Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Motion planning of a quadrotor robot game using a simulation-based projected policy iteration method*

Li-dong ZHANG†[1,2], Ban WANG[2], Zhi-xiang LIU[2], You-min ZHANG†‡[2], Jian-liang AI[1]

[1]*Department of Aeronautics and Astronautics, Fudan University, Shanghai 200433, China*

[2]*Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Quebec H3G1M8, Canada*

†E-mail: lidongzhang13@fudan.edu.cn; ymzhang@encs.concordia.ca

Received Sept. 15, 2018; Revision accepted Nov. 27, 2018; Crosschecked Apr. 28, 2019

Abstract: Making rational decisions for sequential decision problems in complex environments has been challenging researchers in various fields for decades. Such problems consist of state transition dynamics, stochastic uncertainties, long-term utilities, and other factors that assemble high barriers including the curse of dimensionality. Recently, the state-of-the-art algorithms in reinforcement learning studies have been developed, providing a strong potential to efficiently break the barriers and make it possible to deal with complex and practical decision problems with decent performance. We propose a formulation of a velocity varying one-on-one quadrotor robot game problem in the three-dimensional space and an approximate dynamic programming approach using a projected policy iteration method for learning the utilities of game states and improving motion policies. In addition, a simulation-based iterative scheme is employed to overcome the curse of dimensionality. Simulation results demonstrate that the proposed decision strategy can generate effective and efficient motion policies that can contend with the opponent quadrotor and gather advantaged status during the game. Flight experiments, which are conducted in the Networked Autonomous Vehicles (NAV) Lab at the Concordia University, have further validated the performance of the proposed decision strategy in the real-time environment.

Key words: Reinforcement learning; Approximate dynamic programming; Decision making; Motion planning; Unmanned aerial vehicle

https://doi.org/10.1631/FITEE.1800571 **CLC number:** TP242

1 Introduction

Autonomous systems, such as unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs), and unmanned surface vehicles (USVs) are attracting growing interest in science, industry, and military due to their advantages, such as high maneuverability, extended operational range, and improved personnel safety, when accomplishing a variety of missions (Yuan et al., 2015; Ghamry et al., 2016; Liu et al., 2016; Sharifi et al., 2016; Wang and Zhang, 2018). Especially along with the rapid development of intelligent systems, autonomous systems have become powerful to deal with difficult and complex tasks. Challenges remain though and are restricted by computational capabilities, sensor reliabilities, communication stabilities, and other limits in both software and hardware. Among these challenges, solving the sequential decision problem plays a critical role in building an autonomous system with high computational efficiency and functional intelligence. A sequential decision problem can be modeled as a Markov decision process (MDP) that

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61573282 and 61833013), the Scholarships from China Scholarship Council (No. 201606100139), and the Natural Sciences and Engineering Research Council of Canada

ORCID: You-min ZHANG, https://orcid.org/0000-0002-9731-5943

consists of a set of states, a set of actions at each state, a transition model, and a reward/cost function (Russell and Norvig, 2010). The solution of such a problem, known as a policy, determines which action needs to be taken at each state to achieve the goal state from an initial state. The objective is then to find a feasible and efficient policy that can provide rational decisions for the vehicle in a sophisticated environment to reach the goal criteria. To quantitatively evaluate the performance of the policies, a platform with a certain sequential decision model is needed for further development.

To make rational decisions for quadrotors to reach advantaged status and satisfy goal criteria in the game under a certain formulation, one needs to learn the long-term utilities given sample game states of both quadrotors and terrain information of the game field, and then to generate a near-optimal policy with a long planning horizon. A dynamic programming (DP) algorithm can provide an optimal policy for a sequential decision problem. Nevertheless, it is an intractable problem for an exact DP algorithm to solve a complex game due to the famous curse of dimensionality. McGrew (2008) and McGrew et al. (2010) proposed an approximate dynamic programming (ADP) approach and validated its ability to solve an autonomous air-combat problem. However, they assumed a simplified simulated air-combat problem which restricts the aircraft to level flight with constant velocity. Meanwhile, the state transition dynamics of the air-combat problem were assumed to be a deterministic system. Hence, the size of the game state is exceedingly expanded in comparison to that of state in McGrew et al. (2010). Furthermore, terrain combined with other factors, which was not considered in the previous studies, drains the ability of the aforementioned ADP approach to solve complex decision problems, such as the quadrotor game, using a simple batch manner in the policy evaluation step. In spite of the limited competence of existing schemes, the idea of solving complex sequential decision problems using ADP is still a reasonable one, including establishing the improved efficiency and capability.

Returning to the history of research in sequential decision problems, a fundamental philosophy was established, in which the sequential decision problem could be converted to a problem of determining a one-step optimal operation given the utility of the remaining decision problem (Bellman, 1952). Therefore, in theory, the optimal solution of a sequential decision problem can be found by a recursive calculation of the Bellman equation until all states are visited. In practice, however, due to the three curses of dimensionality of state space, outcome space, and action space, DP is intractable for real systems (Powell, 2007). Remarkable developments in ADP and reinforcement learning were published and tremendously enhanced the power of DP theory to solve practical problems with large state and action spaces (Bertsekas, 1971; Sutton and Barto, 1998). Their main idea was to approximate utilities of states through linear or nonlinear approximation structures and to generate a suboptimal policy through value iteration (VI) or policy iteration (PI) methods. While VI is more straightforward for application and understanding, the PI method is more efficient, as the approximation of utilities is evaluated policy to policy rather than state to state. Hence, it normally takes few iterations for policies to converge before the convergence of the approximated utility value. There are two general options for approximating the utility value of a certain policy: one is straightforward, and the other is efficient. The first approach, known as the "gradient method," directly calculates the approximation of the utility value by solving a least squares problem (Bertsekas and Tsitsiklis, 2000; Bertsekas, 2012). The second approach, known as the "indirect method," calculates the approximated utilities for a fixed policy based on a projected form of the Bellman equation (Yu, 2010, 2012; Bertsekas, 2011). The state-of-the-art research combined the simulation-based methods with the indirect approaches to cut down the dimension of the projected Bellman equation (Buşoniu et al., 2010; Thiery and Scherrer, 2010; Bertsekas, 2015). A multistep version of PI was proposed, where policy improvement was solved based on a multistep Bellman equation (Efroni et al., 2018a,b).

In this study, a one-on-one quadrotor robot game formulation is presented as a platform to examine candidate motion planning strategies. Two deployed quadrotors are assumed to fly inside a cuboid space, containing a static obstacle placed on the ground. The goal for each quadrotor is to maneuver towards the tail of the adversary without being exposed to the front of it. Both quadrotors can move in three dimensions with varying velocities and

independently adjust the facing direction from rotational motions to translational motions because of its unique motion property (Wang et al., 2019). This elegant property enables quadrotors to fly near hovering mode with low speed to make full use of the limited game field space. The proposed formulation assumes that both quadrotors have complete information of the game states. However, input uncertainties are added to the state transition function, of which the state transition probabilities are unknown to the quadrotors. Moreover, the existence of the obstacle introduces additional complexity to the formulation that the utility value of the game states is then determined by not only the relative position and motion status of players but also the relationship of the quadrotors with the terrain. To keep the problem manageable, hard constraints on motions are adopted to avoid collisions with the obstacle. Due to the high flexibility of the quadrotors and the low flight speed, the inflicted constraints are reasonable and adequate for the presented game formulation. To achieve a goal of making rational decisions in the studied quadrotor robot game, a simulation-based projected iterative approach is employed in this study. Specifically, the proposed decision scheme is based on a PI structure, consisting of policy evaluation and policy improvement procedures, where the policy evaluation step is approximately done using a simulation-based projected iterative method. The advantage of this method is that the projected approach is calculated in the feature space rather than the sample space, which greatly increases the ability to handle a complex system with a large size of sample states. The iterative approach effectively avoids large-scale matrix inversions. Accompanying these efforts, the proposed decision scheme can properly work in the quadrotor robot game. The main contributions of this work are the following three aspects: (1) A formulation of a velocity varying one-on-one quadrotor robot game problem in the three-dimensional (3D) space is proposed in conjunction with the stochastic state transition dynamics of the quadrotors; (2) An L_2 regulation method is used to reduce the variance of the policy approximation structure, and hence improve the stability of the generated policy; (3) Real flight tests using the latest Quanser QDrone UAVs are conducted, validating the performance of the proposed motion planning algorithm in the real-time environment.

