IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Bat Planner: Aggressive Flying Ball Player

Huan Yu*,1, Jie Tu*,1, Pengqin Wang[2], Zhi Zheng[1],
Kewen Zhang[3], Guodong Lu[1], Fei Gao[4] and Jin Wang†,1

Abstract—In this paper, an aggressive quadrotor Ball plAying sysTem called BAT is proposed, whose goal is to intercept a flying ball and volley it towards a designated target. Aggressive means Bat operates the quadrotor aggressively to intercept balls that are far away and hit them to distant positions in ways that are beyond the reach of existing methods. The trajectory prediction of the ball is achieved by integrating forward the current position and velocity estimates using an extended kalman filter, and implementing cubic interpolation at the time resolution to calculate the continuous gradient for optimization. Facing the challenge of finding feasible hitting actions under extreme circumstances, we propose a two-stage planning approach, including transition point design and hitting primitive generation, with a simplified expression of uncoupled hitting actions. To obtain the best hitting motion, a trajectory optimization method is proposed, which can jointly optimize the hitting terminal states and time cost, considering dynamic feasibility and anticollision constraints. To avoid pathological hitting, a defensive rule constraint and its constraint transcription method are proposed. The largest difference from the existing methods is that Bat Planner can independently decide how to execute more aggressive keyvolleying maneuvers. A large number of simulation and real-world experiments are conducted, which prove the flying ball player can hit arriving balls from different directions and distances to arbitrary targets. To the best of our knowledge, Bat is currently the closest a quadrotor ball player has approached to human ball players' volleying ability.

Index Terms—Aerial systems: applications, motion and path planning, task and motion planning.

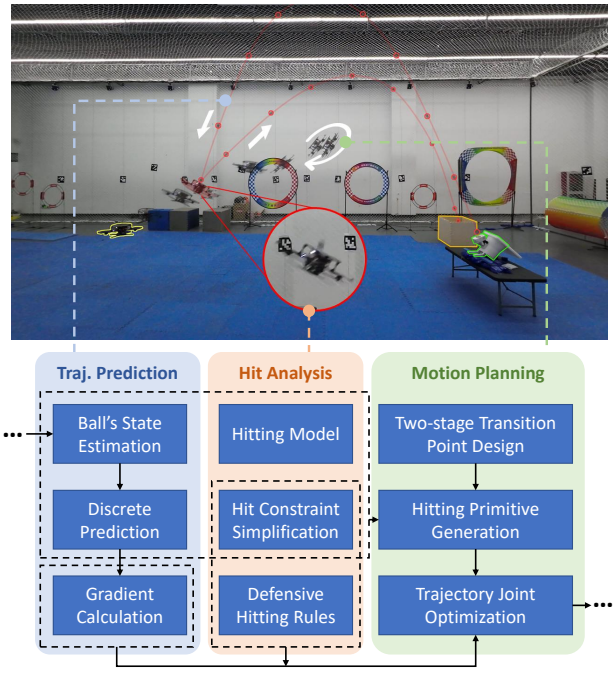

Fig. 1: The overview of Bat Planner. The mission goal is to hit a flying ball into the target box which is marked by orange lines. The green lines mark the pitching machine. The yellow box is the starting position of quadrotor planning. The trajectory of the ball is drawn in red. Traj. is short for Trajectory.

I. Introduction

ROBOTIC ball sports and juggling are popular research topics since they exhibit robots' abilities to perform ornamental and dexterous tasks. Examples include using robots to play table tennis [1], badminton [2], tennis [3], baseball [4], soccer [5][6] and so on. It is extremely difficult for robots to complete such tasks, especially for highly under-actuated quadrotors, because such problems require fast response, adaptive motion planning, accurate prediction and robust control of robots, each of which is a challenge.

This article focuses on a quadrotor playing ball where the quadrotor must volley a ball with an attached racket towards a target. This is like a prototype of a quadrotor tennis match. The main problem of planning how to strike the ball involves various aspects: 1) Deciding when, where and how to hit the ball, and to what target; 2) Motion prediction of the ball; 3) Dynamic analysis of the hitting action; 4) Motion planning of the quadrotor. Existing methods [7][8] solve these four problems separately and piece them together into a system, but have three limitations. First, the height of impact is manually configured and fixed. After predicting the motion of the ball, the hitting position and time of the quadrotor can be uniquely determined, which degenerates the motion planning into a state-to-state planning. However, excessive simplification leads to a lack of flexibility. For example, the quadrotor could have caught the ball at a lower position, but the

This work was supported in part by "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant 2023C01070, the National Natural Science Foundation of China under Grant 52175032, and Robotics Institute of Zhejiang University under Grant K12107 and K11805. (Corresponding author: Jin Wang.) (*These authors contributed to the work equally.)

[1]Huan Yu, Jie Tu, Zhi Zheng, Guodong Lu and Jin Wang are with The State Key Laboratory of Fluid Power and Mechatronic Systems, School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China, also with Robotics Institute of Zhejiang University, Zhejiang University, Hangzhou 310027, China, and also with Robotics Research Center of Yuyao City, Ningbo 315400, China. [2]Pengqin Wang is with The Hong Kong University of Science and Technology, Hong Kong, China. [3]Kewen Zhang is with Zhejiang University of Technology, Hangzhou 310027, China. [4]Fei Gao is with The State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China, and also with Huzhou Institute, Zhejiang University, Huzhou 313000, China. Email: {h.yu, dwjcom}@zju.edu.cn.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

calculated results are dynamically infeasible, resulting in missed opportunities. Second, any two decoupled velocity components of hitting must be fixed, which is a prerequisite for obtaining a trajectory. However, it is impossible to set a parameter to make it applicable to any scene. Third, existing methods need to assume that the highest point of the ball's post-impact trajectory is at a fixed height. Under these assumptions, the unique impact action can be calculated. The intuitive manifestation of the above problems is that the quadrotor can only handle a single scene and hit a ball that falls in a small range around it, but can't do anything about a ball that falls further away. This greatly hinders expansion of the hitting system to a real ball game.

For a real ball game, an excellent "player", which means the quadrotor, should be able to run back and forth to catch the ball and hit it far away from the opponent in the field. These requirements can be summarized into four aspects: 1) Rationality: The hitting action cannot make the trajectory exceed actuator constraints. 2) Aggressiveness: The quadrotor can quickly intercept balls far away from itself, and strike balls toward any target. 3) Adaptability: The quadrotor should have the ability to make independent decisions about where, when and how to hit the ball without manual specification. 4) Lightweight: The whole system must run in real time. Unfortunately, it is difficult, even internally contradictory, to achieve these four aspects at the same time. Rationality is guaranteed by two aspects: the trajectory is dynamically feasible and the hitting action is not pathological, which means that the roll, pitch and angle between the normal vector of the racket and the ball's velocity should be within a reasonable range. This requires that the quadrotor's hitting speed should not be too large, which is against Aggressiveness. Adaptability is difficult to achieve, because the required terminal states and time of hitting are highly coupled, resulting in a highly nonlinear problem which is hard to solve. The expansion of the solution space and nonlinearity lead to a higher complexity of problem solving, which turns Lightweight into a challenge. This is why existing works cannot satisfy these four requirements at the same time.

