Li-dong ZHANG, Ban WANG, Zhi-xiang LIU, You-min ZHANG, Jian-liang AI, 2019. Motion planning of a quadrotor robot game using a simulation-based projected policy iteration method. *Frontiers of Information Technology & Electronic Engineering*, 20(4):525-537. https://doi.org/10.1631/FITEE.1800571

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

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

Corresponding author: You-min ZHANG
E-mail: ymzhang@encs.concordia.ca
ORCID: https://orcid.org/0000-0002-9731-5943

Outline

Outline

Abstract

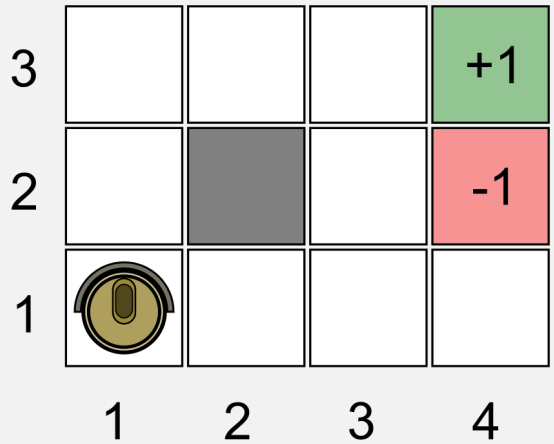
Making rational decisions for sequential decision problems in complex environments has been challenging talented researchers in various fields for d ecades. S uch problems incorporate state transition dynamics, stochastic uncertainties, long-term utilities, and other factors that assemble high barriers including the curse of dimensionality. Recently, the state-of-the-art algorithms in reinforcement learning studies are developed, providing a strong potential to efficiently break the walls 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 three-dimentional space, and an approximate dynamic program-ming approach using 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 can gather advantaged status during the game. Flight experiments, which are conducted in the NAV lab at the Concordia University, have further validated the performance of the proposed decision strategy in real-time environment.

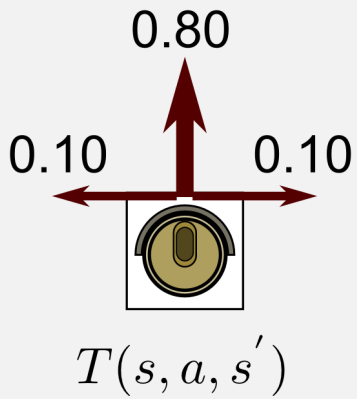
Backgrounds

- Autonomous systems, such as unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs), and unmanned surface vehicles (USVs), are attracting growing interest in scientific, industrial, and military aspects due to their advantageous properties, such as high maneuverability, extended operational range, and improved personnel safety when accomplishing varieties of missions. Challenges remain though, restricted by computational capabilities, sensor reliabilities, communication stabilities, and other limits in both software and hardware.

Backgrounds

- 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 consists of a set of states, a set of actions at each state, a transition model, and a reward/cost function. 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.





Motivations

To make rational decisions for a quadrotor to reach advantaged status and satisfy goal criteria in the game under certain formulation, the problem starts to learn the long-term utilities given the 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.

Main topics

- A one-on-one quadrotor robot game formulation is presented as a platform to examine candidate motion planning strategies.
- A simulation-based projected iterative approach is developed to achieve the goal of making rational decisions in the studied quadrotor robot game.

Outline

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

$$s_{k+1} = f(s_k, a_k, \omega_k), \quad k = 1, 2, \ldots \tag{1}$$

where for all k, the state s_k is an element of a space S, the action a_k an element of a space A, and the random disturbance ω_k an element of a space D. S, A, and D are assumed to be finite spaces with countable elements.

Formulation

The objective is to find a stationary policy μ, where $\mu(s_k) \in A(s_k)$, for all $s_k \in S$, $k = 1, 2, \ldots,$ that minimizes the total cost function as

$$J_\mu(s_i) = \lim_{N\to\infty} \mathbb{E}\left\{ \sum_{k=0}^{N-1} \gamma^k g(s_k, s_{k+1}) \mid s_0 = s_i \right\} \tag{2}$$

where $g(s_k, s_{k+1})$ is the function of cost per stage.
By applying the dynamic programming (DP) mapping to J_μ we can obtain the total cost function in Bellman equation form as

$$\begin{aligned} (T_\mu J_\mu)(s_i) &= \mathbb{E}\left\{ g(s_i, s_j) + \gamma J_\mu(f(s_i, a_i, \omega_i)) \right\} \\ &= \sum_{j=0}^{n} p_{ij}(\mu(s_i))(g(s_i, s_j) + \gamma J_\mu(s_j)) \end{aligned} \tag{3}$$

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

State Transition Function

In the considered quadrotor robot game formulation, the game state s is defined by positions, velocities, and facing angles of two players, the blue and the red, identified by the subscripts as

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

where s is a vector of 14 elements. Constraints are added on motions to keep the quadrotors within a cuboid game field and to avoid collisions with the obstacle placed in the field as shown in Fig. 1.

