise Path Planning Using Reinforcement Learning Optimization

Roham Salehipour[1], Mostafa Abedi[1,*], Fatemeh Jahangiri[1]

[1]*Electrical Engineering Faculty, Shahid Beheshti University, Tehran, Iran*

ARTICLE INFO

Article history:
Received: 07 October 2024
Revised: 26 November 2024
Accepted: 16 December 2024

Keywords:
Quadrotor
Low-Noise trajectory
Noise abatement procedures
Reinforcement learning
Trajectory planning
Obstacle avoidance
Delivery

ABSTRACT

This paper applies the Q-learning method, a reinforcement learning (RL) technique, to a quadrotor for generating a low-noise trajectory while avoiding obstacles. The proposed method introduces a novel Q-value function, constructing a 2D surface that preserves key environmental features. This approach simplifies the path-finding problem in a 3D space to a 2D space problem. By relying on data derived from the pre-calculated 2D surface, online path planning can be executed in the presence of unpredictable environmental changes, significantly reducing computational complexity and addressing a key challenge in the field. The Q-learning algorithm is developed by defining two cost functions to avoid obstacles and reduce the perceived noise level. To calculate the Sound Pressure Level (SPL) of the noise, the perceived noise model is derived using the Gutin equation. Additionally, the Octomap 3D optimizer is employed to map obstacles. Unlike related works, this approach employs noise observers vertically and horizontally, leading to more accurate environmental mapping. Furthermore, the proposed algorithm ensures global optimal paths while avoiding the local minima that are commonly encountered in similar optimization approaches. Finally, the performance of the proposed methodology in pathfinding and noise reduction is demonstrated through a practical example involving a quadrotor.

1. Introduction

Quadrotors have gained considerable interest due to their small size, low cost, and maneuverability. These aerial devices are ideal for various outdoor applications, particularly in urban areas, such as delivery services, response to disasters, and inspections [1, 2].

Using a quadrotor in urban areas amplifies the challenges of dealing with obstacles, which can be either fixed [3-5] or mobile [6]. Song et al. [7] proposed a near-time-optimal trajectory generation method for drones navigating through a set of waypoints with fixed obstacles. For mobile obstacles, trajectory finding often takes place online. Chen et al. [8] introduced a motion primitive generation algorithm focusing on time optimality, utilizing an integrated solver combining offline and online components for mobile obstacles.

Among trajectory-finding methods for unmanned aerial vehicles (UAVs), learning-based approaches have gained significant attention due to their ability to identify optimal paths while considering environmental constraints [9, 10]. Li et al. [11] proposed a deep reinforcement learning (RL) method for path planning in intelligent vehicles, enabling them to adapt to environmental changes. Battocletti et al. [12] introduced an RL-based approach for exploring unknown environments, where two RL agents are trained: the first assigns waypoints to multiple UAVs and explores the environment, while the second agent implements a path planning algorithm to guide the UAVs efficiently. Rubí et al. [13] applied RL and data from a LIDAR sensor

* Corresponding author
E-mail address: mo_abedi@sbu.ac.ir
https://orcid.org/0000-0002-6021-8242
http://dx.doi.org/10.48308/ijrtei.2024.237164.1059

to detect obstacles and precisely track a path by adjusting the vehicle's velocity to match the path's shape. Among RL-based approaches, Q-learning has been widely adopted for path-finding problems. Zhang et al. [14] have utilized the Q-learning algorithm to obtain optimal or suboptimal safe flight trajectories by constantly learning to maximize the reward function. Fuzzy reinforcement learning was proposed by Tran et al. [15] and applied to compute the Strictly Negative Imaginary (SNI) controller gain.

Research on path-finding characteristics has shown that the path pattern can significantly influence application behaviors, including factors such as energy consumption [16, 17], speed [18], conservativeness [19], and noise effects [20, 21]. Han et al. [19] proposed a grid-optimized indoor path-planning algorithm for UAVs in dense environments. Morrell et al. [22] compared three trajectory algorithms, evaluating the results based on path features such as speed and the proximity of the quadrotor to obstacles.

On the other hand, noise reduction has been addressed through various methods based on the type of observers and UAVs. Some works studied the application dynamics, considering their aeroacoustics or aerodynamic characteristics. Wu et al. [23] proposed an aerodynamic noise reduction method to improve the spanwise blade shape of electric propeller aircraft. Treuren et al. [24] defined propeller tip modifications to reduce acoustic noise in quadrotor propellers. Some studies have designed controllers aimed at reducing noise. For example, Mysore et al. [25] focused on training RL agents for a quadrotor, considering the noisy control process and employing a new reward structure. Guan et al. [26] developed a noise attenuation method for a quadrotor using phase synchronization based on genetic optimization. Recently, the problem of path-finding focusing on noise reduction has gained significant attention and has been addressed in only a limited number of studies. Indirect trajectory optimization for small aircraft and reducing the noise impact is studied by Galles et al. [27]. Dieumegard et al. [28] proposed a Mesh-Adaptive Direct Search algorithm to find locally optimal trajectories for a rotorcraft, considering a realistic noise footprint. Additionally, multiple surrogate models are defined based on problem knowledge, including those relying on the noise model and neural network methods. A large number of local minima characterizes the problem.

Among the references related to noise reduction, these studies have not thoroughly addressed the issue of determining an optimal trajectory that avoids collisions with obstacles. Furthermore, in papers focusing on this problem, the computational challenges associated with training reinforcement learning (RL), especially when performed online, are time-consuming and involve significant calculation difficulties. Moreover, many of the papers mentioned above only exhibit local minimum and do not fully address enhancing the noise estimation mechanism, which can be achieved by incorporating additional microphones.

In this study, we focus on small-sized UAVs, such as quadrotors, designed for applications like package delivery while addressing the impact of noise and obstacles in finding the optimal path. To enhance the accuracy of environmental data collection, including details such as noise and obstacles, the quadrotor's missions are conducted in close proximity to microphones (observers). Unlike the approach used in reference [28], both vertical and horizontal microphones are employed to improve the environmental modeling, thereby enhancing path-planning accuracy. A key challenge in noise reduction is accurately estimating the observed noise level. To address this, we propose a three-step process: First, the Rapidly Exploring Random Tree (RRT) algorithm generates random paths and identifies feasible waypoints. Second, a flight simulator is developed to determine the quadrotor's position and attitude. Finally, the noise perceived by the observer is estimated using the Gutin equation, where the observers are assumed to be human. To estimate obstacles, environmental data is generated using sensor-based data maps, which are then refined through optimization with the Octomap 3D optimizer. The Q-learning algorithm uses the combined noise and obstacle estimation data as the reward function. The policy derived from this algorithm is then used to compute the quadrotor's optimal path, balancing noise reduction and obstacle avoidance. A trade-off between noise and obstacles is considered when calculating the shortest path, as minimizing noise may require a longer route to reduce the sound perceived by the observer. Therefore, a parameter reflecting the relative importance of noise in the calculations is introduced, allowing the path to be adjusted for different scenarios based on the significance of noise.