2 Quadrotor robot game formulation

The motion planning problem of the quadrotor robot game is formulated as a stationary discounted total cost minimization problem of finite states, in which the game state is updated based on the first-order Markov model, expressed as

$$\boldsymbol{s}_{k+1} = f(\boldsymbol{s}_k, \boldsymbol{a}_k, \boldsymbol{\omega}_k), \quad k = 1, 2, \dots, \tag{1}$$

where the state $\boldsymbol{s}_k$ is an element of a space S, the action $\boldsymbol{a}_k$ an element of a space A, and the random disturbance $\boldsymbol{\omega}_k$ an element of a space D. S, A, and D are assumed to be finite spaces with countable elements. The player in Eq. (1) always decides to invariably behave at the same state; that is to say, the optimal action at a given state would not change over time. Note that there is no fixed time limit in a stationary problem; thus, it is recognized as an infinite-horizon problem. In this case, if the player does not ever satisfy the goal criteria, the utility with additive and undiscounted costs will generally be infinite, and then it turns out to be difficult to find the optimal policy correlated with the lowest infinite utility value (Russell and Norvig, 2010). Therefore, a discount factor $\gamma \in (0, 1)$ for the cost per stage is introduced, and the cost functions are bounded. The objective is then to find a stationary policy μ, where $\mu(\boldsymbol{s}_k) \in A(\boldsymbol{s}_k)$ for all $\boldsymbol{s}_k \in S$ $(k = 1, 2, \dots)$, which minimizes the total cost function as

$$J_\mu(\boldsymbol{s}_i) = \lim_{N\to\infty} \mathbb{E}\left\{\sum_{k=0}^{N-1} \gamma^k g(\boldsymbol{s}_k, \boldsymbol{s}_{k+1}) \mid \boldsymbol{s}_0 = \boldsymbol{s}_i\right\}, \tag{2}$$

where $g(\boldsymbol{s}_k, \boldsymbol{s}_{k+1})$ is the function of cost per stage. By applying DP mapping to J_μ, we can obtain the total cost function in the Bellman equation as

$$\begin{aligned}(T_\mu J_\mu)(\boldsymbol{s}_i) &= \mathbb{E}\left\{g(\boldsymbol{s}_i, \boldsymbol{s}_j) + \gamma J_\mu(f(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{\omega}_i))\right\} \\ &= \sum_{j=0}^{n} p_{ij}(\mu(\boldsymbol{s}_i))(g(\boldsymbol{s}_i, \boldsymbol{s}_j) + \gamma J_\mu(\boldsymbol{s}_j)),\end{aligned} \tag{3}$$

where p_{ij} $(i, j = 1, 2, \dots, n)$ are the state transition probabilities, and n is the size of the state space S.

In this quadrotor robot game formulation, the game state $\boldsymbol{s}$ is defined by positions, velocities, and facing angles of two players in blue and red, identified as

$$\boldsymbol{s} = [x_p, y_p, z_p, u_p, v_p, w_p, \psi_p], \quad p = \mathrm{b}, \mathrm{r}, \tag{4}$$

where $\boldsymbol{s}$ is a vector with 14 elements, b represents blue, and r represents red. Constraints are added

to motions to keep the quadrotors within a cuboid game field and to avoid collisions with the obstacle placed on the ground (Fig. 1).

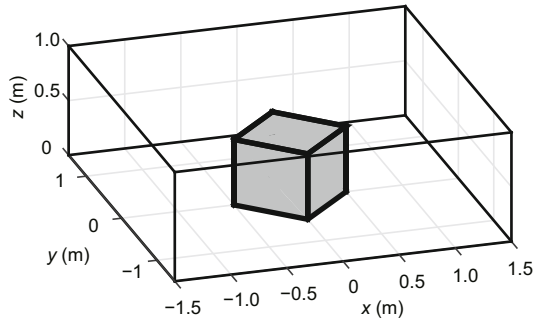

Fig. 1 Game field of the quadrotor robot game

Quadrotors are allowed to fly in three dimensions, and can make pure yaw motion unallied with translational motions. Hence, the action space is set as $A = \{u^+, u^-, v^+, v^-, w^+, w^-, \psi^+, \psi^-, 0\}$, which means that $\boldsymbol{a}_p = [a_p^u, a_p^v, a_p^w, a_p^\psi] \in \{[1, 0, 0, 0], [-1, 0, 0, 0], [0, 1, 0, 0], [0, -1, 0, 0], [0, 0, 1, 0], [0, 0, -1, 0], [0, 0, 0, 1], [0, 0, 0, -1], [0, 0, 0, 0]\}$, $p = \mathrm{b}, \mathrm{r}$. At each step ($\Delta t = 0.25$ s), game states are updated by the state transition function (Algorithm 1). Values of parameters used in the state transition function are listed in Table 1. The obstacle interval and threshold height values for a safe pass over the obstacle are selected based on the size of the quadrotor. In the studied game formulation, $\omega^u, \omega^v, \omega^w$, and ω^ψ are selected from the finite set $D = \{-0.2, -0.15, 0, 0.15, 0.2\}$ in accordance with a probability P_ω. However, the actual distribution of P_ω is unknown to the players. At the beginning, an empirical priori probability

Table 1 Parameters of the state transition function

Parameter	Value	Unit	Parameter	Value	Unit
$\dot{u}_\mathrm{b}$	0.2	m/s^2	$\dot{u}_\mathrm{r}$	0.2	m/s^2
$\dot{v}_\mathrm{b}$	0.2	m/s^2	$\dot{v}_\mathrm{r}$	0.2	m/s^2
$\dot{w}_\mathrm{b}$	0.1	m/s^2	$\dot{w}_\mathrm{r}$	0.1	m/s^2
$\dot{\psi}_\mathrm{b}$	$\pi/9$	rad/s	$\dot{\psi}_\mathrm{r}$	$\pi/9$	rad/s
$u_{\mathrm{b}_\mathrm{max}}$	0.2	m/s	$u_{\mathrm{r}_\mathrm{max}}$	0.2	m/s
$v_{\mathrm{b}_\mathrm{max}}$	0.2	m/s	$v_{\mathrm{r}_\mathrm{max}}$	0.2	m/s
$w_{\mathrm{b}_\mathrm{max}}$	0.1	m/s	$w_{\mathrm{r}_\mathrm{max}}$	0.1	m/s
$u_{\mathrm{b}_\mathrm{min}}$	−0.2	m/s	$u_{\mathrm{r}_\mathrm{min}}$	−0.2	m/s
$v_{\mathrm{b}_\mathrm{min}}$	−0.2	m/s	$v_{\mathrm{r}_\mathrm{min}}$	−0.2	m/s
$w_{\mathrm{b}_\mathrm{min}}$	−0.1	m/s	$w_{\mathrm{r}_\mathrm{min}}$	−0.1	m/s
x_max	1.5	m	x_min	−1.5	m
y_max	1.5	m	y_min	−1.5	m
z_max	1.0	m	z_min	0.2	m