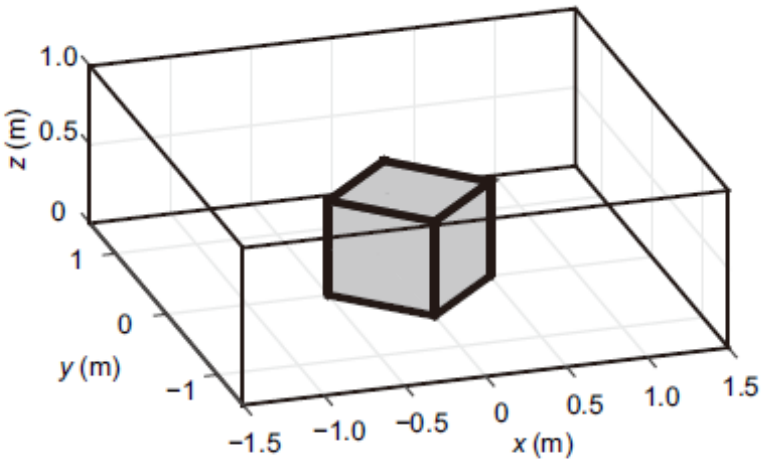


Fig. 1 Game field of the quadrotor robot game

The deployed quadrotor robot: quanser QDrone.

State transition function

The quadrotors are allowed to fly in three dimensions, and can make pure yaw motion unallied with translational motions. Hence, the action set is defined as $A = \{u^+, u^-, v^+, v^-, w^+, w^-, \psi^+, \psi^-, 0\}$, which is equivalent to the mathematical form $a_p = [a_p{}^u,\ a^v{}_p,\ a_{pp}{}^{\psi},\ a\,] \in \{[1,\ 0,\ 0,\ 0],\ [-1,\ 0,\ 0,\ 0],\ [0,\ 1,\ 0,\ 0],\ [0,\ -1,\ 0,\ 0],\ [0,\ 0,\ 1,\ 0],[0,\ 0,\ -1,\ 0],\ [0,\ 0,\ 0,\ 1],\ [0,\ 0,\ 0,\ -1],\ [0,\ 0,\ 0,\ 0]\}$, $p = \mathrm{b}, \mathrm{r}$. At each step ($\Delta t = 0.25$ s), game states are updated by the state transition function (Algorithm 1).

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

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

State transition function

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

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

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

Cost functions

The cost per stage function $g(s_i, s_j)$ consists of three contributions in respect to orientation, projected range, and terrain.

Fig. 3 illustrates the projected geometric relationship of two players in the x- y plane, where ATA_p, $p = \mathrm{b}, \mathrm{r}$ is the antenna train angle (ATA), which is measured from the nose of the 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 straightly facing to the red quadrotor, $\mathrm{ATA_b} = 0$, whereas if the red quadrotor is directly facing to the blue quadrotor, $\mathrm{ATA_r} = \pm\pi$. On the other hand, if the blue quadrotor is directly behind the red quadrotor, $\mathrm{ATA_b} = \pm\pi$, and vice versa for the red quadrotor.

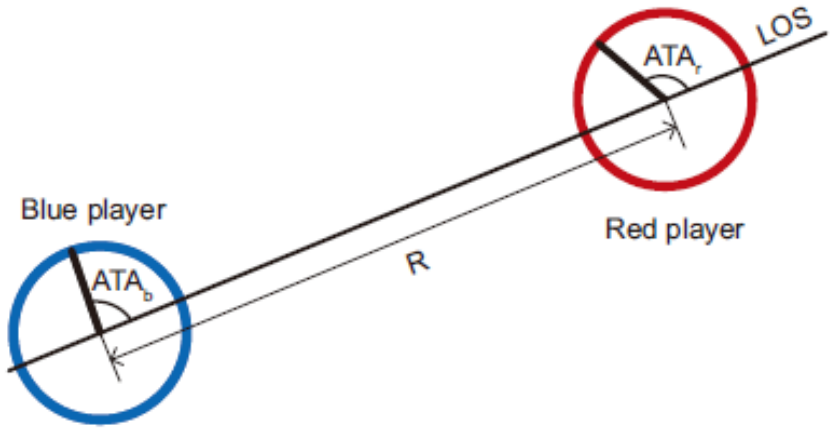


Fig. 2 Projected relative geometry of two players

Cost functions – orientation

Since the quadrotors fly near hover mode, orientation angles are thus defined in the x-y plane. The cost function contribution of orientation is expressed as

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

Hence, the blue player is going to choose actions that minimize S_A, while the red player prefers actions that maximize S_A.

Cost functions – orientation

To verify the relation of S_A with orientation angles, both quadrotors firstly make a complete round pure yaw motion one after the other. The result as shown in Fig. 4 implies that the blue player achieves advantaged status with a lower value of S_A, while the red player prefers the higher value of S_A.