Moreover, the results are expanded by developing a new Q-value function that shapes a 2D surface. This approach allows the derivation of optimal features required for path planning using the surface, eliminating the need for recalculations. As a result, the path-finding problem in a 3D state is simplified into a 2D problem. Since the data can be obtained from the pre-calculated 2D surface, online path planning can handle unpredictable changes in the environment, such as static or dynamic obstacles, with significantly reduced computational effort. This effectively reduces the computational complexity needed to derive the new policy, making it feasible to implement on smaller commercial UAVs without the need for powerful processors.

The main contributions of this paper are summarized as follows:

1) A unique reinforcement learning method based on Q-learning is introduced, combining dual rewards to balance noise reduction and obstacle avoidance. This dual-reward system, along with a noise-weight parameter, allows for an effective trade-off between minimizing noise and ensuring obstacle avoidance, which is particularly relevant for operations in noise-sensitive urban environments. 2) This approach uses a novel 2D Q-value function to create a simplified surface while preserving the essential features of the path. This solution reduces computational demands, making it ideal for urban delivery quadrotors with limited onboard hardware and software. 3) The application of reinforcement learning, especially Q-learning, to path planning with a focus on noise reduction is a relatively unexplored area, particularly in urban settings where quadrotors are used for delivery. 4) Additional innovations include vertical

and horizontal noise observers that capture precise environmental data and enable dynamic adjustments in noisy, obstacle-dense environments. This ensures the algorithm remains accurate and responsive to real-time obstacles or noise sources changes.

The remainder of the paper is organized as follows: Section 2 proposes the main procedure of this paper. The environment's modeling is explained in Section 3. Section 4 describes the waypoints generating process using the RRT algorithm. The steps for noise estimation and calculation of the perceived noise by the observers are explained in Section 5. Section 6 proposes the optimizer procedure containing the reward function, value table, and policy. The simulator block is described in Section 7. The effectiveness of the proposed method is evaluated using a numerical example in Section 8. Finally, the paper is concluded in Section 9.

2. The low-noise quadrotor trajectory problem

The block diagram illustrating the entire procedure is shown in Fig. 1.

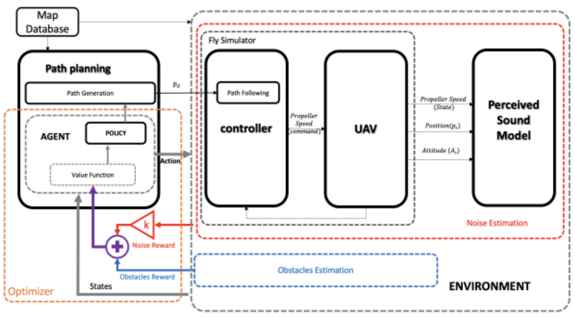


Fig. 1. Block diagram of the overall procedure

Environmental features, which influence both the path and the behavior of the quadrotor, are derived from the map database block. Control signals are generated based on reference paths from the path planning block, which affect the quadrotor's position and attitude through propeller speed commands. The quadrotor's position, attitude, and propeller speeds are then used to estimate and calculate the perceived noise. Additionally, the value functions, part of the optimizer, are computed using data from the reward functions, considering the estimated noise and obstacles. With this value function data and the reward-punishment strategy, the policy determines an optimal path that minimizes perceived noise and avoids obstacles. The main algorithm for this work is provided in Algorithm 1, which will be further explained in the following sections.

Algorithm 1

Inputs

The sensitive and populated places, the altitude (z), the velocity of the quadrotor while landing (v), and the quadrotor's landing position (x_g, y_g, z_g) are used as inputs.

Envrionment modelling (section 3)

Map data based on the 3D scanner.

The 3D scanner's data is converted to Octomap.

The free environment using Octomap is derived.

Waypoint generation (section 4)

The starting points (x and y) are selected randomly for different altitudes.

Using the RRT algorithm, the waypoints are calculated between random starting points and the destination.

Smooth and feasible waypoints are derived based on the quadrotor's position.

Estimation of the perceived noise (section 5)

Based on the mapping, an observer is set for each $5m^2$ of the environment.

Based on the quadrotor's position, the distance between the quadrotor and the observer (r) is calculated.

The sound pressure level (SPL) is obtained using the Gutin equation (Eq. (1) in section 5).

The perceived noise (L) is calculated for each observer at each waypoint (Eq. (5) in section 5).

Optimizer (section 6)

Space Segmentation

Divide the 3D space into smaller segments.

Calculation of the reward functions (section 6.1)

The noise reward function (R_{Noise}) is obtained (Eq. (6) in section 6.1).

The obstacle reward function ($R_{Obstacles}$) is obtained (Eq. (7) in section 6.1).

Multiplying K as the noise importance factor, the total reward function (R_{Total})is obtained (Eq. (8) in section 6.1).

The reward values are assigned to different segments as initial value functions.

Value Table Formation (section 6.2)

The final value functions are calculated and assigned to space segments through an iterative process, and the value table is formed (Eq. (11) in section 6.2).

2D Surface Extraction

Derive a 2D surface by selecting the highest values in each value table column based on the proposed Q-value matrix (Eq. (13) in section 6.2).

Policy (section 6.3)

The optimal policy will be obtained based on the calculated value functions.

Path generation using offline policy (Sub-section 6.3.1)

Process the positions of all waypoints on the obtained 2D surface.

Determine the orientations based on the neighboring waypoints with higher values.

Save all orientations in a policy matrix (Eq. (14) in section 6.3.1).

For each position of the quadrotor, the movement direction is called from the above matrix, and the optimal path of the quadrotor is formed towards the destination point.

Path generation using online policy (Sub-section 6.3.2)

Store 2D surface data on the quadrotor.

Compare the real-time position of the quadrotor with the stored 2D surface data.

Neighboring waypoints with the highest values are selected and commanded as the next quadrotor's position, and thus, the path is generated online.

Simulator (section 7)

The generated path is converted to roll, pitch, yaw, and thrust, and the data is sent to the controller.

The position and attitude, as well as the propeller's velocity, are derived as simulator outputs.

3. Environment's obstacles modelling

In this section, we describe the method used to model the environment. The environmental data primarily originates from sensor-generated data maps, which are then refined through an optimization procedure. A powerful optimization tool, the Octomap 3D optimizer [29], enhances the environmental model by providing greater detail and improved resolution. The optimizer uses point cloud data as inputs. Based on these inputs, the environment is divided into small squares, which are further subdivided to generate more detailed information. This process categorizes the entire environment into two parts: free spaces and obstacles. The environmental model used in this study, along with the division between free space and obstacles, is shown in Fig. 2. The figure on the left is generated from the data map, while the figure on the right represents the environment transformed into Octomap, which is used as the obstacle model in reinforcement learning (RL).