Algorithm 1 State transition function $f(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\omega})$

for $p = \mathrm{b}, \mathrm{r}$ **do**
  if x_p in an obstacle interval **and** $z_p < z_\mathrm{safe}$ **then**
    $u_p = 0$
  else
    $u_p = u_p + \dot{u}_p(a_p^u + \omega^u)\Delta t$
    // Update the velocity component in the x axis
    $u_p = \min(\max(u_p, u_{p_\mathrm{min}}), u_{p_\mathrm{max}})$
    // Constraints on u
  end if
  if y_p in an obstacle interval **and** $z_p < z_\mathrm{safe}$ **then**
    $v_p = 0$
  else
    $v_p = v_p + \dot{v}_p(a_p^v + \omega^v)\Delta t$
    // Update the velocity component in the y axis
    $v_p = \min(\max(v_p, v_{p_\mathrm{min}}), v_{p_\mathrm{max}})$
    // Constraints on v
  end if
  $w_p = w_p + \dot{w}_p(a_p^w + \omega^w)\Delta t$
  // Update the velocity component in the z axis
  $w_p = \min(\max(w_p, w_{p_\mathrm{min}}), w_{p_\mathrm{max}})$
  // Constraints on w
  $\psi_p = \psi_p + \dot{\psi}_p(a_p^\psi + \omega^\psi)\Delta t$ // Update the yaw angle
  $x_p = x_p + u_p\Delta t$; $x_p = \min(\max(x_p, x_\mathrm{min}), x_\mathrm{max})$
  $y_p = y_p + v_p\Delta t$; $y_p = \min(\max(y_p, y_\mathrm{min}), y_\mathrm{max})$
  $z_p = z_p + w_p\Delta t$; $z_p = \min(\max(z_p, z_\mathrm{min}), z_\mathrm{max})$
end for

$P_{\omega 0} = \{0.05, 0.10, 0.70, 0.10, 0.05\}$ is adopted. As the game proceeds, P_ω is updated by the frequencies of visited states. The disturbance set D is assumed to be invariant in spite of the changes of P_ω; thus, the size of the state space S remains unchanged throughout the game.

Cost per stage function $g(\boldsymbol{s}_i, \boldsymbol{s}_j)$ consists of three contributions: orientation, projected range, and terrain. Fig. 2 illustrates the projected geometric relationship of two players on the x-y plane. Since the quadrotors fly near hover mode, orientation angles are thus defined on the x-y plane. The cost function

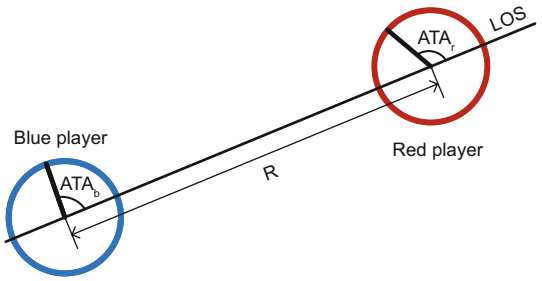

Fig. 2 Projected relative geometry of two players

References to color refer to the online version of this figure

contribution of orientation is calculated as

$$S_A = \frac{|\mathrm{ATA_b}| + |\mathrm{ATA_r}|}{\pi} - 1, \tag{5}$$

where ATA is the antenna train angle (ATA), measured from the nose of a quadrotor to the line of sight (LOS) vector. LOS refers to the vector connecting the centers of mass from the blue quadrotor to the red quadrotor. When the blue quadrotor is directly facing the red quadrotor, $\mathrm{ATA_b} = 0$; if the red quadrotor is directly facing the blue quadrotor, $\mathrm{ATA_r} = \pm\pi$. On the other hand, if the blue quadrotor is directly behind the red quadrotor, then $\mathrm{ATA_b} = \pm\pi$; vice versa for the red quadrotor. Hence, the blue player is going to choose actions that minimize S_A, while the red player prefers actions that maximize S_A. Since both $\mathrm{ATA_b}$ and $\mathrm{ATA_r}$ are within a closed interval $[-\pi, \pi]$, S_A is a bounded cost function. Note that because of the ability to make pure yaw motion independent of translational motions, ATA may be different with the angle between the projected velocity on the x-y plane and the LOS vector, which quite commonly happens in a quadrotor game. With the contribution of orientation, quadrotors may take actions to aim at the tail of the opponent; nonetheless, it may try to fly away from the opponent to prevent exposing its back. Therefore, a contribution of range S_R should attract both players to actively engage in the game. When the projected relative range R is larger than the desired range R_d, S_R pulls the quadrotor towards the opponent; if the projected range value R is smaller than a safety threshold range R_th, S_R pushes the quadrotor away from the opponent to avoid collisions. In practice, a buffer range is added to R_d to prevent overshoot, thus decreasing the risk of collision with the opponent. The cost function of range is formulated as

$$S_R = \begin{cases} k_{r1}(R - R_\mathrm{d}), & R < R_\mathrm{d}, \\ 0, & R_\mathrm{d} \le R \le R_\mathrm{th}, \\ k_{r2}(R - R_\mathrm{th}), & R > R_\mathrm{th}. \end{cases} \tag{6}$$

Note that unlike the air-combat scenario, concepts of pursuing and evading become subtle in the quadrotor game because of the independence of the facing direction and the direction of velocity. Hence, in most circumstances, both players attempt to minimize S_R to achieve the desired range. The cost function combined with orientation and projected range contributions can encourage two players to make maneuvers on the x-y plane to obtain the advantage in the game with a constant clearance from the ground. However, the various terrains offer possibilities for achieving advantaged status rather than solely flying on the x-y plane. For instance, if the opponent is behind the obstacle, the quadrotor can decide either to make a detour while maintaining the height or to fly across the terrain by ascending above the obstacle. Consequently, the contributions of terrain inside and outside the obstacle area are respectively defined as

$$S_H = \begin{cases} 0, & z_\mathrm{safe} < z < z_\mathrm{max}, \\ k_{h0}(z - z_\mathrm{d}), & z < z_\mathrm{safe}, \end{cases} \tag{7}$$

$$S_H = \begin{cases} k_{h1}(z - z_{\mathrm{d}+}), & z_{\mathrm{d}+} < z \le z_\mathrm{max}, \\ 0, & z_{\mathrm{d}-} < z < z_{\mathrm{d}+}, \\ k_{h2}(z - z_{\mathrm{d}-}), & z < z_{\mathrm{d}-}, \end{cases} \tag{8}$$

where $z_{\mathrm{d}+} = z_\mathrm{d} + z_\mathrm{buffer}$ and $z_{\mathrm{d}+} = z_\mathrm{d} - z_\mathrm{buffer}$.

When the quadrotor is outside the obstacle interval or the opponent is flying higher than the obstacle, the desired height z_d is merely the current height value of the opponent; otherwise, z_d equals the safety threshold height to fly over the obstacle. Similar to the contribution of range, both blue and red players attempt to obtain the minimum S_H to achieve the desired height in the game. Moreover, since R and z are within the closed intervals, both S_R and S_H are bounded by the cost functions. Therefore, the costs per stage for blue and red players are defined as

$$\begin{cases} g_\mathrm{b} = S_A + S_R + S_{H_\mathrm{b}}, \\ g_\mathrm{r} = S_A - S_R - S_{H_\mathrm{r}}. \end{cases} \tag{9}$$

The cost per stage function can stimulate the quadrotor to win the game in a greedy manner, where the blue player prefers actions that can minimize g_b, and the red player chooses actions that can maximize g_r. However, the decision strategy for a sequential decision making problem can perform better if the decision is made based on a longer horizon, rather than a single step look ahead greedy method. The presented formulation of the quadrotor robot game provides a platform to evaluate the performance of candidate decision algorithms given the same game environment, including game field setup, quadrotor dynamics, and cost per stage functions. In the subsequent section, the proposed motion planning algorithm based on the ADP approach is introduced. This algorithm will then be adopted by the blue player in Section 4 to evaluate its performance by playing a quadrotor robot game against

the red player driven by the *-minimax algorithm (Ballard, 1983; Hauk et al., 2004).