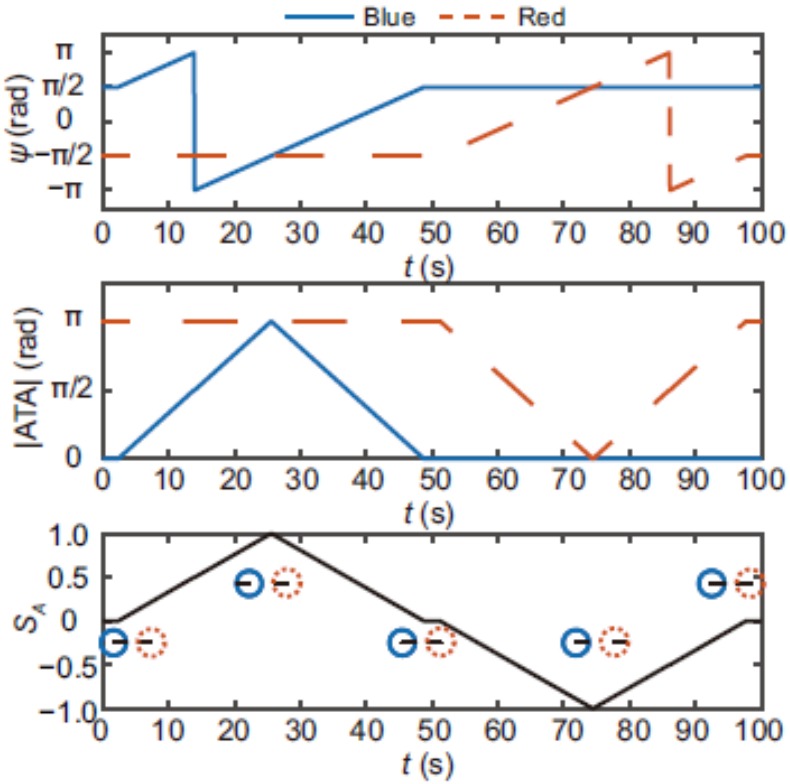


Fig. 5 The relationship of the S_A with orientation angles

Cost functions – range

With the contribution of orientation, the quadrotor may take actions to aim at the tail of the opponent; however, it may try to fly apart from the opponent to prevent to expose its back. Therefore, a contribution of range S_R should be responsible for attracting both players to engage the game actively. When the projected relative range R is larger than the desired range R_{d}, S_R pulls the quadrotor towards the opponent, whereas if the projected range value is less 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

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

Cost functions – range

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 one with different distances. Fig. 5 shows the contour diagram of S_R, which indicates that the blue quadrotor is attracted to the red quadrotor if

the projected range (R) is larger than the desired range (R_d), whereas if R is less than R_d, the blue quadrotor will be pushed away from the red one.

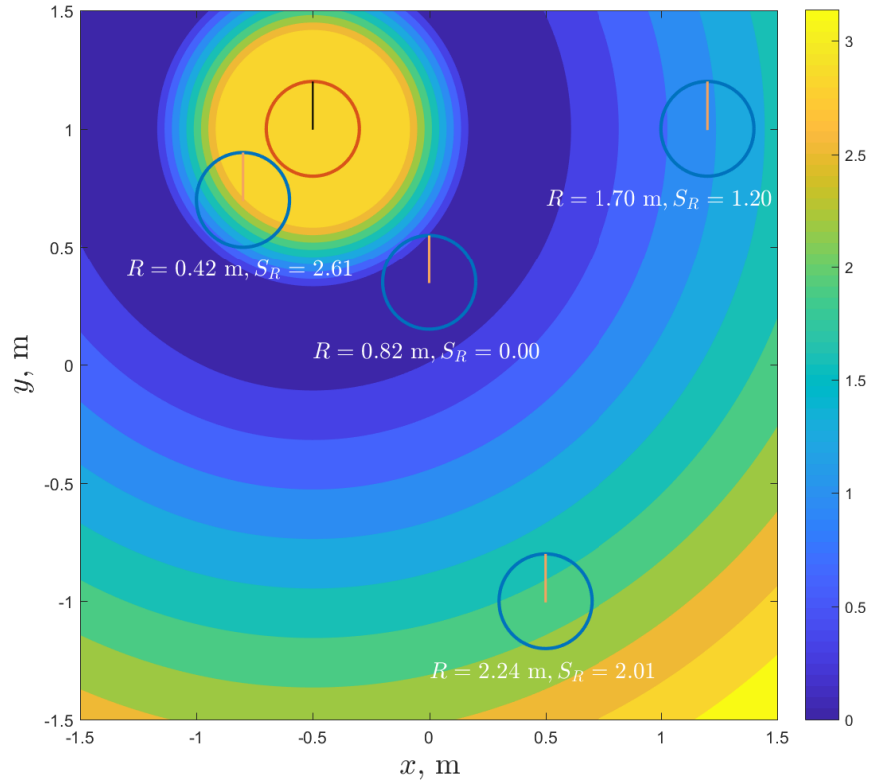


Figure 5: The contour diagram of S_R.

Cost Functions – Terrain

The cost function combined with orientation and projected range contributions can encourage two players to make maneuvers in the $x-y$ plane to get the advantage in the game with a constant clearance from the ground. However, the various terrain makes more possibilities for achieving advantaged status rather than flying in the $x-y$ plane solely. For instance, if the opponent is behind the obstacle, the quadrotor can decide either to make a detour, while maintaining the same height or to fly across the terrain by ascending above the obstacle. Consequently, the contribution of terrain is defined as Eq. (7) (inside obstacle area), or Eq. (8) (outside obstacle area).

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

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

where $z_{d+} = z_d + z_{buffer}$, and $z_{d+} = z_d - z_{buffer}$.

Cost Functions – Terrain

To depict the relation of 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, respectively. Fig. 6 illustrates that the cost function reaches the 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.