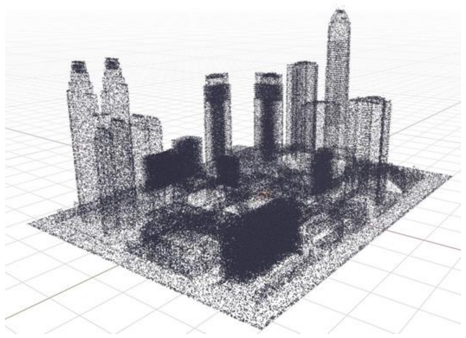
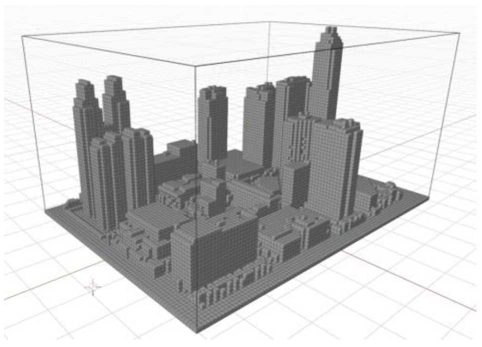

(a)

(b)

Fig. 2. The cloud points (a) and the obtained environment model based on the cloud points (b)

4. Waypoint Generation

This section generates feasible paths, including different waypoints, in the free space using the RRT algorithm. Each waypoint is then assigned a value through reward and value functions, as detailed in the following sections, and the final optimal path is determined based on these values.

Using the RRT algorithm [30], an initial node x_{init} with uniform distribution is randomly selected. If the chosen node belongs to the obstacles, another node x_{new} is selected; otherwise, a path E_i between the node and the destination node x_{goal} is shaped if the node belongs to free space. Doing so for N iterations, different candidate paths toward the destination point are obtained. These paths are then used as set points for a waypoint generator to improve their feasibility and smoothness. This is essential, as the flight mode necessitates paths that the quadrotor can navigate accurately, even around curves. In Fig.3, all random paths found by the RRT algorithm from any arbitrary starting point toward the target position are shown.

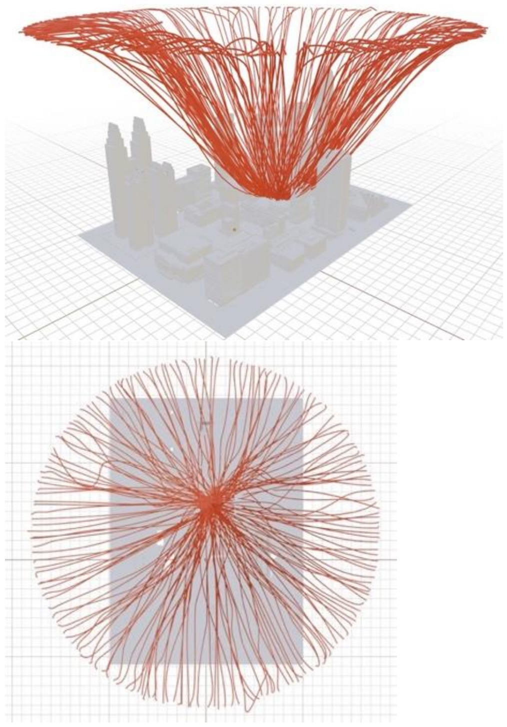


Fig. 3. Random paths found by the RRT algorithm

5. Noise estimation and the perceived noise model

Generally, the quadrotor is the main source of noise, and two main procedures are used to reduce the noise levels perceived by the microphones (observers):

1-Taking distance from the observer: as long as the noise source has a longer distance from the observers, the heard noise by the observers is less severe [27].

2-Noise abatement procedures: by reducing the engine's RPM, the propellers' RPM is also decreased, which lowers the noise [31].

To achieve this, it is essential to estimate the perceived noise by the observers. A modified version of the Gutin pressure equation is used to calculate the perceived noise level at different waypoints. At each waypoint, microphones are arranged both horizontally and vertically (Fig. 4). A vertical microphone is assumed to be present for each section of the environment and on the buildings since there exist auditor. Based on their relative distance from the quadrotor, the perceived noise is then calculated.

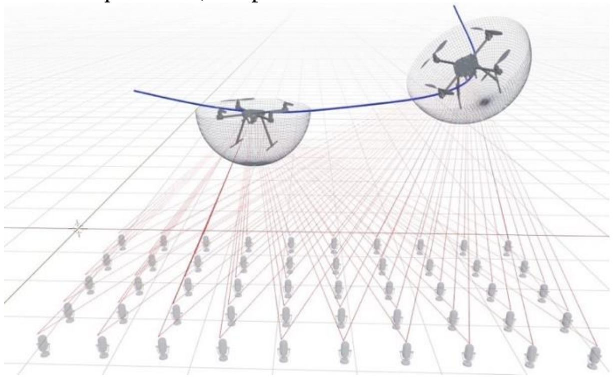


Fig. 4. The relative position of the quadrotor with respect to the array of microphones at different waypoints

Fig.5 shows the overall propeller geometry used in the Gutin model. This model is used as a basis for deriving the perceived sound by the observers.

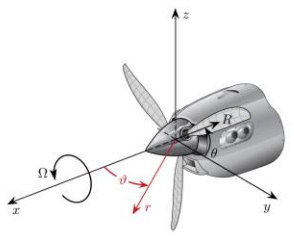


Fig. 5. Gutin model propeller geometry

The sound metric, which is the Sound Pressure Level (SPL) calculated by the engineer form of the Gutin equation, is defined as follows [32]:

$$P_{rms} = \frac{P_{disc}}{2\sqrt{2}}\frac{R_t}{r}M_t(\frac{M}{M_e^2} - cos\vartheta)qnJ_{qn}(qnM_e sin\vartheta), \quad (1)$$

where R_t is the tip radius of the propeller, r is the distance between the center of the propeller and the observer, M is the Mach number of the vehicle, M_t is the Mach number of the propeller, M_e is the Mach number of the effective propeller radius (usually considered as $((0.7) - (0.8)R_t)$, J_{qn} is the ,q is the harmonic order, n is the number of the propeller blades, and the propeller disc pressure is $P_{disc} = T(\pi R_t^2)^{-1}$. In (1), ϑ is the directivity angle which can be calculated as follows:

$$\vartheta = arcos\frac{v_a r_{o/a}}{|v_a|.\left|r_{o/a}\right|}, \quad (2)$$

where v_a is the propeller velocity in revolution per minute. The relative position of the observer to the quadrotor is obtained as follows [32]:

$$r_o = r_a + r_{o/a}, \quad (3)$$

in which r_o is the position of the observer, r_a is the position of the aircraft and their difference is $r_{o/a}$, hence:

$$r_{o/a} = r_a - r_o. \quad (4)$$

The distance between the observer and the noise source is determined by calculating the norm. Since the acoustic sources are considered incoherent and the quadrotor is equipped with four motors, each driving a propeller, an additional term is included to account for the extra propellers. Therefore, the sound perceived by each microphone (observer) is estimated as follows:

$$L = 20log_{10}\left(\frac{P_{rms}}{P_0}\right) + 4log_{10}\left(N_p\right). \quad (5)$$

where N_p is the number of propellers on the quadrotor and P_0 is the atmosphere pressure. In this work, the sum of perceived noise from 1000 observers are

6. Optimizer: Q-learning

The Q-learning algorithm is applied as the optimizer. The proposed method uses Q-learning with a value-table, where each cell represents a location in the environment, influenced by obstacles (e.g., buildings) and noise sources. Since urban obstacles are static, values can be pre-mapped and assigned. The table is mapped to a 2D surface and imported into the quadrotor's system. Noise observers, fixed in place, have constant values on the map. With these pre-mapped factors, the quadrotor (as the RL agent) learns the optimal path, balancing noise reduction and obstacle avoidance. The stages of RL implementation are as follows: First, the environment data is collected, including both obstacles and noise sources. Then, random obstacle-free paths are generated from the start point to the goal using the RRT algorithm. Noise data is created for these obstacle-free paths using MATLAB. Next, the map and noise data are imported to define the environment for the RL agent. The reward function is specified by assigning values to obstacles, free spaces, and noise. The coefficient K is introduced to define the relative importance of noise versus obstacles in the calculations. The value function is then created, assigning values to all points in the space using RL, even for areas that do not have direct data. Finally, the optimal policy is derived, and the environment map is constructed based on the learned value function to guide the quadrotor's path planning. In the remainder of this section, the reward function and the procedure of obtaining policy based on the value table are further indicated.