3 Simulation-based projected policy iteration method

Starting from an initial policy, the PI method generates an improved policy by iteratively executing policy evaluation and policy improvement calculations (Fig. 3). In an ADP-based scheme, the policy evaluation step is approximately done by one feasible approximation architecture. At each iteration,

$$\tilde{J}_\mu(\boldsymbol{s}_i,\boldsymbol{r})=\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_i)\boldsymbol{r}, \tag{10}$$

$$\mu'(\boldsymbol{s}_i)=\arg\min_{a\in A}\sum_{j=1}^{n}p_{ij}(a)(g(\boldsymbol{s}_i,\boldsymbol{s}_j)+\gamma\tilde{J}_\mu(\boldsymbol{s}_j,\boldsymbol{r})), \tag{11}$$

where the utility of the current policy μ at the state $\boldsymbol{s}_i$ is approximated by a linear structure consisting of a feature vector $\boldsymbol{\phi}(\boldsymbol{s}_i)\in\mathbb{R}^q$ and a fit coefficient vector $\boldsymbol{r}\in\mathbb{R}^q$. An improved policy $\mu'(\boldsymbol{s}_i)$ is adopted by the one-step Bellman equation. An optimal approximation of $\boldsymbol{J}_\mu$ can be found by solving a least squares optimization problem, defined as

$$\boldsymbol{r}^*=\arg\min\|\boldsymbol{J}_\mu-\tilde{\boldsymbol{J}}_\mu\|_2^2, \tag{12}$$

where $\tilde{\boldsymbol{J}}_\mu=\boldsymbol{\Phi}\boldsymbol{r}$ and $\boldsymbol{\Phi}=[\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_1),\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_2),\dots,\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_n)]\in\mathbb{R}^{n\times q}$. Assume that $\boldsymbol{\Phi}$ has a rank q. Then $\boldsymbol{r}^*$ can be directly solved by the inversion in batch manner as

$$\boldsymbol{r}^*=(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\hat{\boldsymbol{J}}_\mu, \tag{13}$$

where $\hat{\boldsymbol{J}}_\mu$ is the estimated value of $\boldsymbol{J}_\mu$ based on a sufficient number of sample states. Such an approach has been validated (McGrew, 2008; McGrew et al., 2010; Ma et al., 2014; Fang et al., 2016). However, in the cases where the state space is very large and the number of necessary sample states is also large, calculating Eq. (13) becomes impracticable or extremely time consuming if numerical methods are used. This challenge can be overcome by approximating the utility value $\boldsymbol{J}_\mu$ in the feature space rather than in the state space.

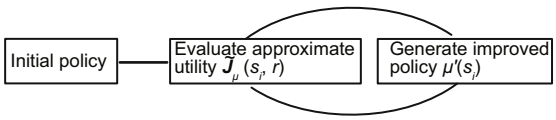

Fig. 3 Policy iteration scheme

To solve this problem, we introduce the projected form of the Bellman equation as

$$\boldsymbol{\Phi}\boldsymbol{r}=\Pi\boldsymbol{T}_\mu(\boldsymbol{\Phi}\boldsymbol{r}), \tag{14}$$

where Π is the projection operator, and $\boldsymbol{T}_\mu$ is the Bellman operator with respect to policy μ. Eq. (14) is an approximation of the Bellman equation $\boldsymbol{J}_\mu=\boldsymbol{T}_\mu\boldsymbol{J}_\mu$ on the feature space. $\boldsymbol{T}_\mu$ is a simple matrix form in Eq. (3), which can be written as

$$\boldsymbol{T}_\mu\boldsymbol{J}_\mu=\boldsymbol{g}+\gamma\boldsymbol{P}\boldsymbol{J}_\mu, \tag{15}$$

where $\boldsymbol{g}=\sum_{j=0}^{n}p_{ij}g(\boldsymbol{s}_i,\boldsymbol{s}_j)$, and $\boldsymbol{P}$ is the matrix with components p_{ij}. The project operator Π denotes a projection onto the feature space, which minimizes a weighted Euclidean norm $\|\cdot\|_{\boldsymbol{\xi}}^2$ with positive weights ξ_i $(i=1,2,\dots,n)$, written as

$$\begin{cases}\Pi\boldsymbol{J}_\mu=\boldsymbol{\Phi}\boldsymbol{r}_\Pi,\\ \boldsymbol{r}_\Pi=\arg\min\limits_{\boldsymbol{r}\in\mathbb{R}^q}\|\boldsymbol{J}_\mu-\boldsymbol{\Phi}\boldsymbol{r}\|_{\boldsymbol{\xi}}^2\\ \quad=\arg\min\limits_{\boldsymbol{r}\in\mathbb{R}^q}\sum\limits_{i=1}^{n}\xi_i(J_\mu(\boldsymbol{s}_i)-\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_i)\boldsymbol{r})^2.\end{cases} \tag{16}$$

Assume that $\boldsymbol{\Phi}$ has a rank q. Then $\boldsymbol{r}_\Pi$ can be solved by taking the gradient of $\|\boldsymbol{J}_\mu-\boldsymbol{\Phi}\boldsymbol{r}\|_{\boldsymbol{\xi}}^2$ to 0, expressed as

$$\boldsymbol{r}_\Pi=(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{J}_\mu. \tag{17}$$

Thus, we have

$$\Pi=\boldsymbol{\Phi}\boldsymbol{r}_\Pi=\boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}, \tag{18}$$

where $\boldsymbol{\Xi}$ is a diagonal matrix with $\Xi_{ii}=\xi_i$, $i=1,2,\dots,n$. Combining the operators Π and $\boldsymbol{T}_\mu$ leads to

$$\Pi\boldsymbol{T}_\mu(\boldsymbol{\Phi}\boldsymbol{r})=\boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}(\boldsymbol{g}+\gamma\boldsymbol{P}\boldsymbol{\Phi}\boldsymbol{r}). \tag{19}$$

It has been proved that if $\boldsymbol{\xi}$ refers to a steady-state probability vector, then $\Pi\boldsymbol{T}_\mu$ is a contraction with respect to $\|\cdot\|_{\boldsymbol{\xi}}$. Therefore, the project Bellman equation has a unique solution and can be successively solved by applying $\Pi\boldsymbol{T}_\mu$, known as the projected value iteration (PVI) method (Bertsekas, 2007). Starting with an arbitrary initial vector $\boldsymbol{\Phi}\boldsymbol{r}_0$, the iteration is formed as

$$\boldsymbol{\Phi}\boldsymbol{r}_{k+1}=\Pi\boldsymbol{T}_\mu(\boldsymbol{\Phi}\boldsymbol{r}_k). \tag{20}$$

By substituting Eq. (19) into Eq. (20), the iteration can be explicitly written as

$$\boldsymbol{r}_{k+1}=\boldsymbol{r}_k-(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{\Phi})^{-1}(\boldsymbol{C}\boldsymbol{r}_k-\boldsymbol{d}), \tag{21}$$

where

$$\begin{cases} \boldsymbol{C} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}(\boldsymbol{I} - \gamma\boldsymbol{P})\boldsymbol{\Phi}, \\ \boldsymbol{d} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{g}. \end{cases} \tag{22}$$

$\boldsymbol{r}_k$ converges to a steady vector $\boldsymbol{r}^*$ after sufficient iterations, while $\boldsymbol{\Phi}\boldsymbol{r}^*$ is the solution of the projected Bellman equation, i.e., the optimal approximation of the utility value. Nevertheless, PVI still depends on matrix products of size n, which again becomes impracticable for a large and complex problem. Furthermore, the probability vector $\boldsymbol{\xi}$ is generally unknown at the beginning. These difficulties can be overcome if we treat the weighted Euclidean norm as an expected value with respect to a probability distribution ($\boldsymbol{\xi}$ here is a steady-state probability vector). Therefore, the projection operation can be viewed as a Monte Carlo simulation, which then allows us to use simulation-based methods to evaluate the approximate utility value.