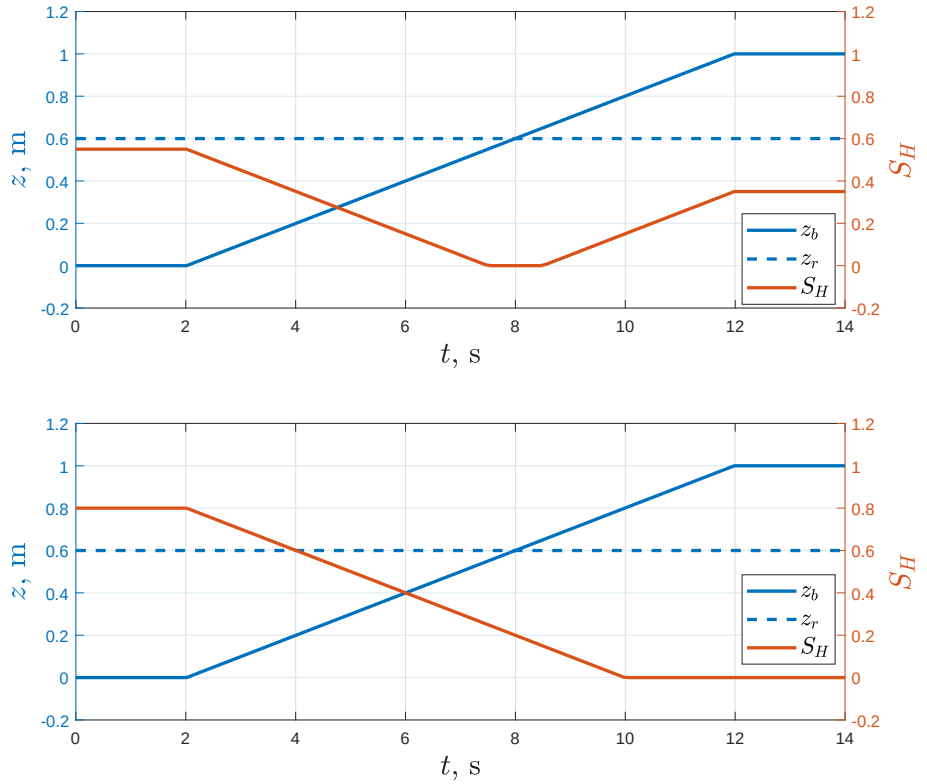


Figure 6: The relation of S_H and flight height, and the influence of the obstacle.

Cost Functionsn

The cost per stage for blue and red players are defined as

$$\begin{aligned} g_b &= S_A + S_R + S_{H_b} \\ g_r &= S_A - S_R - S_{H_r}. \end{aligned} \tag{9}$$

The cost per stage function can stimulate the quadrotor to win the game in a greedy manner, where the blue player prefers actions which can minimize g_b, while the red player chooses actions that 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 looking ahead greedy method.

Outline

Policy Iteration (PI) Scheme

Starting from an initial policy, PI method generates an improved policy by iteratively executing policy evaluation and policy improvement calculations as depicted in Fig. 7. In an ADP-based scheme, the policy evaluation step is done approximately using one feasible approximation architecture. At each iteration,

$$\tilde{J}_{\mu}(s_i, r) = \phi^{\mathsf{T}}(s_i) r \tag{10}$$

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

where the total cost function (the utility value) of the current policy μ at state s_i is approximated by a linear structure consisting of a feature vector $\phi(s_i) \in \mathbb{R}^q$ and a fit coefficient vector $r \in \mathbb{R}^q$, and an improved policy $\mu'(s_i)$ is adopted by one-step Bellman equation.

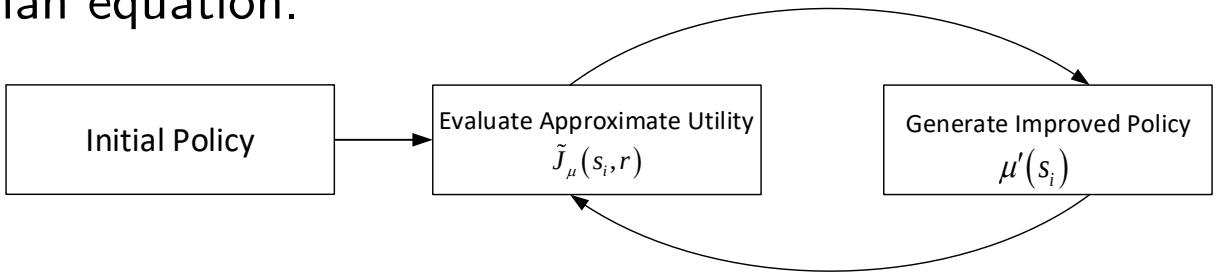


Figure 7: Policy iteration scheme.

Direct Policy Evaluation – Batch Manner

An optimal approximation of $J_\mu$can be found by solving a least squares optimization problem as

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

where $\tilde{J}_\mu = \Phi r$, and $\Phi = [\phi(s_1), \ldots, \phi(s_n)] \in \mathbb{R}^{q \times n}$. Assume that Φ has rank q. Then, r^*can be directly solved by inversion in batch manner as

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

where $\hat{J}_\mu$ is the estimated value of $J_\mu$based on sufficient amount of sample states. Such an approach has been validated in previous studies; however, in the cases where the state space is very large and the number of necessary sample states is a large amount, calculating Eq. (13) becomes impracticable, or extremely time consuming if numerical methods are used. This challenge can be overcome by approximating the utility value $J_\mu$in the feature space rather than in the state space.

Projected Form of Bellman Eqaution

In order to solve this problem, we first introduce the projected form of Bellman equation as

$$\Phi r = \Pi T_\mu(\Phi r) \tag{14}$$

where Π is the projection operator, and T_μ is the Bellman operator with respect to policy μ. Eq. (14) is an approximation of Bellman equation $J_\mu = T_\mu J_\mu$ on the feature space.