6.1. Reward function

The reward function is the sum of the estimated noise and obstacle rewards. The noise reward is calculated using the following equation:

$$R_{Noise} = \sum_{1}^{n_0} L.T.S.P.N_p. \quad (6)$$

where n_0 is the number of observers, L is the sound perceived by each observer obtained from (5), and T is the reward or punishment value assigned for the percentage of annoyance derived from Fig.6. S is the sensitivity of the observers' position and P is the observers' population, where the values are assigned from Table.1.

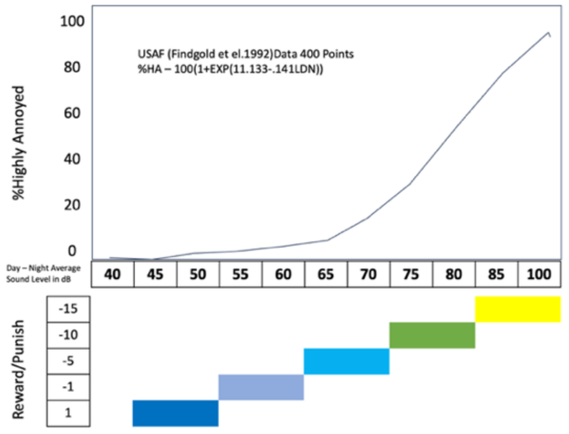


Fig. 6. The highly annoyed (HA) values based on day and night sound levels [33]

Table. 1. Assigned numbers for S and P

P	Small Population =1	Medium Population=3	Large Population=7
S	Working place=1	Residential areas=3	Hospital=7

The obstacles reward is calculated according to the following equation:

$$R_{Obstacles} = \begin{cases} Obstacles & -100 \\ Freespace & -1 \\ Goal & 500 \end{cases} . \qquad (7)$$

Fig.7 shows the initial values of the reward function. The blue regions are obtained from sensors indicating obstacles, and the red regions denote the values for the paths generated by the RRT.

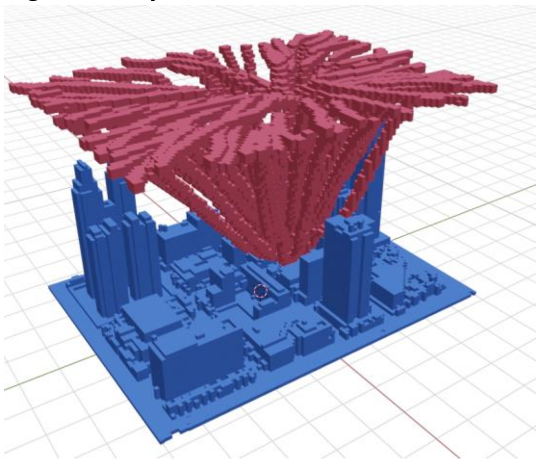


Fig. 7. Calculated reward functions for the noise and obstacles

Considering the noise estimation strategies mentioned earlier, such as adjusting the distance from observers and reducing propeller RPM, it is important to note that the resulting path may not always align with the goal of optimal path planning in certain scenarios. Due to this negative effect, a trade-off between the noise and obstacle rewards is needed. Therefore, using a weighting parameter K to apply the importance of the noise effect compared with the obstacle effect, the total reward is obtained as follows:

$$R_T = R_{Obstacles} + KR_{Noise} \qquad (8)$$

Therefore, the reward values for all segments will be updated according to equation (8) through the learning process, subsequently serving as the initial values for the value function.

6.2. Value table

The Q-learning algorithm is a reinforcement learning method used for policy determination based on the value function. To find the optimal path for the quadrotor, it is important to distinguish between fixed and movable obstacles. In this study, obstacles are initially fixed during the offline learning phase. The results are then extended to develop an online path-planning policy that accounts for movable obstacles. Since this work also considers the impact of noise and aims to find the optimal path that minimizes the quadrotor's noise while avoiding obstacles, both noise and obstacle data are incorporated into the value function as rewards. The values for obstacles in the value table are initially assigned the same, as are the values for microphones and their neighbouring areas. The value function is defined as follows [34]:

$$V(s) = \max\left(R_T(s,a) + \textstyle\sum_{s'} V(s')\right), \qquad (9)$$

where $V(s)$ is the current value, $R_T(s,a)$ is the reward value obtained from (8), which is considered as initial variables of the value function, s is the states which are the quadrotor's position and engine's speed, a is the taken action, s' is the previous states and $V(s')$ is the previous values. Moreover, to include the action's role in the value function (9), the Q-value, which relies on the previous values, is formulated as follows:

$$V(s) = R_T(s,a) + \gamma \sum\nolimits_{s'} (T(s,a,s').V(s')). \qquad (10)$$

Where $0 < \gamma \leq 1$ is the forgetting rate and $T(s,a,s')$ is the temporary difference. Replacing $V(s')$ with $maxQ(s',a')$ which is the maximum value of the previous Q-values, the following equation is derived:

$$Q(s,a) \\ = R_T(s,a) + \gamma \sum\nolimits_{s'} (T(s,a,s').maxQ(s',a')). \qquad (11)$$

Based on the equation above, Q-learning uses an off-policy approach, separating the acting policy from the learning policy. Referring to the Q-value equation in (11), the value for each segment of the environment is computed. This learning process continues for 1500 iterations to build a 3D table of values. Figure 8 shows the final value functions obtained for different space segments. Small circles represent lower values, while larger circles correspond to higher values. The purple areas highlight dominant obstacles, and the pink regions indicate noise dominance.

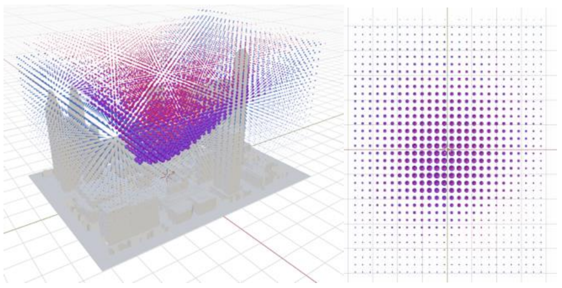


Fig. 8. Value function after 1500 iterations

The Q-value equation derived in (11) resulted in a 3D value table, which can increase the complexity of determining the policy. Therefore, the following algorithm is used to simplify the Q-value function:

```
$from\ \ x = 0\ to\ I_x$
  $from\ \ y = 0\ to\ I_y$
    $Q_{new}(x, y) = Max_z\big(Q(x, y, z)\big),\ \ z = 0\ to\ I_z$        (12)
  $end$
$end$
```

Therefore, by only selecting the highest value of the z-axis data in the value table, it transforms into a two-dimensional Q-value matrix as follows that collects Q-values in different positions:

$$Q_{new} = \begin{pmatrix} Q_{new}(0,0) & \cdots & Q_{new}(0, I_y) \\ \vdots & \ddots & \vdots \\ Q_{new}(I_x, 0) & \cdots & Q_{new}(I_x, I_y) \end{pmatrix}. \tag{13}$$

The above relationship is represented as a 2D surface, with the Q-learning policy using only the values assigned to this surface to generate the path, effectively reducing computational complexity. The 2D surface derived from Equation (13) is shown in Figure 9.