The idea is to express the least squares problem as an expected value with respect to a probability distribution ($\boldsymbol{\xi}$), and then approximate the expected value by sampling depending on that distribution. For instance, by recalling Eqs. (16) and (17), $\boldsymbol{r}_{\pi}$ can be written in an element form as

$$\boldsymbol{r}_{\pi} = \Big(\sum_{i=1}^{n}\xi_i\boldsymbol{\phi}(\boldsymbol{s}_i)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_i)\Big)^{-1}\sum_{i=1}^{n}\xi_i\boldsymbol{\phi}(\boldsymbol{s}_i)J_{\mu}(\boldsymbol{s}_i). \tag{23}$$

Suppose a sequence of samples of $t = 0, 1, \ldots$, are generated regarding the fixed stationary policy μ according distribution $\boldsymbol{\xi}$. At $t = k$, $\boldsymbol{r}_{\pi}$ can be estimated as

$$\begin{aligned} \hat{\boldsymbol{r}}_{\pi k} &= \left(\tfrac{1}{k+1}\sum_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_t)\right)^{-1}\left(\tfrac{1}{k+1}\sum_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)J_{\mu}(\boldsymbol{s}_t)\right) \\ &= \left(\sum_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_t)\right)^{-1}\left(\sum_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)J_{\mu}(\boldsymbol{s}_t)\right), \end{aligned} \tag{24}$$

where k denotes the number of simulation samples. As the simulation is executed in a discrete manner, the simulation pace can be unified with that of the iteration of $\boldsymbol{r}$. Hence, $\boldsymbol{r}$ is updated at each step when a new simulation sample is generated. Note that $\hat{\boldsymbol{r}}_{\pi}$ is guaranteed to converge to $\boldsymbol{r}_{\pi}$ as the number of simulation samples k increases. To clarify this, we are coping with a finite-state MDP with a fixed policy; thus, the distribution of state $\boldsymbol{s}_i$ is consistent with the long-term frequencies of occurrence of $\boldsymbol{s}_i$ during the simulation,

$$\xi_i = \lim_{k\to\infty}\frac{1}{k+1}\sum_{t=0}^{k}\delta(\boldsymbol{s}_t = \boldsymbol{s}_i), \quad i = 1, 2, \ldots, n. \tag{25}$$

If $\boldsymbol{s}_t = \boldsymbol{s}_i$, $\delta(\boldsymbol{s}_t = \boldsymbol{s}_i) = 1$; otherwise, $\delta(\boldsymbol{s}_t = \boldsymbol{s}_i) = 0$.

By applying the simulation method to PVI, we can reform Eqs. (21) and (22) as

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{G}_k^{-1}(\boldsymbol{C}_k\boldsymbol{r}_k - \boldsymbol{d}_k), \tag{26}$$

where

$$\begin{cases} \boldsymbol{G}_k = \frac{1}{k+1}\sum\limits_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_t), \\ \boldsymbol{C}_k = \frac{1}{k+1}\sum\limits_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)(\boldsymbol{\phi}(\boldsymbol{s}_t) - \gamma\boldsymbol{\phi}(\boldsymbol{s}_{t+1}))^{\mathrm{T}}, \\ \boldsymbol{d}_k = \frac{1}{k+1}\sum\limits_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)g(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}). \end{cases} \tag{27}$$

By substituting Eq. (27) into Eq. (26), the iteration can be written as

$$\begin{aligned} \boldsymbol{r}_{k+1} = &\left(\sum_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_t)\right)^{-1} \\ &\cdot\left(\sum_{t=0}^{k}\boldsymbol{\phi}(\boldsymbol{s}_t)(\gamma\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_{t+1})\boldsymbol{r}_k + g(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}))\right), \end{aligned} \tag{28}$$

where $\boldsymbol{r}_{k+1}$ eventually converges to the solution of the simulation-based projected Bellman equation $\boldsymbol{r}*$. This simulation-based PVI method is known as the "least squares policy evaluation" (LSPE) method. The key benefit of the LSPE method is that it involves only q-dimensional matrix-vector calculations, and q can be significantly smaller than n. Moreover, the distribution vector $\boldsymbol{\xi}$ is not required to be predetermined. We can now derive the simulation-based projected policy iteration method as

$$\tilde{J}_{\mu,k}(\boldsymbol{s}_i, \boldsymbol{r}_k) = \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_i)\boldsymbol{r}_k, \tag{29}$$

$$\mu'_k(\boldsymbol{s}_i) = \arg\min_{a\in A}\sum_{j=1}^{n}p_{ij}(a)(g(\boldsymbol{s}_i, \boldsymbol{s}_j) + \gamma\tilde{J}_{\mu,k}(\boldsymbol{s}_j, \boldsymbol{r}_k)), \tag{30}$$

where $\boldsymbol{r}_k$ is updated by Eq. (28). By successively applying Eqs. (29) and (30), a suboptimal policy μ^* can be found for the stationary discounted total cost minimization problem to be solved.

To obtain the solution of the projected Bellman equation, it is assumed that $\boldsymbol{\Phi}$ has a rank q; otherwise, the matrix $\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{\Phi}$ is singular, and thus it is

difficult to calculate the inversion. As the complexity of the problem increases, more features are needed to represent the full properties of game states, and it becomes more challenging to guarantee that all features are linearly independent. Suppose there are two correlated feature vectors in $\boldsymbol{\Phi}$, and this may cause a pair of fit coefficients in $\boldsymbol{r}$ to have large values with the opposite sign to cancel each other (Hastie et al., 2001). The variance of the approximate structure becomes worse as the correlationship enhances, which may eventually cause the instability of the decision process. A less variant structure can be obtained by introducing L_2 regulation to the least squares term in Eq. (16) with a penalty term on $\boldsymbol{r}$. Thus, the least squares problem can be reformed as

$$\boldsymbol{r}_{\Pi} = \arg\min_{\boldsymbol{r}\in\mathbb{R}^q} \|\boldsymbol{J}_{\mu} - \boldsymbol{\Phi}\boldsymbol{r}\|_{\boldsymbol{\xi}}^2 + \lambda\|\boldsymbol{r}\|_{\boldsymbol{\xi}}^2. \tag{31}$$

Hence, $\boldsymbol{r}_{\Pi}$ can be solved as

$$\boldsymbol{r}_{\Pi} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Xi}\boldsymbol{J}_{\mu}, \tag{32}$$

where the first term turns out to be nonsingular, and the existence of the inversion is guaranteed.

Applying the LSPE method along with the L_2 regulation, the synthesized algorithm for making motion decisions in the quadrotor game is listed in Algorithm 2, where $C(\cdot,\cdot)$ is the combination of operators.

Algorithm 2 Motion planning algorithm

$\bar{S} = \varnothing$ // The set of visited states
$\boldsymbol{r}_0 = O_{(1+2q+C(q,2))\times 1}$ // Initial fit coefficients
for each step $k = 1, 2, \ldots$ **do**
  $\boldsymbol{s}_k = [x_{\mathrm{b}}, y_{\mathrm{b}}, z_{\mathrm{b}}, u_{\mathrm{b}}, v_{\mathrm{b}}, w_{\mathrm{b}}, \psi_{\mathrm{b}}, x_{\mathrm{r}}, y_{\mathrm{r}}, z_{\mathrm{r}}, u_{\mathrm{r}}, v_{\mathrm{r}}, w_{\mathrm{r}}, \psi_{\mathrm{r}}]$
  // Current game state
  $\bar{S} = \bar{S} \cup \boldsymbol{s}_k$ // Update the set of visited states
  $\bar{\boldsymbol{\phi}}(\boldsymbol{s}_k) = [\bar{\phi}_1(\boldsymbol{s}_k), \bar{\phi}_2(\boldsymbol{s}_k), \ldots, \bar{\phi}_q(\boldsymbol{s}_k)]$
  // Feature elements
  $\boldsymbol{\phi}(\boldsymbol{s}_k) = [1, \bar{\boldsymbol{\phi}}(\boldsymbol{s}_k), \bar{\boldsymbol{\phi}}^2(\boldsymbol{s}_k), \bar{\phi}_l(\boldsymbol{s}_k)\bar{\phi}_m(\boldsymbol{s}_k)]^{\mathrm{T}}, l \neq m$
  // Augmented feature vector
  if $\boldsymbol{s}_k \notin \bar{S}$ **then**
    $\mu(\boldsymbol{s}_k) = \mu_0(\boldsymbol{s}_k)$ // Initial policy
  else
    $\mu(\boldsymbol{s}_k) = \mu(\boldsymbol{s}_i = \boldsymbol{s}_k | \boldsymbol{s}_i \in S)$
  end if
  $\boldsymbol{s}_{k+1} = f(\boldsymbol{s}_k, \mu(\boldsymbol{s}_k), \boldsymbol{\omega}_k)$ // Update game state
  $\boldsymbol{r}_{k+1} = \left(\sum_{t=0}^{k-1} \boldsymbol{\phi}(\boldsymbol{s}_t)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_t) + \boldsymbol{\phi}(\boldsymbol{s}_k)\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_k) + \lambda\boldsymbol{I}\right)^{-1} \cdot \left(\left(\sum_{t=0}^{k-1} \boldsymbol{\phi}(\boldsymbol{s}_t)\gamma\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_{t+1}) + \boldsymbol{\phi}(\boldsymbol{s}_k)\gamma\boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_{k+1})\right)\boldsymbol{r}_k + \sum_{t=0}^{k-1} \boldsymbol{\phi}(\boldsymbol{s}_t)g(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) + \boldsymbol{\phi}(\boldsymbol{s}_k)g(\boldsymbol{s}_k, \boldsymbol{s}_{k+1})\right)$
  // Update fit coefficients by the LSPE with L_2 regulation
  if k is sufficiently large **then**
    $\tilde{J}_{\mu}(\boldsymbol{s}_i) = \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{s}_i)\boldsymbol{r}_k$ // Policy evaluation
    $\mu(\boldsymbol{s}_i | \boldsymbol{s}_i = \boldsymbol{s}_k) = \arg\min_{a\in A} \sum_{j=1}^{n} p_{ij}(a)(g(\boldsymbol{s}_i, \boldsymbol{s}_j) + \gamma\tilde{J}_{\mu}(\boldsymbol{s}_j, \boldsymbol{r}_k))$ // Policy improvement
  end if
end for