In this paper, an aggressive quadrotor Ball plAying sysTem called Bat is proposed to meet the above demands. A two-stage hitting primitive is generated to provide a guiding trajectory for the quadrotor. A transition point is designed to provide an opportunity for preparing and adjusting the hitting pose. To obtain the best impact, a terminal-flexible trajectory optimization method is proposed, jointly optimizing the terminal hitting states and time. In order to prevent pathological hitting, defensive rule constraints, terminal direct constraints and their constraint transcription methods are proposed. We also propose a differentiable trajectory prediction method to form a complete system. A large number of contrast and ablation experiments have been completed in the real world and simulation. The results show that Bat Planner can return balls arriving from different directions, different distances and different angles, and impact them to any target. As far as we know, this is the closest quadrotor hitting system to a real human tennis match at present. This article is an extension and advancement of our previous work which requires quadrotor catching flying targets [10]. The main contribution is Bat Planner, which includes:

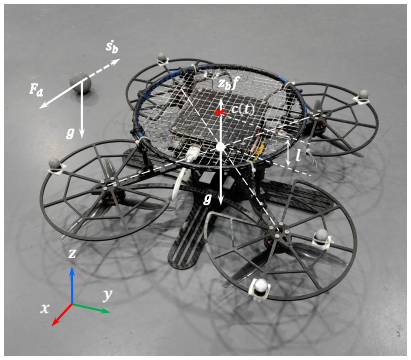

Fig. 2: Quadrotor with attached badminton racket head whose radius is $11cm$. The centroid of the racket $c(t)$ is $\bar{l} = 6cm$ from the paddle plane and on the plane's normal vector.

- State estimation and trajectory prediction of a flying ball, which is in a differentiable form.
- Two-stage hitting primitive generation, which provides guidance for hitting by calculating transition points and a solid initial value for optimization.
- Terminal-flexible trajectory optimization for impact, jointly optimizing the hitting states and time of impact, where defensive rule constraint and direct terminal constraint are proposed to prevent pathological hitting.

II. Dynamics

The quadrotor is modeled as a rigid body with six degrees of freedom: linear translation $p \in \mathbb{R}^3$ and rotation $R \in \mathrm{SE}(3)$. Translation depends on gravitational acceleration $\bar{g}$ and the control input thrust $\tilde{f}$. Rotation takes the body rate $\omega \in \mathbb{R}^3$ as input. The model is [11]

$$\begin{cases} \tau = \tilde{f} R \mathbf{e}_3 / m, \\ \ddot{p} = \tau - \bar{g} \mathbf{e}_3, \\ \dot{R} = R \hat{\omega}, \end{cases} \tag{1}$$

where τ denotes the net thrust. $\mathbf{e}_i$ is the i-th column of $\mathbf{I}_3$, which means Cartesian coordinates in Euclidean space. $\hat{\cdot}$ is the skew-symmetric matrix form of the vector cross product.

The ball's flight is modeled as a point mass under the influence of gravity and aerodynamic drag F_d. We ignore the ball's spin, as in [10]. The ball's motion is expressed as

$$\ddot{\mathbf{s}}_b = \mathbf{g} - K_D \left\| \dot{\mathbf{s}}_b \right\| \dot{\mathbf{s}}_b, \tag{2}$$

where $\mathbf{s}_b$ denotes the ball's position. K_D is drag coefficient. $\|\cdot\|$ refers to the Euclidean norm.

The racket is rigidly mounted on the quadrotor, as shown in Fig. 2. The centroid of the racket can be

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

expressed as $c(t) = p(t) + \bar{l} z_b(t)$, where $\bar{l}$ is the height of the racket from the propeller plane, $z_b(t)$ is the quadrotor attitude in time t. The impact between the racket and the ball is modeled as an impulse acting in the direction of the racket normal $\mathbf{z}_b$. We use the restitution coefficient $\beta = -[(\dot{\mathbf{s}}_b^+)^T \mathbf{z}_b]/[(\dot{\mathbf{s}}_b^-)^T \mathbf{z}_b]$ for $\dot{\mathbf{p}} = \mathbf{0}$ to measure the ball's speed loss when hitting a static racket, where the change of ball's velocity tangential to the racket face is neglected. The ball's post-impact velocity with a moving racket is calculated by

$$\dot{\mathbf{s}}_b^+ = \dot{\mathbf{s}}_b^- - (1-\beta)((\dot{\mathbf{s}}_b^- - \dot{\mathbf{p}})^T \mathbf{z}_b)\mathbf{z}_b. \tag{3}$$

III. Trajectory prediction of the incoming ball

A. State Estimation

The hitting task requires accurate flight state estimation and trajectory prediction of the ball. We use the simplified model [10] for state estimation. The ball's linear equation of motion $s_b(t)$ can be expressed as

$$s(t) = \mathbf{s}_0 - \frac{\dot{\mathbf{s}}_0 + \frac{\mathbf{g}}{K_i}}{K_i} e^{-K_i t - 1} - \frac{\mathbf{g}}{K_i} t, \tag{4a}$$

where the proportionality coefficient K_i is decoupled in $\mathbf{e}_3$, $\mathbf{s}_0$ is the initial ball's position each update. Then an extended kalman filter [14] (EKF) is used to optimize the observation value. The kalman state estimate at time t is

$$\mathbf{s}_b^{cor} = \mathbf{s}_b^t + K_a(\mathbf{Z} - \mathbf{H}\mathbf{s}_b^t), \tag{5}$$

where the observation matrix $\mathbf{H} = [\mathbf{I}_{3\times3}\ \mathbf{0}_{3\times3}]^T$. s_b^t is the transformed state. K_a denotes the kalman gain. $\mathbf{Z}$ denotes the motion capture observation. To make K_i more accurate, we use a nonlinear least squares method (LSM) [15] to minimize the discrete integral of the square difference between $\mathbf{s}_b^{cor}$ and $\mathbf{s}_b$. Then the estimated ball's position and velocity at time t can be calculated.

B. Motion Prediction

For Eq. 2, the drag coefficient K_D is a fixed value theoretically in Euclidean space. Taking the time derivative of the ball's specific mechanical energy $E_b = 1/2\|\dot{\mathbf{s}}_b\|^2 + \mathbf{g}^T \mathbf{s}_b$, yields the specific power extracted from the system by the drag force

$$\dot{E}_b = -K_D \|\dot{\mathbf{s}}_b\|^3, \tag{6}$$

where $\mathbf{s}_b$ can be obtained from estimated $\mathbf{s}_b^{cor}$. Then a measurement $\tilde{K}_D$ can be approximated as [7]

$$\tilde{K}_D = -\frac{(\bar{E}[k] - \bar{E}[k-1])}{\zeta_k(\frac{1}{2}(\bar{v}[k] + \bar{v}[k-1]))^3}, \tag{7}$$

where ζ_k is the step size of a discrete time system. $\bar{E}[k]$ is the estimate of ball's mechanical energy at time t_k. Then the measurements are used to form an estimate of $\hat{K}_D$ using a recursive least squares (RLS) estimator [13].