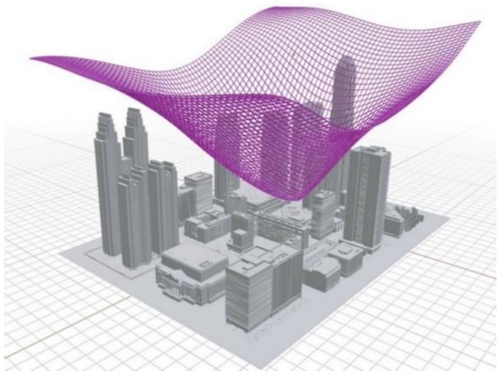


Fig. 9. The 2D surface that is derived to obtain the policy

6.3. Policy

In this section, policies are derived for offline and online path planning scenarios, which are detailed in the following subsections.

6.3.1 Offline Path Planning Policy

In offline path planning, the quadrotor's starting points are not predefined during the learning stage. Instead, all potential paths to the designated destination are generated during the training phase on the 2D surface to derive the offline policy. For each point on the 2D surface, the neighboring waypoint with the highest value function is selected. According to Table 2, one of the directions is then assigned to the current location to guide the quadrotor toward the chosen waypoint. The quadrotor's path is constructed by following waypoints with higher value functions, leading toward the target point. Based on this strategy, a policy matrix is stored according to (14), and a direction o_i, $i = \{1, \dots, 9\}$ is assigned to each x and y component on the proposed surface:

Table. 2. Assigned values to save the quadrotor's orientations

Orientations o_i	➡	⬅	⬆	⬇	⬈	⬉	⬋	⬊	**goal**
Assigned values	1	2	3	4	5	6	7	8	9

$$\pi = \begin{pmatrix} o_i(0,0) & \cdots & o_i(0, I_y) \\ \vdots & \ddots & \vdots \\ o_i(I_x, 0) & \cdots & o_i(I_x, I_y) \end{pmatrix}, \quad i = \{1, \dots, 9\} \tag{14}$$

In each position of the quadrotor, the movement direction is called from the above matrix, and therefore, the quadrotor is optimally directed towards the destination. Fig. 10 shows the stored orientations for different points on the 2D surface.

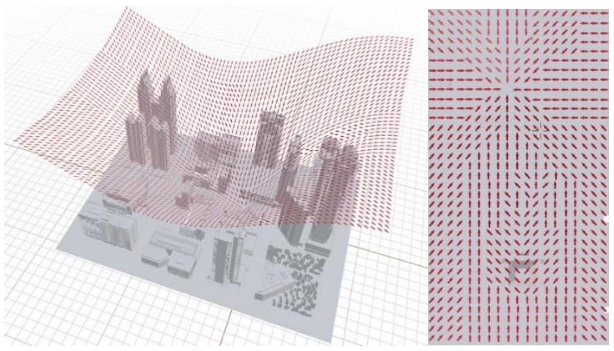


Fig. 10. The extracted policy from any starting point toward the destination

6.3.2 Online Path Planning Policy

The 2D surface data is initially stored on the quadrotor in online path planning. The quadrotor's real-time position is compared with the stored 2D surface data during the path planning process. At each position, unlike the offline policy, the online system dynamically selects the nearby waypoints with the highest value function. Subsequently, the path with selected waypoints is commanded in real-time; the policy will be executed without needing to precompute the paths. This facilitates the potential for real-time calculations, which is especially advantageous

in situations with mobile obstacles. Furthermore, global path planning becomes feasible as the policy-finding process occurs online, in contrast to the offline policy, which restricts path calculations to local scopes.
The proposed algorithm can be used for online applications by combining the pre-mapped method with occasional real-time updates from LiDAR. This hybrid approach enables the quadrotor to adapt to environmental changes, making it more suitable for dynamic urban environments.

Remark 1. The parameter K, introduced in equation (8), enables a flexible trade-off between obstacle avoidance and noise reduction, adjustable based on quadrotor size and environmental complexity. As the environment or quadrotor size increases, the emphasis shifts from noise reduction to obstacle avoidance. This ensures that obstacle avoidance takes priority in challenging conditions, while noise reduction is prioritized in simpler environments with smaller quadrotors.

Assumption 1. Obstacles are not large enough to block all possible path to the goal. This assumption guarantees that the proposed path-planning algorithm remains feasible.

7. Flight simulator

In this step, the quadrotor dynamics model and flight control system collaborate to guide the quadrotor along a predefined path. The flight control system adjusts the quadrotor's position based on this path, utilizing a path-following controller that translates reference position data into essential control parameters: roll, pitch, yaw, and thrust. The controller block integrates feedback on the quadrotor's real-time position with the input data to calculate the appropriate propeller speeds, subsequently applied to the dynamic model. The quadrotor dynamics model and the backstepping control approach employed here are referenced in [35].

8. Simulation results

In this section, the developed method is evaluated using a numerical example of a quadrotor. The parameters used in this work are gathered in the table. 3. as it is shown:

Table. 3. Parameters used in the simulation section

Parameters	values
Quadrotor's mass	$m = 320g$
Distance between two engines	$l = 10\ inch$
Maximum velocity of engines	$v_{amax} = 10\ inch$
Forgetting factor	$\gamma = 1$
Quadrotor's propellers	$N_p = 2$
Atmospheric pressure	$P_0 = 100000\ Pa$

The required CPU time for the proposed algorithm, the initial path planning using the RRT algorithm, took approximately 1835 seconds (around 30 minutes) and was performed only once during the setup phase. The noise estimation process required approximately 180,000 seconds (around 2 days), and the reinforcement learning (RL) policy computation took about 120,000 seconds (approximately 1.5 days), both of which were also executed only once during the training phase. Once the policy function was computed, the path planning for any selected starting point within the mapped environment could be executed in approximately 2.2 seconds, enabling rapid path generation during deployment. These computations were carried out on a system with an Intel Pentium G4400 @ 3.30GHz CPU and 20 GB of RAM. The proposed approach minimizes online computational demands by conducting computationally intensive processes offline, such as noise estimation and policy training, ensuring practicality for real-time applications on quadrotors with limited processing capabilities.

Figure 11 illustrates the optimal path for the quadrotor, calculated based on specific assigned values and a randomly chosen starting point. In the figure, the purple areas represent the highest values, constituting the derived policy. With this information and the selection of a starting point, an optimal path for the quadrotor is determined.