The project operator Π denotes a projection onto the feature space which minimizes a weighted Euclidean norm $\|\cdot\|_\xi^2$ with positive weights $\xi_i,\ i = 1, \ldots, n$. It can be written as

$$\begin{aligned} \Pi J_\mu &= \Phi r_\Pi \\ r_\Pi &= \arg\min_{r\in\mathbb{R}^q} \|J_\mu - \Phi r\|_\xi^2 \\ &= \arg\min_{r\in\mathbb{R}^q} \sum_{i=1}^{n} \xi_i (J_\mu(s_i) - \phi^\intercal(s_i) r)^2 \end{aligned} \tag{15}$$

Projected Form of Bellman Eqaution

Assume that Φ has rank q. Then, r_Π can be solved by taking the gradient of $\|J_\mu - \Phi r\|_\xi^2$ to 0 as

$$r_\Pi = (\Phi^T \Xi \Phi)^{-1} \Phi^T \Xi J_\mu \tag{16}$$

thus,

$$\Pi = \Phi r_\Pi = \Phi(\Phi^T \Xi \Phi)^{-1} \Phi^T \Xi \tag{17}$$

where Ξ is a diagonal matrix with $\Xi_{ii} = \xi_i,\ i = 1, \ldots, n$. Combining operator Π and T_μ leads to

$$\Pi T_\mu(\Phi r) = \Phi(\Phi^T \Xi \Phi)^{-1} \Phi^T \Xi (g + \gamma P \Phi r). \tag{18}$$

Projected Value Iteration Method

It has been proved that if ξ refers to a steady-state probability vector, then ΠT_μ is a contraction with respect to $\|\cdot\|_\xi$. Therefore, the project Bellman equation has a unique solution and can be solved by successively applying ΠT_μ, known as projected value iteration (PVI) method. Starting with an arbitrary initial vector Φr_0, the iteration is formed as

$$\Phi r_{k+1} = \Pi T_\mu(\Phi r_k) \tag{19}$$

By substituting Eq. (18) into Eq. (19), the iteration can be explicitly written as

$$r_{k+1} = r_k - (\Phi^T \Xi \Phi)^{-1}(Cr_k - d) \tag{20}$$

where

$$\begin{aligned} C &= \Phi^\intercal \Xi (I - \gamma P)\Phi \\ d &= \Phi^\intercal \Xi g \end{aligned} \tag{21}$$

r_k converges to a steady vector r^* after sufficient iterations, while Φr^* is the solution of the projected Bellman equation, i.e., the optimal approximation of the totoal cost function.

Projected Value Iteration Method

Nevertheless, PVI still depends on matrix products of size n, i.e., $(\Phi^T \Xi \Phi$, $\Phi^\intercal \Xi (I - \gamma P)\Phi$, and $\Phi^\intercal \Xi g)$, which again becomes impracticable for a large and complex problem. Furthermore, the probability vector ξ 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 (ξ 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.

Simulation-Based PVI – Least Squares Policy Evaluation Method

The idea is to express the least squares problem as an expected value with respect to a probability distribution (ξ), and then approximate the expected value by sampling depending on that distribution. For instance, by recalling Eq. (16), r_{Π}can be written in element form as

$$r_{\Pi} = (\sum_{i=1}^{n} \xi_i \phi(s_i)\phi^{\mathsf{T}}(s_i))^{-1} \sum_{i=1}^{n} \xi_i \phi(s_i) J_{\mu}(s_i). \tag{22}$$

Suppose a sequence of samples of $t = 0, 1, \ldots$ are generated regarding the fixed stationary policy μ, according to distribution ξ. At $t = k$, r_{Π} can be estimated as

$$\begin{aligned} \hat{r}_{\Pi k} &= (\tfrac{1}{k+1} \sum_{t=0}^{k} \phi(s_t)\phi^{\mathsf{T}}(s_t))^{-1} (\tfrac{1}{k+1} \sum_{t=0}^{k} \phi(s_t) J_{\mu}(s_t)) \\ &= (\sum_{t=0}^{k} \phi(s_t)\phi^{\mathsf{T}}(s_t))^{-1} (\sum_{t=0}^{k} \phi(s_t) J_{\mu}(s_t)). \end{aligned} \tag{23}$$

Simulation-Based PVI – Least Squares Policy Evaluation Method

Note that $\hat{r}_\Pi$ is guaranteed to converge to r_Π as the number of simulation samples k increases. To clarify this, recall that we are coping with a finite-state Markov decision process (MDP) with a fixed policy, thus the distribution of state s_i is consistent with the long-term frequencies of occurrence of s_i during the simulation, i.e.,

$$\xi_i = \lim_{k\to\infty} \frac{1}{k+1} \sum_{t=0}^{k} \delta(s_t = s_i), \quad i = 1, \ldots, n \tag{24}$$

where $\delta(s_t = s_i) = 1$ if $s_t = s_i$, and $\delta(s_t = s_i) = 0$ if $s_t \neq s_i$.

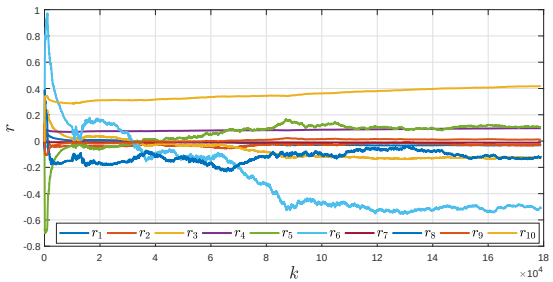


Figure 8: The convergence process of the fit coefficient vector r.

