Attachment for

Bat Planner: Aggressive Flying Ball Player

This attachment contains several charts to support and complement the experimental results and analysis mentioned in our paper. These charts provide visual representation of the performance of our method and the comparison algorithms.

1 Trjectory of the real-world experiments

We visualized the trajectories from 6 successful real-world experiment cases, each corresponding to one of our constructed scenarios, as shown in Fig. 1. The curve corresponding to the y-axis is the most informative, as it visually depicts from which direction the ball was launched and where the drone began its movement.

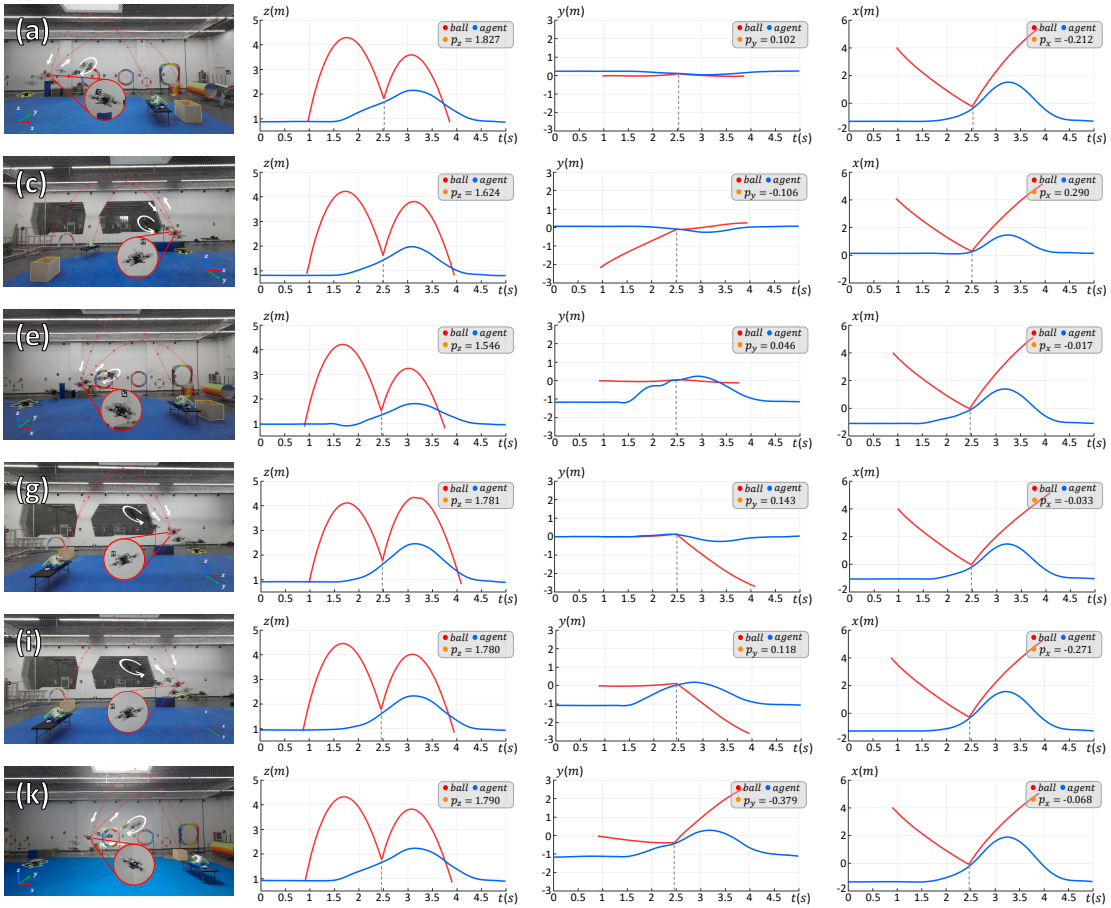

Figure 1: Visualization of the trajectories of the quadrotor and balls in the real-world experiments in Fig. 8 of the manuscript.

2 Trjectory of the simulations

We have repeated the simulation experiments for each of the 36 scenarios 1000 times, documenting crucial experiment data. This includes serve position (random), serve velocity (random), landing position (if the ball was not catched by the quadrotor), quadrotor's hitting position, hitting velocity and hitting attitude (quaternion). These results showcase the feasibility of our method in handling a variety of extreme scenarios. The varying heights, velocities, and attitudes of each ball hitting demonstrate the adaptability of our method. Noting that since the launches are randomized, we can't guarantee the results are identical to those initially reported in the paper, but they are very close, with an average discrepancy of less than 5%. For more details, please refer to the attached documents.