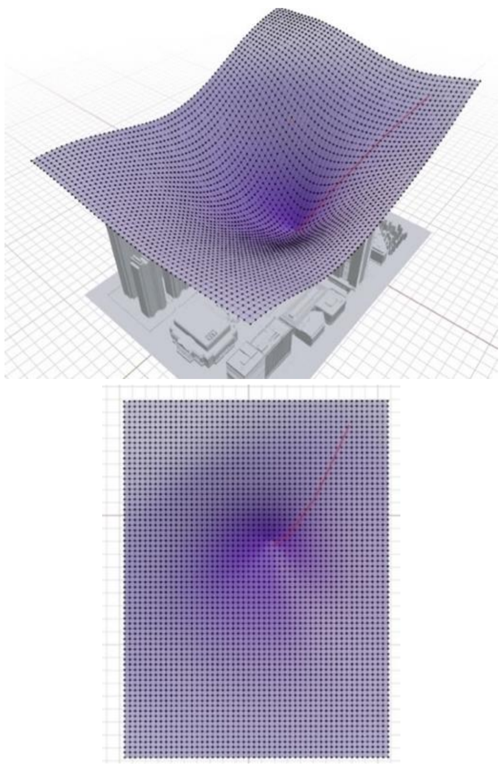


Fig. 11. The optimal path derived for the quadrotor

In Fig.12, the optimal path and the noise footprint for the quadrotor are shown based on the offline policy. Areas marked in yellow indicate the highest noise levels, while noise decreases as the color shifts to blue and purple. Figure 12 shows the path from a single random starting point. For a more comprehensive analysis, Figure 13 illustrates the policy generated for 200 different local starting points. This figure displays the optimal paths for each starting point, considering obstacle avoidance and noise reduction principles.

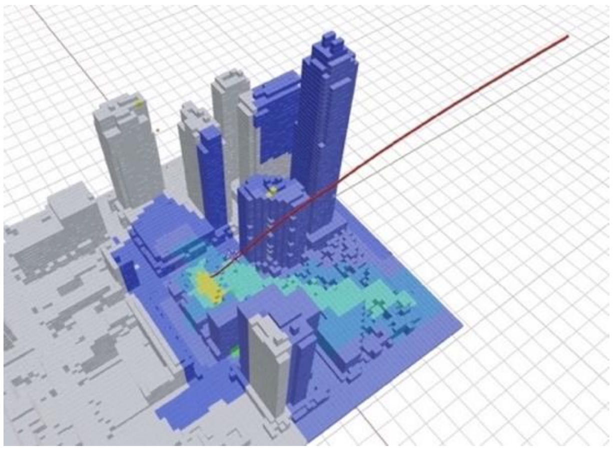


Fig. 12. The noise footprint derived for the quadrotor based on an optimal path

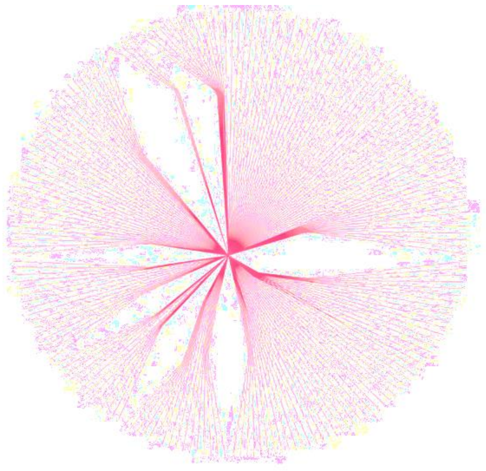


Fig. 13. The optimal paths for 200 local starting points from the above point of view

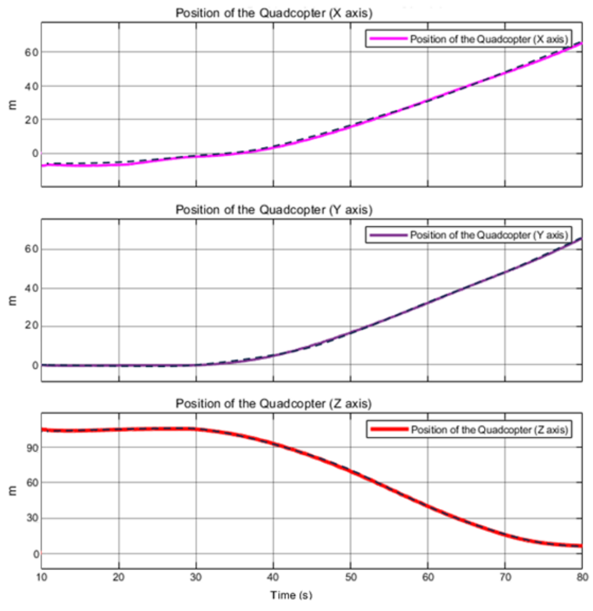


Fig. 14. The quadrotor's position according to the derived desired path

Figure 14 illustrates the quadrotor's position as it follows the path derived from the policy, showing successful adherence to the desired trajectory.

Figure 15 compares paths generated with and without the 2D surface for a mobile obstacle (the red sphere). Using the 2D surface effectively reduces the number of generated paths. In this scenario, the online path planning integrates pre-computed conditions and new dynamic conditions, such as moving obstacles. This approach already saves the previous conditions on a 2D surface. This surface does not store the orientation but represents the value function of the space. When dynamic conditions like moving obstacles arise, their effect is directly considered in the value function, and an updated value function is computed. Subsequently, the policy function and the revised path are derived. The pre-computed surface significantly enhances computational efficiency. Instead of recalculating the entire 3D space during online updates, only the 2D surface needs to be recomputed. This will eventually reduce computational complexity while ensuring accurate path planning.

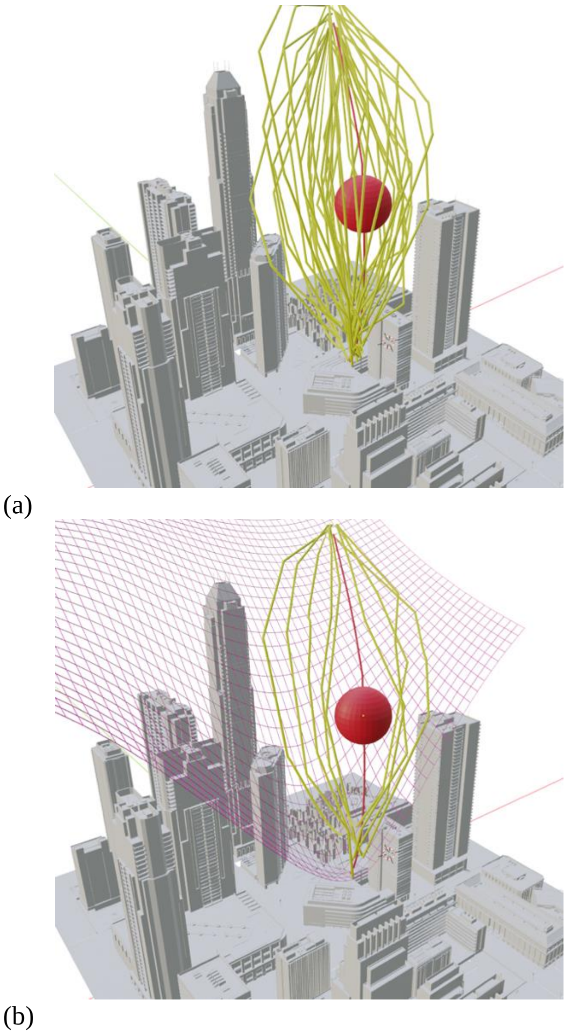


Fig. 15. The path comparison with the 2D surface (b) and without it (a) for a mobile obstacle

A new scenario is considered for a quadrotor to compare the effect of noise on path deviation based on the online policy. In this study, the punishments are fixed, as they influence the distance between the obtained path and the obstacles. Based on (8), the effect of the noise importance for $K = 0, 1, 2, 4$ is evaluated, and the path is calculated for each value, as shown in Fig.16. As shown in Figure 16, all the buildings, which are made up of smaller sections, are both obstacle sections and microphone sections. The reason for the color variation is the sound intensity detected in each section.