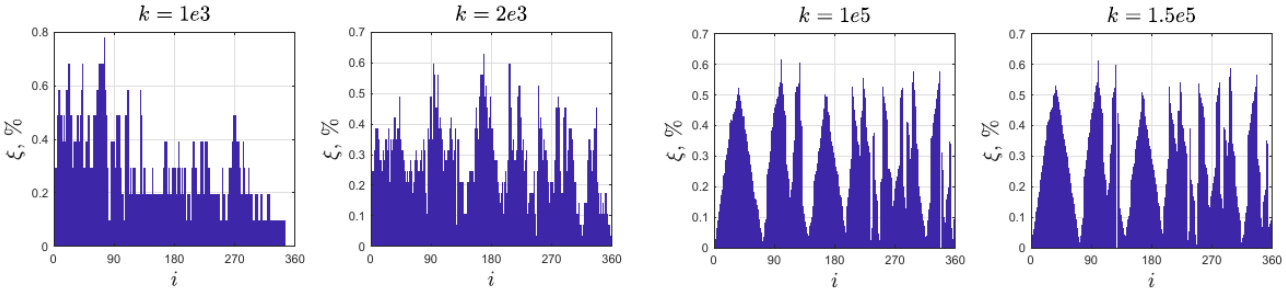


Figure 9: The convergence process of probability vector of states.

Simulation-Based PVI – Least Squares Policy Evaluation Method

Apply the simulation method to PVI, and reform Eqs. (20) and (21) as

$$r_{k+1} = r_k - G_k^{-1}(C_k r_k - d_k) \tag{25}$$

where

$$\begin{aligned} G_k &= \frac{1}{k+1}\sum_{t=0}^{k}\phi(s_t)\phi^{\mathsf{T}}(s_t) \\ C_k &= \frac{1}{k+1}\sum_{t=0}^{k}\phi(s_t)(\phi(s_t)-\gamma\phi(s_{t+1}))^{\mathsf{T}} \\ d_k &= \frac{1}{k+1}\sum_{t=0}^{k}\phi(s_t)g(s_t,s_{t+1}). \end{aligned} \tag{26}$$

Simulation-Based PVI – Least Squares Policy Evaluation Method

By substituting Eq. (26) into Eq. (25), the iteration can be written more compactly as

$$r_{k+1} = \Big(\sum_{t=0}^{k} \phi(s_t)\phi^{\mathsf{T}}(s_t)\Big)^{-1} \Big(\sum_{t=0}^{k} \phi(s_t)\big(\gamma\phi^{\mathsf{T}}(s_{t+1})r_k + g(s_t, s_{t+1})\big)\Big) \tag{27}$$

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

Simulation-Based PVI – Least Squares Policy Evaluation Method

We can now derive the simulation-based projected policy iteration method as

$$\tilde{J}_{\mu,k}(s_i, r_k) = \phi^{\mathsf{T}}(s_i) r_k \tag{28}$$

$$\mu'_k(s_i) = \arg\min_{a \in A} \sum_{j=1}^{n} p_{ij}(a)(g(s_i, s_j) + \gamma \tilde{J}_{\mu,k}(s_j, r_k)) \tag{29}$$

where r_k is updated by Eq. (27). By successively applying Eqs. (28) and (29), a near-optimal policy μ^* can be found for the stationary discounted total cost minimization problem to be solved.

L2 Regulation

To obtain the solution of the projected Bellman equation, it is assumed that Φ has rank q; otherwise, the matrix $\Phi^{\mathrm{T}}\Xi\Phi$ 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 Φ, and this may cause a pair of fit coefficients in r to have large value with the opposite sign to cancel each other. The variance of the approximate structure becomes worse as the correlation increases, which may eventually cause the instability of the decision process. A less variant structure can be obtained by introducing *L2 regulation* to the least squares term in Eq. (15) with a penalty term on r. Thus, the least squares problem can be reformed as

$$r_{\Pi} = \arg\min_{r\in\mathbb{R}^q} \|J_\mu - \Phi r\|_\xi^2 + \lambda\|r\|_\xi^2 \tag{30}$$

Hence, r_{Π} can be solved as

$$r_{\Pi} = (\Phi^T\Xi\Phi + \lambda I)^{-1}\Phi^T\Xi J_\mu \tag{31}$$

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

Simulation-Based Projected Policy Iteration Algorithm

Applying the LSPE method along with the $L2$ regulation, the synthesized algorithm for making motion decisions in the quadrotor game is listed in Algorithm 2, where $C(\cdot,\cdot)$ is the combination operator.

Algorithm 2 The motion planning algorithm

$\bar{S} = \varnothing$ // Set of visited states
$r_0 = O_{(1+2q+C(q,2))\times 1}$ // Initial fit coefficients
for each step $k = 1, 2, \ldots$ **do**
  $s_k = [x_b, y_b, z_b, u_b, v_b, w_b, \psi_b,$
    $x_r, y_r, z_r, u_r, v_r, w_r, \psi_r]$ // Current game state
  $\bar{S} = \bar{S} \cup s_k$ // Update set of visited states
  $\bar{\phi}(s_k) = [\bar{\phi}_1(s_k), \bar{\phi}_2(s_k), \ldots, \bar{\phi}_q(s_k)]$ // Feature elements
  $\phi(s_k) = [1, \bar{\phi}(s_k), \bar{\phi}^2(s_k), \bar{\phi}_l(s_k)\bar{\phi}_m(s_k)]^{\mathrm{T}}, l \neq m$ // Augmented feature vector
  if $s_k \notin \bar{S}$ **then**
    $\mu(s_k) = \mu_0(s_k)$ // Initial policy
  else
    $\mu(s_k) = \mu(s_i = s_k | s_i \in S)$
  end if
  $s_{k+1} = f(s_k, \mu(s_k), \omega_k)$ // Update game state
  $r_{k+1} = (\sum_{t=0}^{k-1} \phi(s_t)\phi^{\mathrm{T}}(s_t) + \phi(s_k)\phi^{\mathrm{T}}(s_k) + \lambda I)^{-1}$
    $((\sum_{t=0}^{k-1} \phi(s_t)\gamma\phi^{\mathrm{T}}(s_{t+1}) + \phi(s_k)\gamma\phi^{\mathrm{T}}(s_{k+1}))r_k$
    $+ \sum_{t=0}^{k-1} \phi(s_t)g(s_t, s_{t+1}) + \phi(s_k)g(s_k, s_{k+1}))$ // Update fit coefficients by LSPE with L2 regulation
  if k is sufficiently large **then**
    $\tilde{J}_\mu(s_i) = \phi^{\mathrm{T}}(s_i)r_k$ // Policy evaluation
    $\mu(s_i | s_i = s_k) = \arg\min_{a\in A} \sum_{j=1}^{n} p_{ij}(a)(g(s_i, s_j)$
      $+\gamma\tilde{J}_\mu(s_j, r_k))$ // Policy improvement
  end if
end for