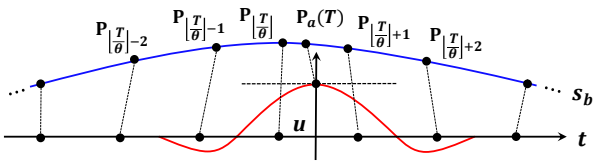

Fig. 3: Process of cubic interpolation. The ball's flight path is drawn in blue. The interpolated curve is drawn in red with the unit weight as ordinate.

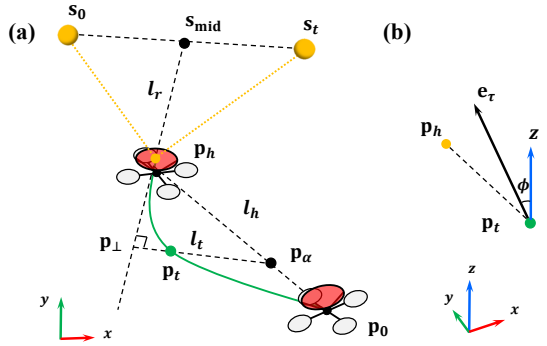

Fig. 4: Design process of the transition point. (a) Calculation of the position $\mathbf{p}_t$. (b) Calculation of the normalized thrust $\mathbf{e}_\tau$.

C. Gradient Calculation

After obtaining the stable drag coefficient K_D online, ball states can be predicted by numerically integrating forward. To provide the gradient for trajectory optimization in Sec. V, we use cubic interpolation [12] to smooth discrete states. A discrete state table is constructed as a vector $\mathbf{G} \in \mathbb{R}_{3\times i}$ with time resolution θ. The interpolation function is designed as

$$W(x) = \begin{cases} 1.5|x|^3 - 2.5|x|^2 + 1, & |x| \le 1, \\ -0.5|x|^3 + 2.5|x|^2 - 4|x| + 2, & 1 < |x| < 2, \\ 0, & |x| > 2, \end{cases} \tag{8}$$

whose first order derivative is continuous. Then we take four adjacent discrete positions $\mathbf{P}_a \in \mathbb{R}_{3\times4}$ uniformly spaced in time for interpolation. The position calculated by cubic interpolation $P_c(T)$ at time T can be expressed as

$$P_c(T) = \mathbf{P}_a \cdot \mathbf{W}, \tag{9}$$

where

$$\mathbf{P}_a = \left[\mathbf{G}(\lfloor \tfrac{T}{\theta} \rfloor - 1) \quad \mathbf{G}(\lfloor \tfrac{T}{\theta} \rfloor) \quad \mathbf{G}(\lfloor \tfrac{T}{\theta} \rfloor + 1) \quad \mathbf{G}(\lfloor \tfrac{T}{\theta} \rfloor + 2)\right], \tag{10a}$$

$$\mathbf{W} = \left[W(1+u) \quad W(u) \quad W(1-u) \quad W(2-u)\right]^T, \tag{10b}$$

$$u = \tfrac{T}{\theta} - \lfloor \tfrac{T}{\theta} \rfloor. \tag{10c}$$

The continous velocity can be obtained in the same way. The interpolation process is visualized in Fig. 3.

IV. Two-stage Hitting Primitive Generation

A. Design of Transition Point

It can be seen from Eq. 3 that the state of the post-impact ball is directly related to the velocity of

the quadrotor, which is required to be not too large. Considering another requirement, when hitting a ball far away from itself, the quadrotor needs to maintain enough thrust and speed to reach the hitting point in time. The above two requirements may be contradictory, bringing difficulty to the trade-off between velocity control and mobility. To solve this problem, we propose a method to divide the hitting process into two stages by calculating a transition point which provides smooth guidance for the trajectory and ensures reasonable velocity and attitude for hitting.

As shown in Fig. 4, the quadrotor moves from the current initial position $\mathbf{p}_0$, passes through the transition point $\mathbf{p}_t$, reaches the hitting position $\mathbf{p}_h$, and hits the ball to the expected target landing position $\mathbf{s}_t$. The transition point can be calculated by the following steps. The initial position $\mathbf{s}_0$ means the intersection of simulated ball state integrating backward and the ground. We can project a ray l_r with endpoint $\mathbf{s}_{mid}$, which is the midpoint of $\mathbf{s}_0$ and $\mathbf{s}_t$. A temporary point $\mathbf{p}_\alpha$ is calculated by offsetting a distance $\alpha\bar{l}_h$ from $\mathbf{p}_0$ towards $\mathbf{p}_h$, where $\bar{l}_h = \|\mathbf{p}_h - \mathbf{p}_0\|_2$. Then $\mathbf{p}_\alpha$ projects perpendicularly to ray $\bar{l}_r$ to obtain $\mathbf{p}_\perp$. Finally, $\mathbf{p}_t$ can be calculated by offsetting distance $\eta\bar{l}_t$ from $\mathbf{p}_\perp$ towards $\mathbf{p}_\alpha$, where $\bar{l}_t = \|\mathbf{p}_\perp - \mathbf{p}_\alpha\|_2$. The quadrotor's thrust $\boldsymbol{\tau}_t$ at the transition point can be determined by using the normalized direction vector $\mathbf{e}_\tau$ and $\tau_{t_z} = \overline{g} + 2(\tau_z - \overline{g})/3$ where τ_z is the thrust component of hitting on the z-axis, as shown in Fig. 4. The transition velocity $\dot{\mathbf{p}}_t$ is calculated from the velocity of hitting $\dot{\mathbf{p}}_h$ as $\dot{\mathbf{p}}_t = \varpi\dot{\mathbf{p}}_h$ where $\varpi = (1/2, 1/2, 1/3)$ is a three-dimensional decoupled coefficient vector. To avoid the computational burden of constructing optimization problems with equality constraints, we generate the trajectories in each stage separately. So far, we have obtained the initial transition states, which will be used for optimization in Sec. V.