According to the above figure, when $K = 0$, the problem only considers obstacle avoidance and noise effects are not part of the optimal path calculation. This approach ensures obstacle avoidance but may cause the optimal path to pass close to obstacles. It is important to note that the noise feature is calculated specifically within the free space of the environment. Therefore, increasing the importance of noise directs the path to the free space to reduce the noise but makes it longer. Therefore, a compromise must be made to choose the appropriate value of K.

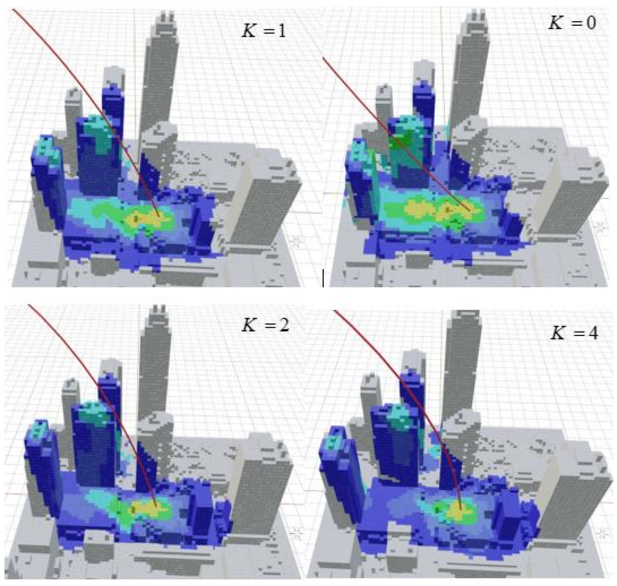


Fig. 16. Paths obtained based on changing the noise importance factor

The results comparing each derived path for different values of K are shown in Table.3.

Table. 4. Results comparison based on different values for K

Noise importance value	$K = 1$	$K = 1$	$K = 2$	$K = 4$
Points making the waypoints	25	29	34	37
Obstacle avoidance value	458	406	385	365
Number of engaged observers over 40 dB	953	897	865	864
Noise value over 60 dB	443	658	783	1254
Final noise value	0	658	1634	5373

From Table 4, it's evident that as the value of K increases and in relation to the number of primary waypoints (column 2), the resulting paths tend to be longer. Conclusively, fixing the time intervals for each waypoint increases the time required to traverse the paths. This is evident in the total obstacle avoidance values (column 3), indicating an increase in path length. Within the reinforcement learning (RL) framework, this results in harsher penalties for the agent due to the defined death penalty.

In urban areas, typical background noise levels are around 40 dB, with sounds above 60 dB being more noticeable to observers. As shown in the table above, increasing the K factor reduces the number of engaged observers (column 4), guiding the quadrotor's path toward a lower-noise trajectory. The noise values in Table 4 (column 5) demonstrate that, although the number of engaged observers decreases, the quadrotor receives a higher reward for noise reduction, indicating a decrease in the average noise perceived by observers. However, as shown in the final noise value column, increasing K linearly does not produce a proportional change in the final noise estimation. This is because, with each iteration, the perceived noise decreases for the observers, increasing the estimated noise value.

Fig.17 shows the paths derived from different values of K based on the time taken to reach the destination. As illustrated in this figure, the more the value of K increases, the path will become more sensitive to the noise; however, the CPU time will increase due to calculation difficulties. This also confirms the increase in transportation time.

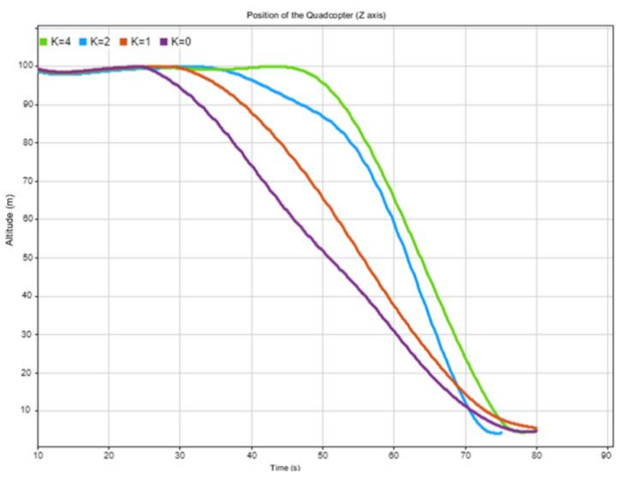


Fig. 17. A comparison of the paths obtained based on different noise importance

9. Conclusion

This paper proposes a Q-learning algorithm for a quadrotor to find an optimal path that avoids obstacles and minimizes noise, using reinforcement learning. Environmental features, including noise and obstacles, are estimated separately using MATLAB's Gutin equation and Octomap. The estimated noise and obstacle data serve as rewards for the value function, which is ultimately used to derive the optimal policy. This policy allows the determination of the best path, balancing noise reduction and obstacle avoidance. The value function table is also used to construct a 2D surface, simplifying calculations by identifying the highest value, as demonstrated in this study. In conclusion, the quadrotor's position results confirm the successful application of the proposed algorithm, which effectively finds a path that allows the quadrotor to navigate near obstacles while minimizing noise. Future work may involve integrating additional noise reduction methods based on dynamic structures or exploring quadrotor velocity optimization.

Declarations

Funding
The authors received no financial support for this article's research, authorship, and publication.

Conflicts of interest/Competing interests
Not applicable.

Availability of data and material
Not applicable.

Code availability
Not applicable.

Ethics approval
There are no ethical issues about this article.

Consent to Participate
Not applicable.

Consent to Publish
Not applicable.

Authors Contributions
Conceptualization: [R. Salehipour], [M. Abedi], [F. Jahangiri]; Methodology: [R. Salehipour], [M. Abedi], [F. Jahangiri]; Formal analysis and investigation: [R. Salehipour]; Writing - original draft preparation: [R. Salehipour], [M. Abedi], [F. Jahangiri]; Writing - review and editing: [R. Salehipour], [M. Abedi], [F. Jahangiri]; Resources: [R. Salehipour]; Supervision: [M. Abedi], [F. Jahangiri].