4 Simulation and flight test results

In this study, six scenarios in the simulation are selected to demonstrate the efficacy of the proposed methods. The presented formulation of the one-on-one quadrotor robot game was tested in the simulation. The state transition function considering boundaries of the game field and static obstacles, and the scoring functions consisting of orientation, projected range, and terrain are examined in Scenarios 1–4. Then, the proposed motion planning algorithm is validated in Scenarios 5 and 6. After that, flight experiments are further conducted to confirm the real-time performance of the proposed decision strategy.

4.1 Scenario 1

Direct action commands are first provided to validate the state transition function (Algorithm 1). The blue quadrotor starts at the initial point $(-1.2, 0, 0.4)$ m and approaches the obstacle, while the red quadrotor flies towards the obstacle from the initial position $(-1.2, 0.7, 0.8)$ m. Both quadrotors maintain the invariant flight height. Hard constraints on translational motions are adopted; thus, the quadrotors are expected to neither exceed the scope of the game field, nor impact the obstacle. Fig. 4 illustrates that the quadrotor detours near the obstacle if the flight height is below the safety height (blue), and flies across the obstacle if the flight height is beyond the safety height (red). Both quadrotors stop at the boundaries of the game field.

4.2 Scenario 2

To verify the relationship of the S_A with the orientation angles and projected range, both quadrotors make a complete round pure yaw motion one after the other. Fig. 5 shows that the blue player achieves advantaged status with a lower value of S_A, while

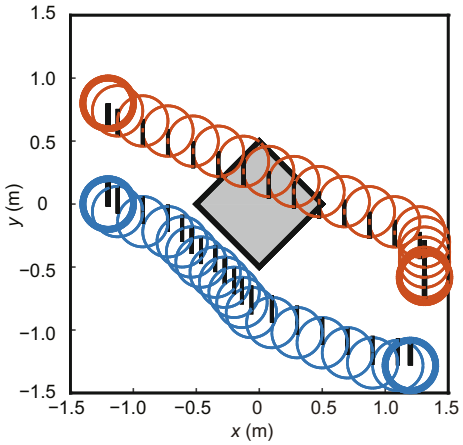

Fig. 4 Simulation results of the state transition function

Quadrotors are represented by two circles distinguished by blue and red. Solid line, starting from the center of the circle, points out the facing direction of the quadrotor. References to color refer to the online version of this figure

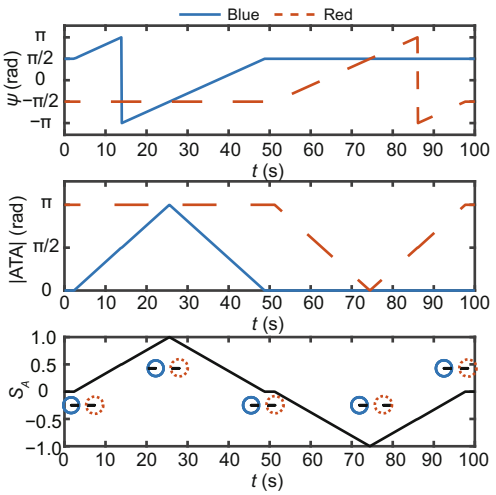

Fig. 5 The relationship of the S_A with orientation angles

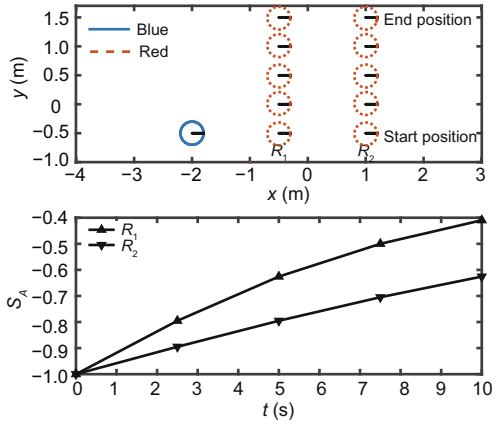

Fig. 6 Influence of the projected range to S_A

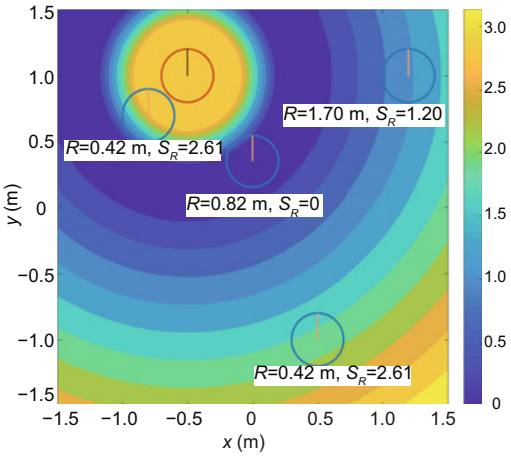

Fig. 7 The contour diagram of S_R

the red player prefers a higher value of S_A. After that, the blue quadrotor stays at the original point, whereas the red quadrotor is commanded to move perpendicular to the facing direction from two starting points with projected ranges R_1 and R_2. Fig. 6 illustrates that it becomes insensitive to change S_A by translational motions when the distance between the quadrotors increases. Hence, it is necessary to introduce the contribution of range to encourage the quadrotors to engage in the game.

4.3 Scenario 3

To reveal the variance of S_R according to the projected range R, the red quadrotor hovers at a fixed point, while the blue quadrotor flies around the red quadrotor at different distances. Fig. 7 shows the contour diagram of the S_R, which indicates that the blue quadrotor is attracted by the red quadrotor if the projected range R is larger than the desired range R_d, whereas if R is smaller than R_d, the blue quadrotor will be pushed away from the red one.

4.4 Scenario 4

To depict the relationship of the S_H and flight height, and the influence of the obstacle, the red quadrotor hovers at a fixed point, while the blue quadrotor is commanded to lift from the ground to the ceiling of the game field outside and inside the obstacle area. Fig. 8 illustrates that the cost function reaches its minimum when the blue and red quadrotors are at the same height outside the obstacle area, whereas the cost function decreases to zero when the

blue quadrotor is ascending inside the obstacle area.

According to the above simulation results, the presented formulation of the quadrotor robot game is qualified to further testify the performance of the proposed decision algorithm in Scenarios 5 and 6.