B. Impact Dynamic Simplification

Proper simplification can help reduce the decision variables involved in optimization. By decoupling motions in Euclidean space, the required terminal state of the quadrotor at impact can be calculated using Eq. 3 as

$$\mathbf{z}_d(T) = \frac{\dot{\mathbf{s}}_b^- - \dot{\mathbf{s}}_b^+}{\left\|\dot{\mathbf{s}}_b^- - \dot{\mathbf{s}}_b^+\right\|}, \tag{11}$$

$$V_{\perp,des} := (\dot{\mathbf{s}}_r(T))^T\mathbf{z}_d = \frac{1}{1+\beta}(\beta\dot{\mathbf{s}}_b^- + \dot{\mathbf{s}}_b^+)^T\mathbf{z}_d, \tag{12}$$

where $V_{\perp,des}$ is the racket's desired velocity in the direction of the desired hitting attitude $\mathbf{z}_d$. We simplify that the ball is influenced by gravity only after being hit. The ball's pre-impact velocity $\dot{\mathbf{s}}_b^-$ can be obtained by the trajectory prediction. The target $\mathbf{s}_t$ is determined by users. The unknown variables of the quadrotor ball hitting problem include the ball's flying time after being hit $t_{post} \in \mathbb{R}_+$, the ball's post-impact initial velocity $\dot{\mathbf{s}}_b^+$, the quadrotor's hitting velocity $\dot{\mathbf{p}}_h = (\dot{p}_x, \dot{p}_y, \dot{p}_z)$ and its hitting thrust $\boldsymbol{\tau}_h = (\tau_x, \tau_y, \tau_z)$. The quadrotor's hitting position $\mathbf{p}_h = (p_x, p_y, p_z)$ can be analytically calculated from the pre-impact flying time $t_{pre} \in \mathbb{R}_+$. According to Eq. 11 and Eq. 12, we decouple motions in Euclidean space and it can be calculated that

$$\dot{s}_{b_z}^+ = -\frac{\tau_z}{\tau_y}(\dot{s}_{b_y}^- - \dot{s}_{b_y}^+) + \dot{s}_{b_z}^-, \tag{13a}$$

$$\dot{s}_{b_y}^+ = \frac{\xi_y(\tau_y\dot{s}_{b_x}^- - \tau_x\dot{s}_{b_y}^-)}{\xi_x\tau_y - \xi_y\tau_x}, \tag{13b}$$

$$\dot{s}_{b_x}^+ = \frac{(\sqrt{(\dot{s}_{b_z}^- - \delta s_{b_x}^-)^2 + 2\overline{g}(\delta\xi_x - \xi_z)} + \delta\dot{s}_{b_x}^- - \dot{s}_{b_z}^-)\xi_x}{2\delta\xi_x - 2\xi_z}, \tag{13c}$$

$$\delta = \frac{\tau_z}{\tau_x}, \quad \boldsymbol{\xi} = \mathbf{s}_t - \mathbf{p}_h. \tag{13d}$$

For decoupled velocity and thrust when hitting, any component can be calculated from the other two components which are $\dot{p}_y, \dot{p}_z, \tau_x, \tau_z$ in our practice as follows.

$$\dot{p}_x = \frac{(\beta\dot{\mathbf{s}}_b^- + \dot{\mathbf{s}}_b^+)^T \cdot \boldsymbol{\tau}_h}{(1+\beta)\tau_x} - \frac{\dot{p}_y\tau_y + \dot{p}_z\tau_z}{\tau_x}, \tag{14a}$$

$$\tau_y = \frac{\dot{s}_{b_y}^- - \frac{\xi_y}{\xi_x}\dot{s}_{b_x}^+}{\dot{s}_{b_x}^- - \dot{s}_{b_x}^+}\tau_x, \tag{14b}$$

which is simplified as $\tau_y = \mathcal{F}(\tau_x, \tau_z, T_i)$, $\dot{p}_x = \mathcal{P}(\tau_x, \tau_z, \dot{p}_y, \dot{p}_z, T_i)$. These will be used for subsequent primitives and trajectory optimization.

C. Hitting Primitive Generation

In order to obtain a smooth trajectory from the hover state to the hitting terminal state, the minimum jerk criterion is imposed on the optimal trajectory generation problem to minimize the cost function from the start of the maneuver at $t = 0$ to the hitting time $t = T$ as

$$\mathcal{J}_{MP} = \frac{1}{T}\int_0^T \left\|p^{(3)}(t)\right\|^2 dt, \tag{15}$$

Following [9], the optimal state trajectory is generated by employing Pontryagin's minimum principle [14] and introducing a Hamiltonian function as

$$s^*(t) = \begin{bmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{\ddot{\mathbf{p}}_0}{2}t^2 + \dot{\mathbf{p}}_0 t + \mathbf{p}_0 \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + \ddot{\mathbf{p}}_0 + \dot{\mathbf{p}}_0 \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t + \ddot{\mathbf{p}}_0 \end{bmatrix}, \tag{16}$$

with

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = \frac{1}{T^5}\begin{bmatrix} 60T^2 & -360T & 720 \\ -24T^3 & 168T^2 & -360T \\ 3T^4 & -24T^3 & 60T^2 \end{bmatrix}\begin{bmatrix}\Delta\ddot{\mathbf{p}}\\ \Delta\dot{\mathbf{p}}\\ \Delta\mathbf{p}\end{bmatrix}, \tag{17}$$

where $\Delta\mathbf{p}^{(i)} = \mathbf{p}^{(i)}(T) - \mathbf{p}_0^{(i)}$. To sum up, we plan two primitives, which are divided by a transition point. It can be seen from Sec. IV-A where the quadrotor's transition states can be calculated from terminal hitting states. After setting decision variables $\dot{p}_y, \dot{p}_z, \tau_x, \tau_z$, the hitting time T, and using $\mathcal{F}(\cdot)$ and $\mathcal{P}(\cdot)$ in Sec. IV-B, we can obtain the unique hitting primitive $s^*(t)$, which is used for trajectory optimization as an initiation $p_0(t)$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

V. Trajectory Optimization for Hitting

A. Problem Formulation

The basic requirements of the trajectory $p(t)$ include dynamic feasibility and safety. Meanwhile, it is preferable to minimize the control effort. As for playing ball, a quick reaction time is necessary due to the ball's rapid landing speed. For best performance, appropriate constraints should also be imposed. In conclusion, the requirements for optimal hitting can be expressed as a sum over separate trajectory sections i with durations T_i as the following problem.

$$\begin{aligned}\min_{c,T}\ \sum_i \mathcal{J} = \sum_i \Big(&\int_0^{T_i} \|p_i^{(s)}(t)\|^2 \mathrm{d}t + \rho T_i \\ &+ \underbrace{W_D \frac{T_i}{\kappa} \sum_{j=0}^{\kappa} \bar{\omega}_j \mathcal{L}_\mu(\|D(\frac{j}{\kappa}T_i)\|^2 - D_{max})}_{\textit{velocity, acceleration and body rate limit}} \\ &+ \underbrace{W_c \frac{T_i}{\kappa} \sum_{j=0}^{\kappa} \bar{\omega}_j \mathcal{L}_\mu(z_{min}^2 - \|\mathbf{e}_3^T p_i(\frac{j}{\kappa}T_i)\|^2)}_{\textit{ground collision avoidance}}\Big) \qquad (18a)\end{aligned}$$