References

[1]. K.M. Thu, A. Gavrilov, Designing and modeling of quadcopter control system using L1 adaptive control, Procedia Computer Science, 103 (2017) 528-535.
[2] S. Watkins, J. Burry, A. Mohamed, M. Marino, S. Prudden, A. Fisher, N. Kloet, T. Jakobi, R. Clothier, Ten questions concerning the use of drones in urban environments, Building and Environment, 167 (2020) 106458.
[3] H.-Y. Lin, X.-Z. Peng, Autonomous quadrotor navigation with vision based obstacle avoidance and path planning, IEEE Access, 9 (2021) 102450-102459.
[4] I. Iswanto, A. Ma'arif, O. Wahyunggoro, A.I. Cahyadi, Artificial potential field algorithm implementation for quadrotor path planning, International Journal of Advanced Computer Science and Applications, 10 (2019).
[5] B. Zhou, J. Pan, F. Gao, S. Shen, Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight, IEEE Transactions on Robotics, 37 (2021) 1992-2009.
[6] D. Ghorpade, A.D. Thakare, S. Doiphode, Obstacle detection and avoidance algorithm for autonomous mobile robot using 2D LiDAR, in: 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), IEEE, 2017, pp. 1-6.
[7] Y. Song, M. Steinweg, E. Kaufmann, D. Scaramuzza, Autonomous drone racing with deep reinforcement learning, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 1205-1212.
[8] G. Chen, D. Sun, W. Dong, X. Sheng, X. Zhu, H. Ding, Computationally efficient trajectory planning for high speed obstacle avoidance of a quadrotor with active sensing, IEEE Robotics and Automation Letters, 6 (2021) 3365-3372.
[9] A. Puente-Castro, D. Rivero, A. Pazos, E. Fernandez-Blanco, A review of artificial intelligence applied to path planning in UAV swarms, Neural Computing and Applications, (2022) 1-18.
[10] S.T. Brassai, B. Iantovics, C. Enăchescu, Artificial Intelligence in the path planning optimization of mobile agent navigation, Procedia Economics and Finance, 3 (2012) 243-250.
[11] J. Li, Y. Chen, X. Zhao, J. Huang, An improved DQN path planning algorithm, The Journal of Supercomputing, 78 (2022) 616-639.
[12] G. Battocletti, R. Urban, S. Godio, G. Guglieri, RL-based Path Planning for Autonomous Aerial Vehicles in Unknown Environments, in: AIAA AVIATION 2021 FORUM, 2021, pp. 3016.
[13] B. Rubí, B. Morcego, R. Pérez, Deep Reinforcement Learning for Quadrotor Path Following and Obstacle Avoidance, Deep Learning for Unmanned Systems, (2021) 563-633.
[14] T. Zhang, X. Huo, S. Chen, B. Yang, G. Zhang, Hybrid path planning of a quadrotor UAV based on Q-Learning algorithm, in: 2018 37th Chinese control conference (CCC), IEEE, 2018, pp. 5415-5419.
[15] V.P. Tran, M.A. Mabrok, S.G. Anavatti, M.A. Garratt, I.R. Petersen, Robust Fuzzy Q-Learning-Based Strictly Negative Imaginary Tracking Controllers for the Uncertain Quadrotor Systems, IEEE Transactions on Cybernetics, (2022).
[16] D.C. Gandolfo, L.R. Salinas, A. Brandão, J.M. Toibero, Stable path-following control for a quadrotor helicopter considering energy consumption, IEEE Transactions on Control Systems Technology, 25 (2016) 1423-1430.
[17] H. Alkomy, J. Shan, Investigating the Effects of Polynomial Trajectories on Energy Consumption of Quadrotors, IEEE/ASME Transactions on Mechatronics, (2022).
[18] C.S. Gadde, M.S. Gadde, N. Mohanty, S. Sundaram, Fast Obstacle Avoidance Motion in Small Quadcopter operation in a Cluttered Environment, in: 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), IEEE, 2021, pp. 1-6.
[19] B. Han, T. Qu, X. Tong, J. Jiang, S. Zlatanova, H. Wang, C. Cheng, Grid-optimized UAV indoor path planning algorithms in a complex environment, International Journal of Applied Earth Observation and Geoinformation, 111 (2022) 102857.
[20] R. Morris, M. Johnson, K.B. Venable, J. Lindsey, Designing noise-minimal rotorcraft approach trajectories, ACM Transactions on Intelligent Systems and Technology (TIST), 7 (2016) 1-25.
[21] S. Padula, Design of Quiet Rotorcraft Approach Trajectories: Verification Phase, in: 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, 2010, pp. 9309.
[22] B. Morrell, R. Thakker, G. Merewether, R. Reid, M. Rigter, T. Tzanetos, G. Chamitoff, Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles, IEEE Robotics and Automation Letters, 3 (2018) 4399-4406.
[23] Y. Wu, Y.-t. Ai, W. Ze, T. Jing, X. Song, Y. Chen, A novel aerodynamic noise reduction method based on improving spanwise blade shape for electric propeller aircraft, International Journal of Aerospace Engineering, 2019 (2019) 1-10.
[24] K.W.V. Treuren, C.F. Wisniewski, Testing propeller tip modifications to reduce acoustic noise generation on a quadcopter propeller, Journal of Engineering for Gas Turbines and Power, 141 (2019) 121017.
[25] S. Mysore, B. Mabsout, K. Saenko, R. Mancuso, How to train your quadrotor: A framework for consistently smooth and responsive flight control via reinforcement learning, ACM Transactions on Cyber-Physical Systems (TCPS), 5 (2021) 1-24.
[26] S. Guan, Y. Lu, T. Su, X. Xu, Noise attenuation of quadrotor using phase synchronization method, Aerospace Science and Technology, 118 (2021) 107018.
[27] M.B. Galles, B.A. Newman, Reducing the Noise Impact of Small Aircraft through Indirect Trajectory Optimization, in: AIAA AVIATION 2020 FORUM, 2020, pp. 2594.
[28] P. Dieumegard, S. Cafieri, D. Delahaye, R.J. Hansman, Rotorcraft low-noise trajectories design: black-box optimization using surrogates, Optimization and Engineering, (2023) 1-38.
[29] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Autonomous robots, 34 (2013) 189-206.
[30] I. Noreen, A. Khan, Z. Habib, A comparison of RRT, RRT* and RRT*-smart path planning algorithms, International Journal of Computer Science and Network Security (IJCSNS), 16 (2016) 20.
[31] M.B. Galles, N.H. Schiller, K.A. Ackerman, B.A. Newman, Feedback control of flight speed to reduce unmanned aerial system noise, in: 2018 AIAA/CEAS Aeroacoustics Conference, 2018, pp. 2950.
[32] T. Theodorsen, A.A. Regier, The problem of noise reduction with reference to light airplanes, in, 1946.
[33] K. Persson, M. Björkman, Annoyance due to low frequency noise and the use of the dB (A) scale, Journal of Sound and Vibration, 127 (1988) 491-497.

[34] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, Robotica, 17 (1999) 229-235.
[35] A. Das, F. Lewis, K. Subbarao, Backstepping approach for controlling a quadrotor using lagrange form dynamics, Journal of Intelligent and Robotic Systems, 56 (2009) 127-151.