Outline

Simulation Results

The performance of the proposed motion planning algorithm has been tested under the presented game formulation. The blue player has started from the initial position $[-1.0, 1.0, 0.4]$ m with the facing direction of $\pi/2$, while the red player has commenced at the initial position $[1.0, -1.0, 0.4]$ m with the same facing direction. The initial policy of the blue player has been set to a fixed policy $\mu_0(s_k) = \psi^-$. After sufficient iterations, the improved policy generated by Algorithm 2 has been adopted by the blue player to compete with the red player driven by the *-minimax algorithm. Fig. 10 presents one trajectory of the quadrotor robot game.

The blue player starts at an adverse status 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.

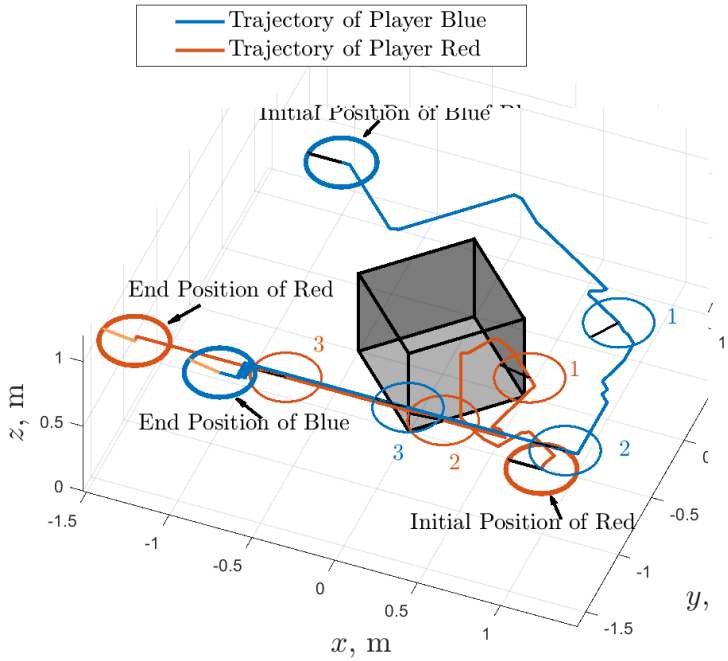


Figure 10: The simulation trajectory of the quadrotor robot game.

Simulation Results

To better evaluate the performance of the algorithm, extensive simulations under the same setup have been conducted, where the blue player equips both the proposed algorithm and the *-minimax algorithm to play against the red player. The simulation results as shown in Fig. 11 suggest that compared with the reference algorithm, the proposed algorithm can perform better and more steadily.

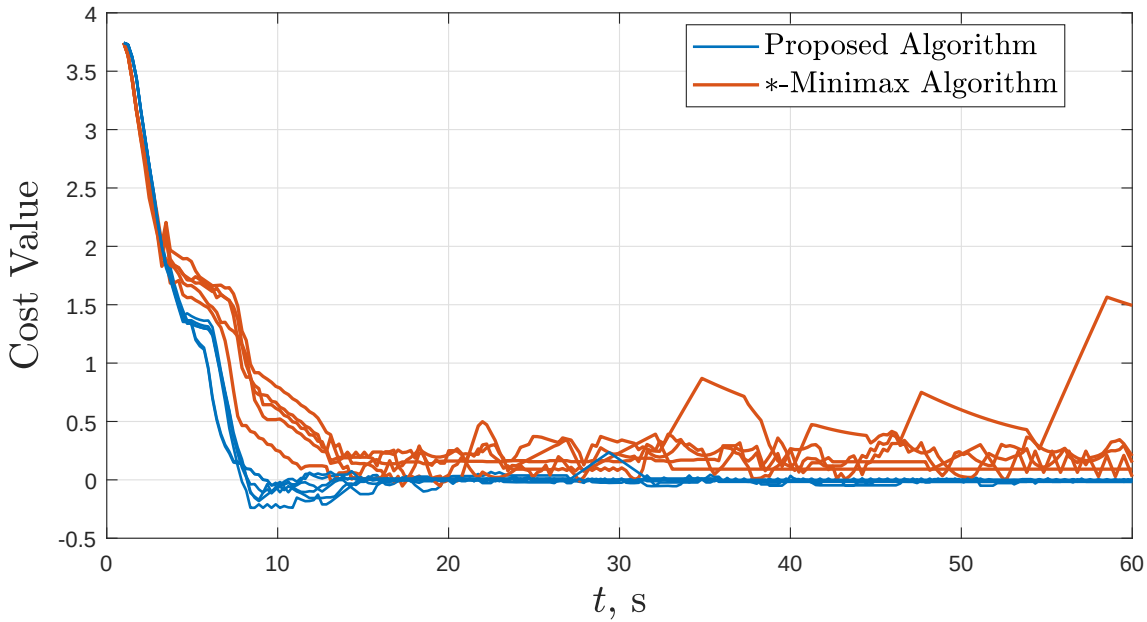


Figure 11: The performance of the proposed algorithm in comparison to *-Minimax algorithm.