$$s.t.\quad T_i > 0, \qquad (18b)$$

$$p_i^{(s-1)}(0) = \mathbf{p}_0^{(s-1)}, \qquad (18c)$$

$$p_i^{(s-1)}(T_i) = \mathbf{p}_h^{(s-1)}, \qquad (18d)$$

$$p_i^{(s-1)}(0) = p_{i-1}^{(s-1)}(T_i) = \mathbf{p}_t^{(s-1)}, \qquad (18e)$$

$$H_{min} < H < H_{max}, \qquad (18f)$$

$$\dot{\mathbf{p}}_h < \mathbf{v}_{max}, \qquad (18g)$$

where $D = p_i^{(1)}, p_i^{(2)}, \omega$, Eq. 18a is a form that trades off smoothness, aggressiveness, dynamic feasibility and safety, which is the same as our previous work [10]. In this paper, $i = 1, 2$ due to the existence of transition point. ρ is the weight of time regularization. W_D and W_c are penalty weights. $\mathcal{L}_\mu[x]$ is a logistic penalty function. The constraint Eq. 18b, which requires the strict positiveness of T_i, is eliminated by explicit diffeomorphism in Euclidean spaces by substituting $T_i = e^{t_i}$. Eq. 18c represents the initial state of the quadrotor, where i=1. Eq. 18d and 18e provides a guarantee for hitting constraints, where i=2. Hitting constraints include defensive rule constraint 18f and the direct terminal constraint 18g, only restricting the trajectory in the hitting stage. The transition states are optimized which aims to minimize the cost $\mathcal{J}$ without other constraints. Note that the hitting model is implicitly considered in the hitting constraints.

The polynomial trajectory class $\mathfrak{T}_{\text{MINCO}}$ [16] is adopted for trajectory representation,

$$\begin{aligned}\mathfrak{T}_{\text{MINCO}} = \Big\{ p(t) : [0,T] \mapsto \mathbb{R}^m \ \Big|\ &\mathbf{c} = \mathcal{M}(\mathbf{q},\mathbf{T}), \\ &\mathbf{q} \in \mathbb{R}^{m(M-1)},\ \mathbf{T} \in \mathbb{R}^M_{>0} \Big\},\end{aligned} \qquad (19)$$

where an m-dimensional trajectory $p(t)$ is represented by a piece-wise polynomial of M pieces and $N = 2s - 1$ degree. In this paper, we use $s = 4$ for enough freedom of trajectory. All trajectories in $\mathfrak{T}_{\text{MINCO}}$ have compact parameterization by only the intermediate waypoint vector $\mathbf{q}$ and time vector $\mathbf{T}$ via the linear-complexity formulation $\mathbf{c} = \mathcal{M}(\mathbf{q},\mathbf{T})$. Furthermore, any cost function $\mathcal{J}(\mathbf{q},\mathbf{T}) = F(\mathcal{M}(\mathbf{q},\mathbf{T}),\mathbf{T})$. The mapping also gives a linear-complexity way to compute $\partial\mathcal{J}/\partial\mathbf{q}$ and $\partial\mathcal{J}/\partial\mathbf{T}$ from $\partial F/\partial\mathbf{q}$ and $\partial F/\partial\mathbf{T}$. We achieve an evenly distributed time for the segmented trajectories, so we only need to decide one $T \in \mathbb{R}$ variable to determine $\mathbf{T}$. Then $\mathbf{q}$ can be easily obtained by using $p_0(t)$ and $\mathbf{T}$. In this way, we have completed the reparameterization of the trajectory. After that, a high-level optimizer can optimize the objective efficiently.

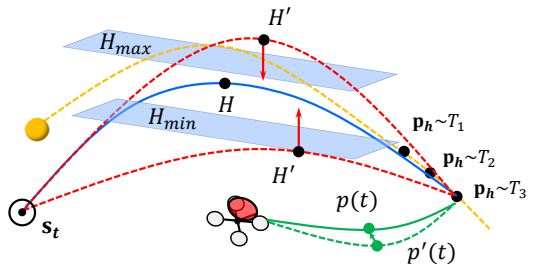

Fig. 5: Trajectory optimization process. The quadrotor's hitting state $(\mathbf{p}_h, \dot{\mathbf{p}}_h, \ddot{\mathbf{p}}_h)$ and time T_i are jointly optimized to obtain the best impact. The ball's pre-impact trajectory is drawn in yellow. The blue post-impact trajectory, whose highest point H meets the defensive rule constraint, is calculated by $\mathbf{p}_h$ and $\mathbf{s}_b^+$.

B. Trajectory Joint Optimization

We transform the problem Eq. 18 into a nonlinear optimization problem as

$$\min_{\mathbf{c},\mathbf{T}}\ \mathcal{J} + W_r\mathcal{J}_r + W_t\mathcal{J}_t, \qquad (20)$$

where $\mathcal{J}_r$ denotes the cost with defensive rule constraint. $\mathcal{J}_t$ denotes the cost with direct terminal constraint. W_r and W_t are the weights.

1) Main cost $\mathcal{J}$: $\mathcal{J}$ includes the control effort, time, dynamic feasibility constraint and collision avoidance constraint. The constraint transformation and gradient calculation methods of $\mathcal{J}$ can be found in our previous work [10] and will not be described here.

2) Defensive rule constraint $\mathcal{J}_r$: We propose the defensive rule constraint to limit the ball's flying area, while preventing the occurrence of pathological hitting motions. *Pathological* hitting motions mainly include the two cases: 1) Roll/pitch angle is too large, leading to too small or even negative post-impact ball's velocity component on z-axis (in Euclidean space). This requires large velocity components on x-axis and y-axis when hitting. 2) Desired $\langle -\dot{\mathbf{s}}_b^-, \dot{\mathbf{p}}_h \rangle$ is large, although the roll/pitch angle is within the safe range. This requires the quadrotor to provide large $\dot{\mathbf{p}}_h$ to meet the expectation of velocity component in $\mathbf{z}_d$ direction. The above pathological situations often make trajectories infeasible. A defensive rule constraint is put forward to prevent them by constructing a virtual roof and a virtual floor, as shown in Fig. 5. It ensure that the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

post-impact ball's flight apex is neither too high nor too low. The cost is represented by

$$\mathcal{J}_r = \mathcal{L}_\mu\left[\mathcal{G}_r(\tau_x, \tau_z, T)\right],$$
$$\mathcal{G}_r(\tau_x, \tau_z, T) = \begin{cases} H_{min} - H, & H < H_{min}, \\ H - H_{max}, & H > H_{max}, \\ 0, & else, \end{cases} \tag{21}$$

where H is the post-impact ball's greatest height in the flight. $\mathcal{L}_\mu[x]$ can be found in [10]. H is calculated by

$$H = \frac{(\dot{s}_{b_x}^+)^2}{2\bar{g}} + p_z. \tag{22}$$

It can be seen that defensive rule constraint essentially limits the decoupling velocity. The gradient can be calculated using Eq. 11, 12, 13 as

$$\begin{aligned} \frac{\partial \mathcal{J}_r}{\partial \mathbf{c}_i} &= \frac{\partial \mathcal{J}_r}{\partial \mathcal{G}_r}\frac{\partial \mathcal{G}_r}{\partial c_i} = 0, \frac{\partial \mathcal{J}_r}{\partial T} = \frac{\partial \mathcal{J}_r}{\partial \mathcal{G}_r}\frac{\partial \mathcal{G}_r}{\partial H}\frac{\partial H}{\partial s_b}\frac{\partial s_b}{\partial T}, \\ \frac{\partial \mathcal{J}_r}{\partial d} &= \frac{\partial \mathcal{J}_r}{\partial \mathcal{G}_r}\frac{\partial \mathcal{G}_r}{\partial H}\frac{\partial H}{\partial d}, d = \tau_x, \tau_z. \end{aligned} \tag{23}$$