4.5 Scenario 5

A simplified quadrotor game setup is tested to verify the convergence and fundamental performance of the motion planning algorithm (Algorithm 2). At the beginning, the deployed quadrotors hover at the initial positions as shown in Fig. 9. The red quadrotor remains at the start position during the simulation, while the blue quadrotor begins with an initial policy $\mu(\boldsymbol{s}_i) = \psi^+$, $i = 1, 2, \ldots, n$. The feature elements are chosen as $\bar{\boldsymbol{\phi}} = [1 + \psi_\mathrm{b}, 1 + \mathrm{ATA}_\mathrm{b}, 1 + S_A]$ for demonstration. Hence, the augmented feature vector is expressed as $\boldsymbol{\phi} = [1, 1 + \psi_\mathrm{b}, 1 + \mathrm{ATA}_\mathrm{b}, 1 + S_A, (1 + \psi_\mathrm{b})^2, (1 + \mathrm{ATA}_\mathrm{b})^2, (1 + S_A)^2, (1 + \psi_\mathrm{b})(1 + \mathrm{ATA}_\mathrm{b}), (1+\psi_\mathrm{b})(1+S_A), (1+\mathrm{ATA}_\mathrm{b})(1+S_A)]$. At the early stage of the simulation, γ is set close to zero to suppress the influence of the history value of $\boldsymbol{r}_k$, because the number of simulation samples is largely insufficient to represent the distribution of game states by the frequencies of the visited states. As the number of samples increases, γ gradually approaches one after the distribution tends to stabilize; therefore, the long-term utility value can be learned during the policy evaluation. Fig. 10 illustrates the convergence of the fit coefficient vector $\boldsymbol{r}$, and Fig. 11 shows the convergence of the probability vector of states $\boldsymbol{\xi}$. Fig. 9 demonstrates the performance of the improved policy compared with the initial policy in four cases.

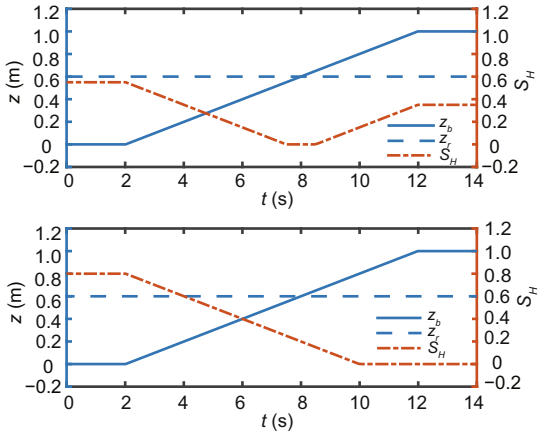

Fig. 8 The relationship of S_H and flight height and the influence of the obstacle

Initial facing directions in the first three cases are chosen as $0, -\pi/2$, and π, respectively. In the fourth case, a random angle is selected as the initial condition. The results validate that the motion policy generated by the proposed algorithm can drive the quadrotor to advantageous states despite the initial positions.

4.6 Scenario 6

A synthesized scenario is finally tested to verify the performance of the proposed motion planning algorithm in a complex environment framed as the presented game formulation defined in Section 2. The blue player starts from the initial position $(-1.0, 1.0, 0.4)$ m with the facing direction of $\pi/2$, while the red player commences at the initial position $(1.0, -1.0, 0.4)$ m with the same facing direction. The initial policy of the blue player is set as

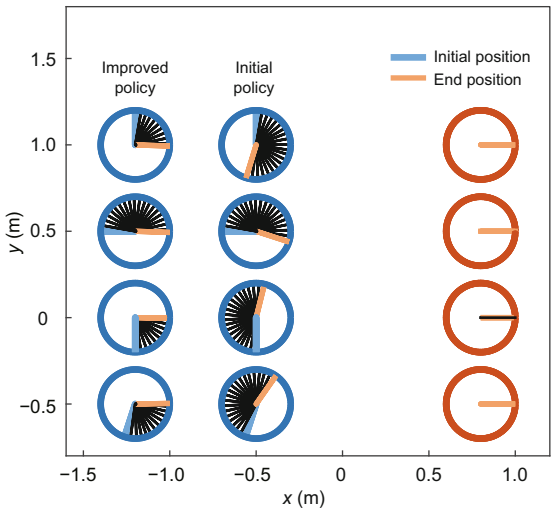

Fig. 9 Fundamental performance of the proposed decision algorithm

References to color refer to the online version of this figure

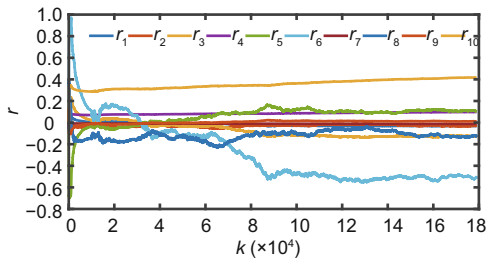

Fig. 10 Convergence process of the fit coefficient vector $\boldsymbol{r}$

References to color refer to the online version of this figure

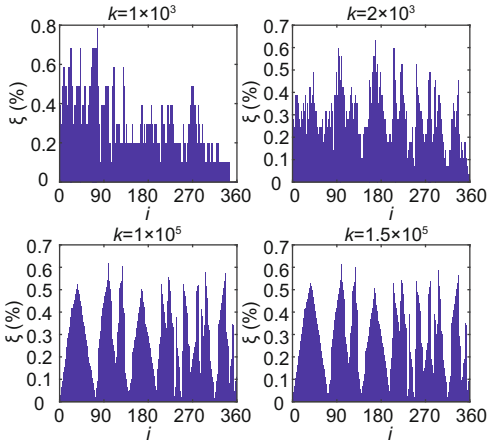

Fig. 11 Convergence process of the probability vector of states

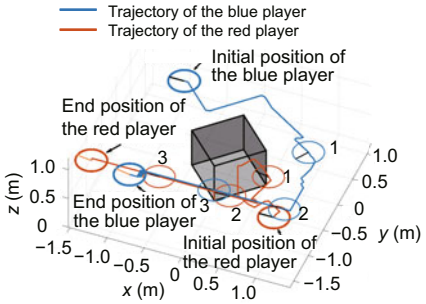

Fig. 12 Simulation trajectory of the quadrotor robot game

References to color refer to the online version of this figure

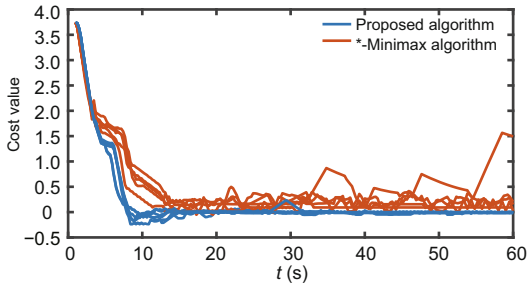

Fig. 13 The performance of the proposed algorithm in comparison to the *-minimax algorithm

References to color refer to the online version of this figure

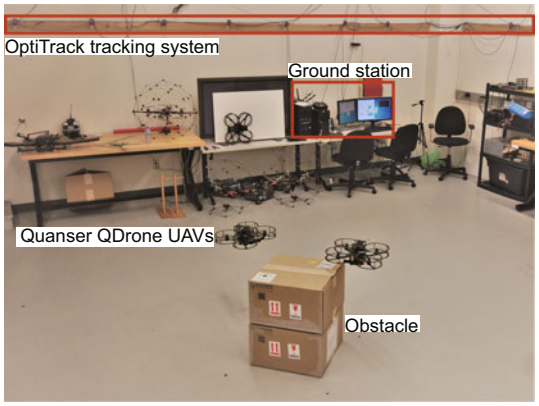

Fig. 14 Flight test conducted in the Networked Autonomous Vehicles (NAV) Lab at the Concordia University

a fixed policy $\mu_0(\boldsymbol{s}_k) = \psi^-$. After sufficient iterations, the improved policy generated by Algorithm 2 is adopted by the blue player to compete with the red player driven by the *-minimax algorithm. Fig. 12 presents one trajectory of the quadrotor robot game. The blue player starts at an adverse state since it is in front of the red player with its back exposed to the opponent. However, as the game proceeds, the blue player gains advantageous status over the red player. To better evaluate the performance of the algorithm, extensive simulations under the same setup are conducted, where the blue player equips with both the proposed and the *-minimax algorithms to play against the red player. The simulation results suggest that compared with the reference algorithm, the proposed algorithm can perform better and more steadily (Fig. 13).