Flight Tests

Finally, real flight tests are conducted with two quanser QDrone UAVs available at the NAV Lab of Concordia University to further validate the real-time performance of the proposed decision strategy. The commands for both UAVs are produced according to the real-time position 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.

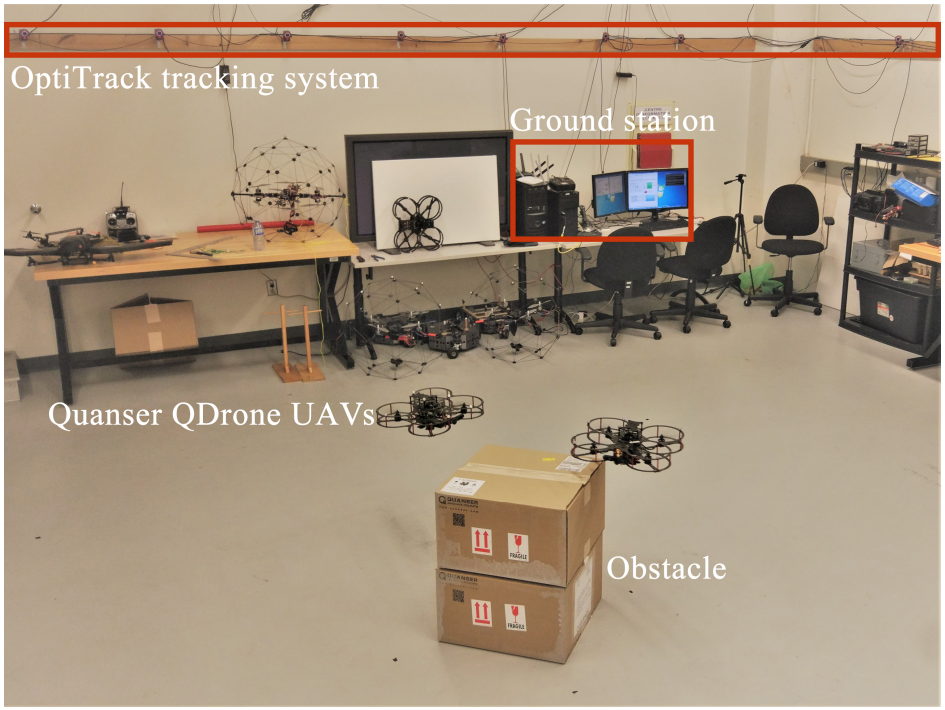


Figure 12: The flight test conducted in the NAV Lab.

Flight Tests

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 due to the intense disturbances caused by the rolling propellers. It can be observed in Fig. 13 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 better performance compared to a fixed policy.

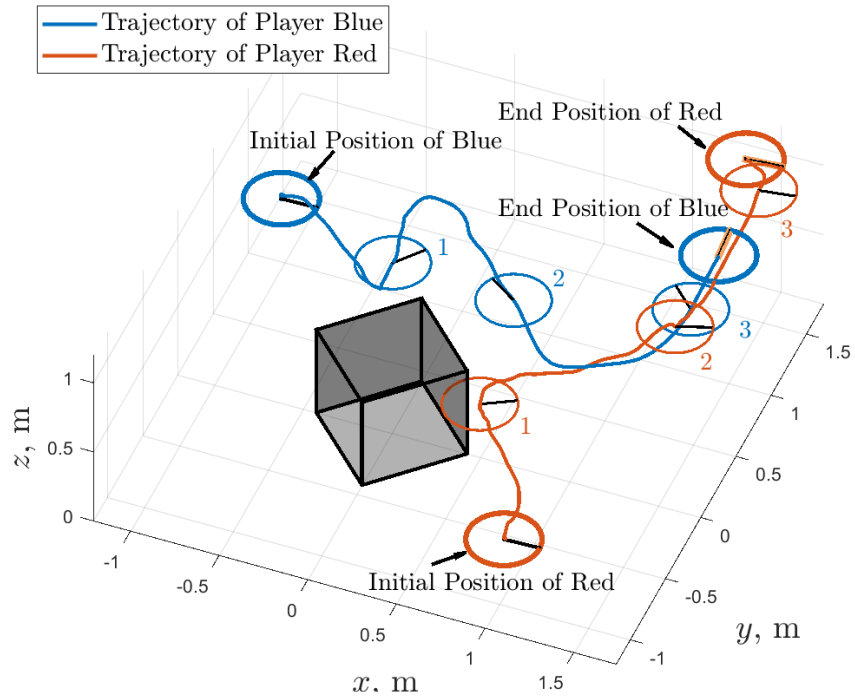


Figure 13: The flight trajectory of the quadrotor robot game.

Outline

Conclusions

The formulation of a velocity varying one-on-one quadrotor robot game motion planning problem has been presented. Under this formulation, a new simulation-based quadrotor motion planning algorithm using projected policy iteration method has been proposed, and its effectiveness has been verified by extensive simulation and experimental tests.

Main Contributions

1. A formulation of a velocity varying one-on-one quadrotor robot game problem in 3D space has been proposed in conjunction with the stochastic state transition dynamics of the quadrotors.
2. $L2$ regulation method has been used to reduce the variance of the policy approximation structure, and hence improves the stability of the generated policy.
3. Real flight tests using QDrone UAVs have been conducted that validate the performance of the proposed motion planning algorithm in the real-time environment.