3) Direct terminal constraint $\mathcal{J}_t$: Direct terminal constraint is another powerful guarantee against pathological hitting. By constraining the decoupled velocity of hitting $v_j = (v_x, v_y, v_z)$ less than v_{max_j}, the desired hitting attitude $\mathbf{z}_d$ can also be indirectly adjusted using $\mathcal{F}(\cdot)$ and $\mathcal{P}(\cdot)$. The cost $\mathcal{J}_t$ and its gradient can be expressed as follows.

$$\mathcal{J}_t = \sum_j \mathcal{L}_\mu\left[\|v_j\|_2 - v_{max_j}\right], \tag{24a}$$

$$\frac{\partial \mathcal{J}_t}{\partial \mathbf{c}_i} = 0, \quad \frac{\partial \mathcal{J}_t}{\partial v_r} = \frac{\partial \mathcal{J}_{t_r}}{\partial v_r} + \frac{\partial \mathcal{J}_{t_x}}{\partial v_x}\frac{\partial v_x}{\partial v_r}, \tag{24b}$$

$$\frac{\partial \mathcal{J}_t}{\partial d} = \frac{\partial \mathcal{J}_{t_x}}{\partial v_x}\frac{\partial v_x}{\partial d}, \quad \frac{\partial \mathcal{J}_t}{\partial T} = \frac{\partial \mathcal{J}_{t_x}}{\partial v_x}\frac{\partial v_x}{\partial s_b}\frac{\partial s_b}{\partial T}, \tag{24c}$$

$$j = x, y, z, \quad r = y, z, \quad d = \tau_x, \tau_z. \tag{24d}$$

It is worth noting that they are calculated with the terminal states decoupled. We set initial values as $p_0(t)$ in Sec. IV. The problem is then efficiently solved by using a Quasi-Newton method (*i.e.* L-BFGS [17]).

VI. Experiments

A. Experimental Setup

To validate the feasibility and capability of our proposed method in handling various quadrotor ball playing scenarios, we conducted simulations and real-world experiments, comparing success rate (SR) in the quadrotor hitting balls to the target against the state-of-the-art approaches [7] in which the planner was replaced by [8] as a comparison, called Juggle. In addition, ablation study is performed. The key two elements of our approach includes front-end primitive generation in Sec. IV and trajectory optimization in Sec. V. We replace the frontend by [11] and use our backend, then a degraded version of Bat Planner called Bat-Lite is obtained, which is taken as another comparison. More detail results and analysis can be seen in the website[1].

[1]https://github.com/YuHuan1021/bat_planner

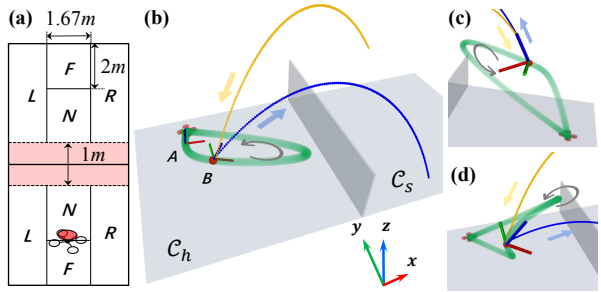

Fig. 6: (a) Simulated court, where dead zone is drawn in red. (b) Real-world experiment in $S.6(LtoF)$ (Bat). A and B represent the transition and hitting position. The yellow and blue lines represent the ball's pre-impact and post-impact flight path. The planned trajectory is visualized in green. (c) Hit using Bat-Lite. (d) Hit using Juggle.

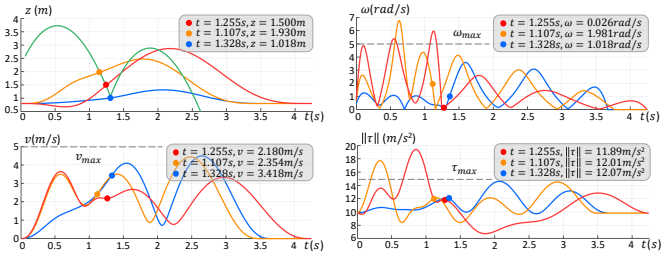

Fig. 7: Trajectories generated by Bat (Blue), Bat-Lite (orange) and Juggle(Red), including hitting height z, body rate ω, velocity v and thrust $\|\tau\|$. Ball's trajectory is visualized as green.

We set $p_{max}^{(1)} = 5m/s$, $p_{max}^{(2)} = 15m/s^2$, $z_{min} = 0.25m$, $\omega_{max} = 5rad/s$ for feasibility and safety constraints, $H_{max} = 6.2m$, $H_{min} = 2.2m$, $v_{max} = (4, 3, 4)m/s$ for defensive rule constraints and direct terminal constraints, and $\phi = 1/2$, $\eta = 2/3$, $\alpha = 1/2$ for calculating transition point. The restitution coefficient β between the ball and the quadrotor is identified offline as approximately 0.7. The required attitude $\mathbf{z}_d$ can be analytically determined when the post-impact ball's velocity is 60° from the horizontal. We set $\tau_z = 1.1\bar{g}$, so that τ_x, τ_y are obtained. The primitives' initial time $T_0 = 0.7s, T_1 = 0.5s$. $\dot{p}_x$ can be calculated by setting $\dot{p}_y = 0, \dot{p}_z = 1m/s$ using Eq.14.

B. Simulation Experiments

To quickly and extensively validate the motion planning module, we conduct simulations and compare planning success rate (PSR) of Bat, Bat-Lite and Juggle. Planning success refers that the planned trajectory is dynamically feasible and satisfies the equation constraints for hitting the ball to a specific point. In other words, assuming there are no real-world errors, such as ball prediction errors, trajectory tracking errors, communication delays etc., the quadrotor will definitely be able to hit the ball to the desired target. A $5m * 10m$ court $\mathcal{C}$ is constructed and divided into two halves, including the hitting half on the quadrotor's side $\mathcal{C}_\text{h}$ and the service court on the opponent's side $\mathcal{C}_\text{s}$, as presented in Fig. 6(a). The quadrotor needs to

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE I: Planning Success Rates of Simulations (%)