Finally, real flight tests are conducted in the Networked Autonomous Vehicles (NAV) Lab at the Concordia University with two Quanser QDrone UAVs to further validate the real-time performance of the proposed decision strategy. Videos of the experiments are available at https://youtu.be/Omzab2Rt2Ns. Fig. 14 shows the scene of the flight test, which consists of a tracking system, a ground station, an obstacle, and two quadrotor UAVs. Fig. 15 shows the flight trajectory of the quadrotors in one game. The involved quadrotors start at two corners of the game field, separated by the obstacle on the ground. The commands for both UAVs are produced according to the real-time positions and motion states of the quadrotors. Therefore, the motion planning algorithm must be adequately efficient to generate rational decisions; otherwise, delayed commands may cause unexpected reactions to the varying game situations or collisions with the obstacle or the opponent UAV during the game. In the flight test, the blue player employs the proposed motion planning algorithm, while the red player uses the *-minimax algorithm as in the simulation. However, the real motion in the flight test differs from the output of the state transition model in the simulation because of the intense disturbances

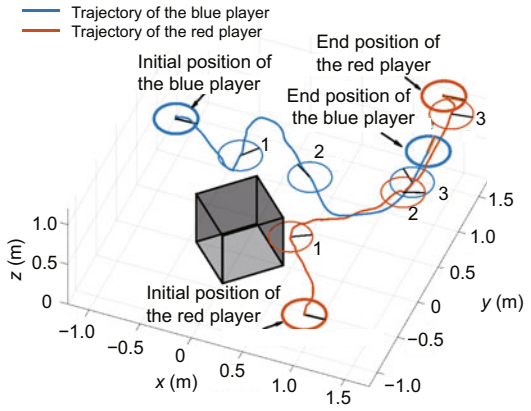

Fig. 15 Flight trajectory of the quadrotor robot game

References to color refer to the online version of this figure

caused by the rolling propellers. It can be observed in Fig. 15 that the blue player can obtain advantaged status in the game in spite of the existence of strong disturbances. The flight tests emphasize the advantage of the proposed algorithm that the generated policy is improved by the history data of the game. As a result, the mismatch of the state transition model of the UAV can be automatically corrected during the flight. Therefore, this dynamic policy can achieve a better performance compared with a fixed policy.

5 Conclusions and future work

The formulation of a velocity varying one-on-one quadrotor robot game motion planning problem has been proposed in this study. Under this formulation, a new simulation-based quadrotor motion planning algorithm using the projected policy iteration method has been proposed, and its effectiveness has been verified by extensive simulation and experimental tests. In future work, a feature refining method is to be developed in addition to the employed L_2 regulation method. Based on this, an upgraded motion planning algorithm is then to be formed to improve the capability of making rational decisions in a more sophisticated environment.

References

Ballard BW, 1983. The *-minimax search procedure for trees containing chance nodes. *Artif Intell*, 21(3):327-350. https://doi.org/10.1016/S0004-3702(83)80015-0

Bellman R, 1952. On the theory of dynamic programming. *Proc Nat Acad Sci*, 38(8):716-719. https://doi.org/10.1073/pnas.38.8.716

Bertsekas DP, 1971. Dynamic Programming and Optimal Control. Athena Scientific, Belmont, Massachusetts, USA.

Bertsekas DP, 2007. Dynamic Programming and Optimal Control (3rd Ed.) Athena Scientific, Belmont, Massachusetts, USA.

Bertsekas DP, 2011. Temporal difference methods for general projected equations. *IEEE Trans Autom Contr*, 56(9):2128-2139. https://doi.org/10.1109/TAC.2011.2115290

Bertsekas DP, 2012. Incremental gradient, subgradient, and proximal methods for convex optimization: a survey. In: Suvrit Sra SN, Wright SJ (Eds.), Optimization for Machine Learning. MIT Press, Massachusetts, USA.

Bertsekas DP, 2015. Lambda-policy iteration: a review and a new implementation. https://arxiv.org/abs/1507.01029

Bertsekas DP, Tsitsiklis JN, 2000. Gradient convergence in gradient methods with errors. *SIAM J Optim*, 10(3): 627-642. https://doi.org/10.1137/S1052623497331063

Buşoniu L, Ernst D, de Schutter B, et al., 2010. Online least-squares policy iteration for reinforcement learning control. Proc American Control Conf, p.486-491. https://doi.org/10.1109/ACC.2010.5530856

Efroni Y, Dalal G, Scherrer B, et al., 2018a. Beyond the one step greedy approach in reinforcement learning. https://arxiv.org/abs/1802.03654

Efroni Y, Dalal G, Scherrer B, et al., 2018b. Multiple-step greedy policies in online and approximate reinforcement learning. https://arxiv.org/abs/1805.07956

Fang J, Zhang LM, Fang W, et al., 2016. Approximate dynamic programming for CGF air combat maneuvering decision. 2nd IEEE Int Conf on Computer and Communications, p.1386-1390. https://doi.org/10.1109/CompComm.2016.7924931

Ghamry KA, Dong YQ, Kamel MA, et al., 2016. Real-time autonomous take-off, tracking and landing of UAV on a moving UGV platform. 24th Mediterranean Conf on Control and Automation, p.1236-1241. https://doi.org/10.1109/MED.2016.7535886

Hastie T, Tibshirani R, Friedman J, 2001. The Elements of Statistical Learning. Springer, New York, USA.

Hauk T, Buro M, Schaeffer J, 2004. Rediscovering *-minimax search. Int Conf on Computers and Games, p.35-50. https://doi.org/10.1007/11674399_3

Liu ZX, Zhang YM, Yu X, et al., 2016. Unmanned surface vehicles: an overview of developments and challenges. *Ann Rev Contr*, 41:71-93. https://doi.org/10.1016/j.arcontrol.2016.04.018

Ma YF, Ma XL, Song X, 2014. A case study on air combat decision using approximated dynamic programming. *Math Probl Eng*, 2014:183401. https://doi.org/10.1155/2014/183401

McGrew JS, 2008. Real-Time Maneuvering Decisions for Autonomous Air Combat. MS Thesis, Massachusetts Institute of Technology, Massachusetts, USA.

McGrew JS, How JP, Williams B, et al., 2010. Air-combat strategy using approximate dynamic programming. *J Guid Contr Dynam*, 33(5):1641-1654. https://doi.org/10.2514/1.46815

Powell WB, 2007. Approximate Dynamic Programming: Solving the Curses of Dimensionality. John Wiley & Sons, New Jersey, USA.

Russell SJ, Norvig P, 2010. Artificial Intelligence: a Modern Approach (3rd Ed.). Prentice Hall, New Jersey, USA.

Sharifi F, Chamseddine A, Mahboubi H, et al., 2016. A distributed deployment strategy for a network of cooperative autonomous vehicles. *IEEE Trans Contr Syst Technol*, 23(2):737-745. https://doi.org/10.1109/TCST.2014.2341658

Sutton RS, Barto AG, 1998. Reinforcement Learning: an Introduction. MIT Press, Massachusetts, USA.

Thiery C, Scherrer B, 2010. Least-squares λ policy iteration: bias-variance trade-off in control problems. Proc 27th Int Conf on Machine Learning, p.1071-1078.

Wang B, Zhang YM, 2018. An adaptive fault-tolerant sliding mode control allocation scheme for multirotor helicopter subject to simultaneous actuator faults. *IEEE Trans Ind Electron*, 65(5):4227-4236. https://doi.org/10.1109/TIE.2017.2772153

Wang B, Yu X, Mu LX, et al., 2019. Disturbance observer-based adaptive fault-tolerant control for a quadrotor helicopter subject to parametric uncertainties and external disturbances. *Mech Syst Signal Process*, 120:727-743. https://doi.org/10.1016/j.ymssp.2018.11.001

Yu HZ, 2010. Convergence of least squares temporal difference methods under general conditions. 27th Int Conf on Machine Learning, p.1207-1214.

Yu HZ, 2012. Least squares temporal difference methods: an analysis under general conditions. *SIAM J Contr Optim*, 50(6):3310-3343. https://doi.org/10.1137/100807879

Yuan C, Zhang YM, Liu ZX, 2015. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Can J Forest Res*, 45(7):783-792. https://doi.org/10.1139/cjfr-2014-0347