3 Success rate of the real-world experiments

The success rate plots are generated for each scenario based on 100 attempts, including the planning success rate of Bat Planner, the success rate of hitting the ball within a range of $0.5m$ and $1m$ from the target, and the planning success rate of the benchmarks, as shown in Fig. 2(a).

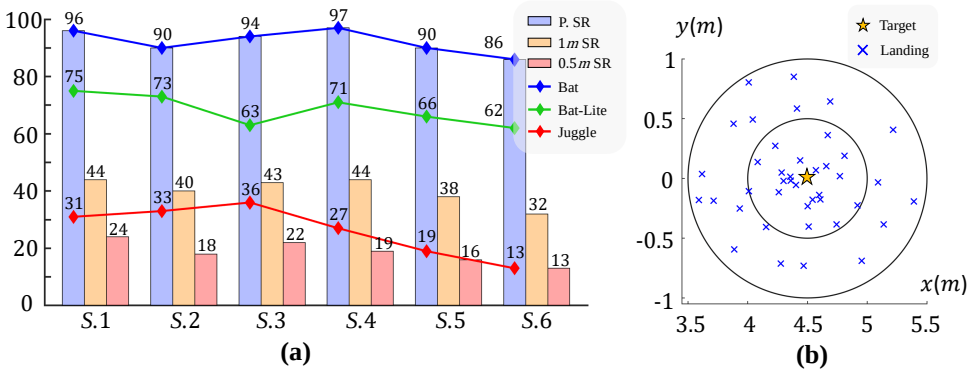

Figure 2: Results of real-world experiments. (a) Success rates for 6 scenarios, including planning success rate (P. SR), success rate within $0.5m$ ($0.5m$ SR) and $1m$ ($1m$ SR) from the target point, and the planning success rate of the benchmarks. (b) The hit balls' landing points for 100 attempts in $S.1$.

We also drawn the landing positions to present the experimental results more clearly, as shown in Fig. 3.

From the results, we can observe that the success rates of landing within the $1m$ and $0.5m$ range in the actual experiments are not over half of the planning success rate. The reasons for failure are analyzed in Sec. 4. Despite the aforementioned issues, our approach still outperforms the benchmark. Fig. 2 demonstrates that the planning success rate of the benchmark is even lower than the success rate of hitting the ball within a $1m$ range from the target. In actual ball hitting scenarios, the success rate of the benchmark would be lower. It is important to note that we have optimized the parameters for the benchmark to the best of our ability for each scenario, which greatly improves the success rate. If we were to use the original parameters of the benchmark, the success rate would be less than 5% for all scenarios. In reality, it is not feasible to rely on a single set of parameters to accommodate

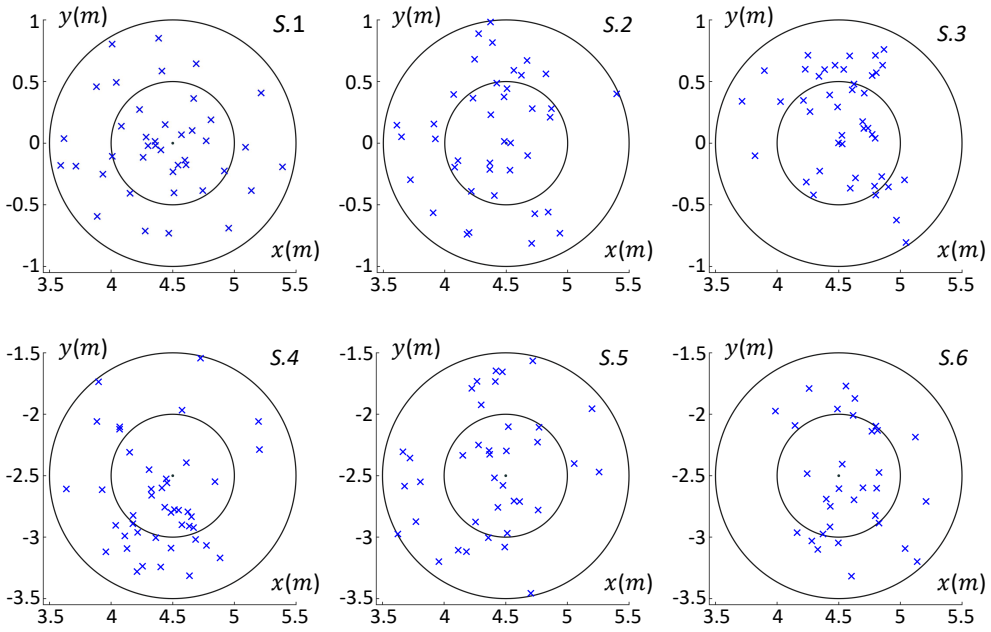

Figure 3: Landing plots for the 6 scenarios after the ball is hit.

all possible scenarios. Through comparison, we believe that our approach can handle a wide range of ball hitting scenarios, which is not achievable with existing methods.