Scenario*	NNN	NNF	NNL	NFN	NFF	NFL*	NLN	NLF	NLL	NLR	FNN	FNF	FNL	FFN	FFF	FFL	FLN	FLF
Juggle[7]	24.9	0	7.9	41.9	0	0	33.9	0	9.5	0	22.1	3.9	19.4	42.3	0	0.6	38.2	0.4
Bat-Lite	48.6	42.5	53.6	54.6	17.0	21.6	48.7	30.3	42.4	28.7	25.9	77.1	76.8	77.7	39.6	49.5	51.8	55.0
Bat	66.5	77.1	78.1	82.8	62.5	83.5	70.8	83.0	83.9	80.2	48.4	83.7	81.3	82.1	85.4	89.6	61.8	86.2
Scenario	FLL	FLR	LNN	LNF	LNL	LNR	LFN	LFF	LFL	LFR	LLN	LLF	LLL	LLR	LRN	LRF	LRL	LRR
Juggle[7]	21.7	0.3	23.2	1.9	10.4	12.1	40.1	0	0.2	0.3	36.7	0	12.8	0.1	34.1	0.2	0	16.8
Bat-Lite	61.8	60.6	33.2	53.0	54.9	66.8	63.4	25.9	32.2	35.2	46.4	45.8	51.7	45.1	55.4	44.7	51.9	53.4
Bat	81.0	85.4	53.8	80.7	77.6	81.9	84.3	75.7	79.2	83.8	59.4	84.2	83.1	86.5	68.0	82.9	85.4	85.1

* Scenarios are marked by Serve-Land-Target regions. For example, NFL means the ball is served in N-region of $\mathcal{C}_\mathrm{s}$ and will land in F-region of $\mathcal{C}_\mathrm{h}$ if without hit. The target is set in the center of L-region of $\mathcal{C}_\mathrm{s}$.

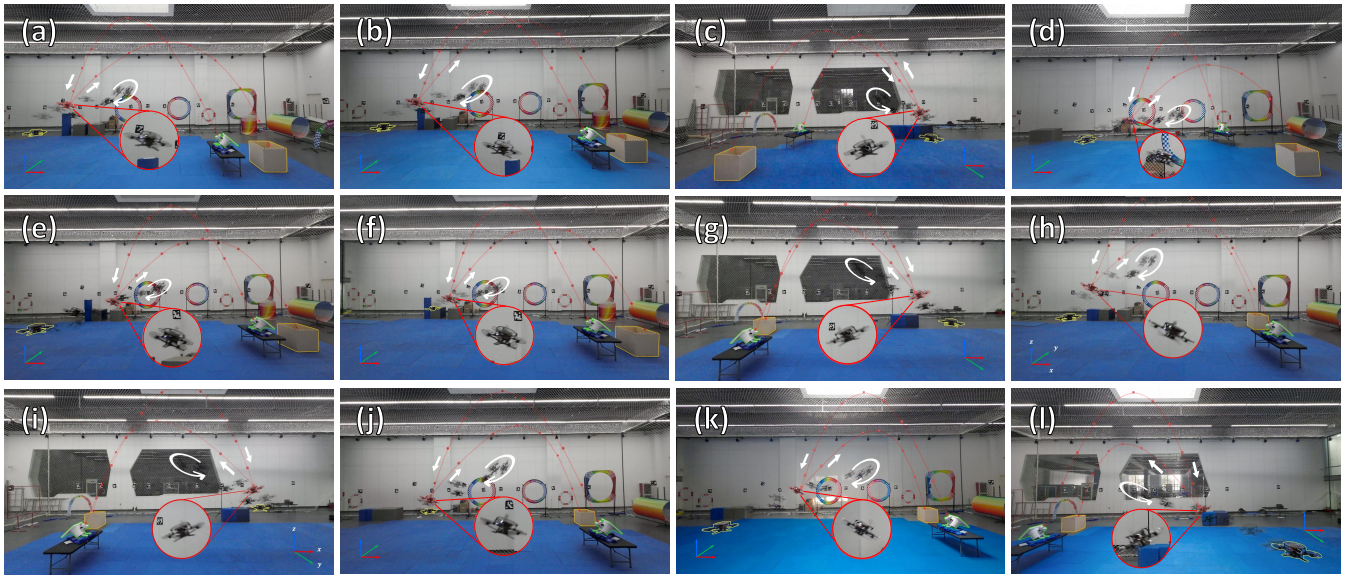

Fig. 8: Real-world experimental process. (a)(b) S.1. (c)(d) S.2. (e)(f) S.3. (g)(h) S.4. (i)(j) S.5. (k)(l) S.6. S.2 to S.6 is symmetric. The images were captured from the experimental video and stitched together. The trajectory of the quadrotor is visualized by the ghost images at $0.3s$ intervals. The image of the moment of hitting the ball is partially enlarged.

start from a hovering position in $\mathcal{C}_\mathrm{h}$, hit the ball from $\mathcal{C}_\mathrm{s}$ to the target in $\mathcal{C}_\mathrm{s}$, and then return to the initial position. Each half is divided into four subareas, N(ear), F(ar), R(ight), L(eft) besides the dead zone. According to different positions of service, landing and target, as well as the elimination of repetition (such as left and right symmetry), 36 scenarios are arranged and combined. 1000 experiments are done in each scenario to verify the capability of the methods. The position and velocity of the service are randomly generated under the conditions of the designated service and the landing subarea. Each target is set as the center of subarea. The ball's motion is simulated using Eq. 2.

The results listed in Tab. I show that the PSR of Bat is far greater than Bat-Lite and Juggle, demonstrating that Bat can handle a wider range of ball hitting scenarios with high aggressiveness. Due to the lack of guidance on transition points, Bat-Lite is more likely to fail than Bat, especially in extreme scenarios when the quadrotor turns through a large attitude change and hits the ball to a faraway target. Nevertheless, Bat-Lite still performs better than Juggle. Juggle loses its adaptability due to the manually determined hitting height and any two velocity components. Bat-Lite and Juggle perform passably in a few scenes, such as down the line shots. The examples contain NFN, where the quadrotor is more readily able to reach the desired velocity than in other scenarios using Juggle, and generate a primitive similar to Bat although without transition point using Bat-Lite.

C. Real World Experiments

To validate the feasibility and capability of the entire system, we conducted real-world experiments. The experiments are completed in a motion capture gym, which integrates Bat, as shown in Fig. 1. A quadrotor with a badminton racket head rigidly mounted is designed, as shown in Fig. 2. The output trajectory is converted into the control command by using the differential flatness output model [18] and executed by an $SE(3)$ cascade PID controller, which uses Hopf fibration [19] to avoid singularities and align the attitude calculation. The quadrotor's state is estimated by an EKF of the pose from Vicon and the IMU data from a PX4 Autopilot. All modules can run below $20ms$ on an NUC with Intel-i5-1135G7 CPU. Approximately covering all simulated scenarios, 6 real scenarios are constructed as:

- S.1: Hit the ball from different distances,
- S.2: Serve the ball from different directions,
- S.3: Hit the ball from different directions,
- S.4: Hit straight & shoot sideways,
- S.5: Hit & shoot sideways to the same side,
- S.6: Hit & shoot sideways to different sides,

where S. is short for Scenario. Serve means the ball is served by a pitching machine. Straight, sideways and side are relative to x-axis. The first three require the quadrotor to hit the ball to the same target. The last three require different targets.

The quadrotor performed 100 ball hitting attempts in each scene, allowing for a comprehensive evaluation of its performance. The Bat's SR of the ball landing within $0.5m$ and $1m$ of the target after being hit by the quadrotor are drawn as Fig. 9. The PSRs are used to compare with Bat-Lite and Juggle. Note that we specifically optimized Juggle's parameters to the best of our ability for each scenario, which greatly improves the PSR. If we were to use the original parameters, Juggle's PSR would be less than 5% for all scenarios. The results demonstrate that our method can handle a wider range of ball hitting scenarios. The main reason for successful planning but failed hitting to the target region is the cumulative error in trajectory prediction. In our experiments, the maximum prediction error for the ball's launch and descent below a height of $1m$ is $(15, 5, 25)cm$. Re-planning is not performed because even minor updates in the terminal state could lead to drastic deformations of the trajectories, which are unfavorable for control. Snapshots of 12 experiments in 6 scenarios are shown as Fig. 8.

The quality of the trajectories is also evaluated. Fig. 7 provides an example of trajectories planned by Bat, Bat-Lite and Juggle for the same hit in S.6. Note that the dynamically feasible constraints of Bat-Lite and Juggle are relaxed to avoid failed planning. We can observe that the dynamics of Bat are well bounded and change more gradually. On the contrary, Juggle and Bat-Lite have more oscillations and exceed feasible constraints. One intuitive visualization is Fig. 6. For Juggle, the quadrotor dives first and then accelerates up suddenly which is control-unfriendly. The main reason is the improper manual assignment of terminal states. In fact, it is impossible to set only one group of parameters to fit all scenarios for Juggle. As for Bat-Lite, the lack of transition points may lead to poor initial solutions, causing the optimization to get trapped in local minima.

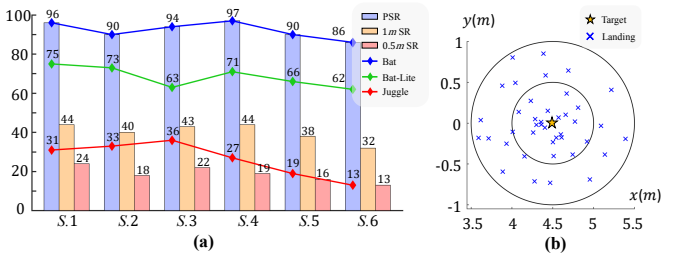

Fig. 9: Results of real-world experiments. (a) Success rates for 6 scenarios, including Bat's PSR, hitting SR $0.5m$-near ($0.5m$ SR) and $1m$-near ($1m$ SR) from the target. The polylines represent the PSR of Bat, Bat-Lite and Juggle. (b) The hit balls' landing points for 100 attempts in S.1.

VII. Conclusion

In this paper, we propose Bat Planner, a novel quadrotor ball playing system, which allows the quadrotor to run back and forth and hit the ball aggressively like a human player. The key of the system is the motion planning module, including a front-end two-stage primitive generator and a back-end optimizer. Transition points are designed to provide guidance for hitting and make the higher-order control quantity change gradually. Primitives are generated by solving optimal state problems, with simplified hitting dynamics used. To obtain the best impact, highly coupled hitting states and time are jointly optimized, considering dynamic feasibility and collision avoidance. The defensive rule constraint and direct terminal constraint are proposed to prevent pathological hitting motions. Moreover, a differentiable trajectory prediction method is proposed to calculate gradients for the optimizer. To traverse almost all scenes in a human ball game, real-world experiments are designed to verify the robustness and a large number of simulation experiments have been constructed. Results show that Bat Planner can hit the ball from different directions, distances and angles to different targets. In the future, we plan to incorporate a residual term into the ball prediction model and use neural networks for online parameter identification to reduce trajectory prediction error.

References

[1] D. B chler, S. Guist, R. Calandra, V. Berenz, B. Sch lkopf and J. Peters, "Learning to Play Table Tennis From Scratch Using Muscular Robots," IEEE Trans. Robot., vol. 38, no. 6, pp. 3850-3860, Dec. 2022.

[2] S. Mori, K. Tanaka, S. Nishikawa, R. Niiyama and Y. Kuniyoshi, "High-Speed and Lightweight Humanoid Robot Arm for a Skillful Badminton Robot," IEEE Robot. Automat. Lett., vol. 3, no. 3, pp. 1727-1734, July 2018.

[3] M. Hattori et al., "Fast Tennis Swing Motion by Ball Trajectory Prediction and Joint Trajectory Modification in Standalone Humanoid Robot Real-time System," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 3612-3619.

[4] T. Senoo and I. Ishii, "Baseball Robots Based on Sensory-Motor Integration," in Proc. 21st Int. Conf. on Control, Automat. and Syst., 2021, pp. 1772-1777.

[5] Y. Ji et al.,"Hierarchical Reinforcement Learning for Precise Soccer Shooting Skills using a Quadrupedal Robot," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2022, pp. 1479-1486.

[6] E. Antonioni, V. Suriani, F. Riccio and D. Nardi, "Game Strategies for Physical Robot Soccer Players: A Survey," IEEE Trans. Games, vol. 13, no. 4, pp. 342-357, 2021.

[7] M. W. Mueller, S. Lupashin and R. D'Andrea, "Quadrocopter ball juggling," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2011, pp. 5113-5120.

[8] W. Dong, G. -Y. Gu, Ye Ding, X. Zhu and H. Ding, "Ball juggling with an under-actuated flying robot," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2015, pp. 68-73.

[9] M. W. Mueller, M. Hehn and R. D'Andrea, "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation," IEEE Trans. Robot., vol. 31, no. 6, pp. 1294-1310, Dec. 2015.

[10] H. Yu, P. Wang, J. Wang, et al. "Catch Planner: Catching High-Speed Targets in the Flight," arXiv preprint, arXiv:2302.04387, 2023.

[11] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in Proc. Eur. Control Conf., 2013, pp. 1383-1389.

[12] Fritsch, Frederick N., and Ralph E. Carlson. "Monotone piecewise cubic interpolation." SIAM J. Numer. Anal., 17.2 (1980): 238-246.

[13] D. Simon, Optimal State Estimation, John Wiley & Sons, 2006.

[14] D. P. Bertsekas, Dynamic Programming and Optimal Control, Vol. I. Athena Scientific, 2005.

[15] Levenberg, Kenneth, "A method for the solution of certain nonlinear problems in least squares," in Q. Appl. Math., 2.2 (1944): 164-168.

[16] Z. Wang, X. Zhou, C. Xu and F. Gao, "Geometrically Constrained Trajectory Optimization for Multicopters," IEEE Trans. Robot., vol. 38, no. 5, pp. 3259-3278, Oct. 2022.

[17] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," Math. Program., vol. 45, no. 1, pp. 503-528, 1989.

[18] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in Proc. IEEE Int. Conf. Robot. Automat., 2011, pp. 2520-2525.

[19] Watterson M, Kumar V, "Control of quadrotors using the hopf fibration on so(3)," in Robotics Research: The 18th International Symposium ISRR. Springer, Cham, 2020: 199-215.