4 Failure analysis in real-world experiments

We have identified the following reasons for this:

(1) Trajectory prediction error (primary reason): In our experiments, a majority of cases where the planning was successful but the ball was not hit within the target area can be attributed to errors in trajectory prediction. During our experiments, the three-dimensional decoupling errors when the ball is launched and falls below $1m$ height are within the ranges of $e_x = 2 - 15cm$, $e_y = 0 - 5cm$ and $e_z = 10 - 25cm$. Notably, the z-axis error is relatively larger, which can be explained. According to ball's dynamic $\ddot{\mathbf{s}}_b = \mathbf{g} - K_D \left\|\dot{\mathbf{s}}_b\right\| \dot{\mathbf{s}}_b$, it can be observed that the wind drag (the primary source of prediction error) is dependent on ball's velocity. However, the velocity in the z-axis is constantly changing, making drag coefficient K_D unstable and posing challenges for prediction. One potential solution to this problem is continuous re-planning. However, minor updates in the terminal state, especially velocity and attitude, may lead to drastic deformations in the trajectory. Intuitively, re-planning may result in abrupt turns or oscillations, which are unfavorable for control. Therefore, we did not implement re-planning in our experiments. To address this issue, we can optimize the trajectory predictor. One potential solution is to introduce residual terms to the trajectory prediction and use neural networks for identification. We will further investigate this in our future work.

(2) Model errors (secondary reason): There are errors in the drone's differential flatness model (which is linear simplified), the ball's dynamics model (which includes small forces such

as Magnus force and buoyancy in addition to gravity and drag), and the ball's rebound model (without considering air resistance after rebound). Regarding the differential flatness model error, especially the expression of angular velocity, we adopted state-of-the-art models [1]. For the ball's dynamics model, due to its nonlinearity, there is no perfect solution yet. We believe that introducing a residual term and using neural networks for fitting is a promising approach, which we will address in our future work. For the ball's rebound model error, we referred to the work [2] for target correction. All the aforementioned errors are cumulative, interdependent, and difficult to accurately assess and quantify. We believe that compared to trajectory prediction errors, these errors remain relatively small.

(3) Time delay (secondary reason). There are many factors that can cause time delay, such as the communication delay when the captured data is transmitted to the onboard computer through TCP, and the system delay generated when ROS subscribes or publishes, and so on. These delays are little, but still have an impact on the system as the drone and target moving at high speeds. Our solution is to set a backward time bias based on experience when predicting the trajectory of flying targets. In this article, we set it to $10ms$. In the actual testing process, we found that this is simple and effective.

(4) Control error (can be negligible): This is due to actuator errors, the uncertainty of the dynamic model and so on. To solve this problem, we adopted the SE(3) PID controller with cascade of angle velocity, angle, velocity and position. Before experiments, we firstly debug the controller. Expectations are calculated from trajectories in the shape of '8'. The angular velocity and angle are measured by IMU. We use the officially recommended Flight Review to observe the flight logs. The velocity and position are measured by the Vicon motion captures. We save the logs of the experiments by Rosbag toolkit and use Plotjuggler to view the errors. To ensure precise control, we carefully adjust parameters from the inner-loop to outer-loop. Finally, the control error is kept small, almost negligible.

(5) Measurement deviation (can be negligible): This is mainly due to the existence of sensor noises, especially in IMU caused by mechanical structural vibration, sensor temperature drift, etc. which are difficult to eliminate. Nevertheless, with the help of motion capture data and the use of EKF for data fusion, we are able to make the measurement values as accurate and smooth as possible.

References

[1] Watterson M, Kumar V, "Control of quadrotors using the hopf fibration on so(3)," in *Robotics Research*. Springer, Cham, 2020: 199-215.

[2] M. Müller, S. Lupashin and R. D'Andrea, "Quadrocopter ball juggling," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 5113-5120.