Trajectory Planning for Racing Drones

Atharva Navsalkar

I. Introduction

Research in racing drones is gaining popularity mainly due its immense application in agile drone flights, which have seen very limited deployment in real world. Human pilots have demonstrated the skills to fly quadrotors with extreme speeds and limited visibility. Their autonomous counterparts have not been even close in terms of performance. Typically, racing drones recommended by Drone Racing League, a very popular professional drone racing league, have top speeds of around 150-180 kmph, with thrust to weight ratio of 7:1 for a 1kg quad. Drones used in autonomous racing research and competitions are comparatively less aggressive. For example, the AlphaPilot Challenge provided a 3.4 kg drone with a thrust to weight ratio of 1.4 [1]. A custom drone developed by RPG Lab, University of Zurich, used in experimental research, have the ratio of 4 and weight of 0.8kg [2].

The characteristics discussed previously mainly differ according to the focus of the particular research goal. Autonomous drone racing poses four key challenges -

1) **Gate or Waypoint Detection**: The drone requires the pose of the gates with reasonable accuracy. Both traditional computer vision and learning based methods have been used in exiting research. Image Processing techniques have been used to efficiently the gates [3], with prior knowledge its structure. Others use CNNs to approximate gate positions or its features [1], [4].
2) **Localisation and Mapping**: Drones typically use Visual-inertial Odometry (VIO) [1], [4], [5], [3], to estimate the state. Stereo cameras and depth cameras are usually used to construct the map of the environment. Works that focus the aspects of gate ad waypoint detection and mapping, usually are incapable of extremely agile flights.
3) **Trajectory Generation**: Another important focus area trajectory generation for minimum lap times. Techniques like optimization [2], [6], [5], and reinforcement learning [7] have been primarily used in existing research. Additionally, sampling-based search methods [8], [1], [9] Research focusing on this aspect have previously succeeded in matching the times of human pilots, but work in an idealised environment with motion capture system for accurate localisation.
4) **Control**: Due to high speed and extreme orientation flights, PID control results in poor performance. The work [10] has been widely used to for trajectory tracking in $SE(3)$. Differential-flatness of the quadrotor inputs has been exploited in [11], [12] to design a non-linear controller. Model predictive control [13], [14] has also been increasingly used, thanks to its capability of considering model dynamics and constraints.

Atharva Navsalkar is a final-year undergraduate student in the Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, India anavsalkar@iitkgp.ac.in.

This work was carried out during the summer internship with Prof. David Saussié, Department of Electrical Engineering, Polytechnique Montréal, under the Mitacs Globalink Internship Program.

Code and additional videos can be accessed at https://github.com/anavsalkar/race_drone.

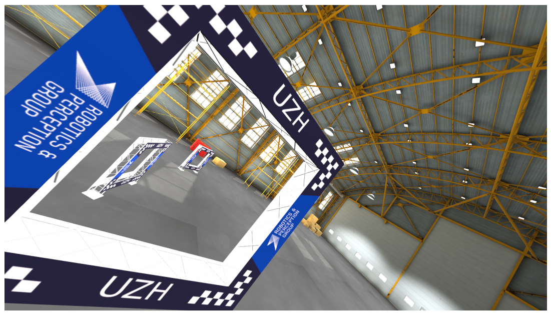

Fig. 1. First person view from the drone while passing through a gate.

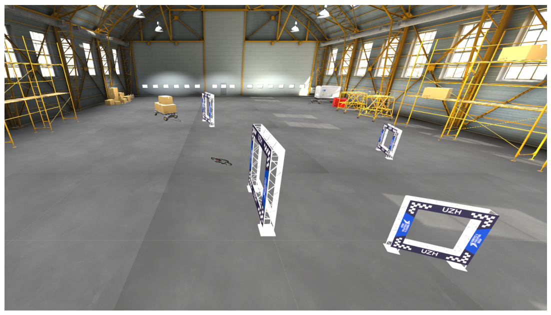

Fig. 2. External view of the drone and the track at same instance as above.

II. Prior Work

The work [1] describes the techniques used in the AlphPilot Drone Challenge, using only the onboard sensors. The authors use CNN-based algorithm to detect the gate corners and use them as features in the VIO pipeline to minimise drifts. On the control side, they use a receding horizon planner for few waypoints. Minimum time segments are found by randomly sample the possible states at each waypoint joined by a polynomial trajectory. This approach has been demonstrated to work in real world conditions without external measurements, but the lap times do not compare close with the optimization-based techniques.

To attain the maximum potential of the drone, authors in [2] resort to numerical optimization. They construct a

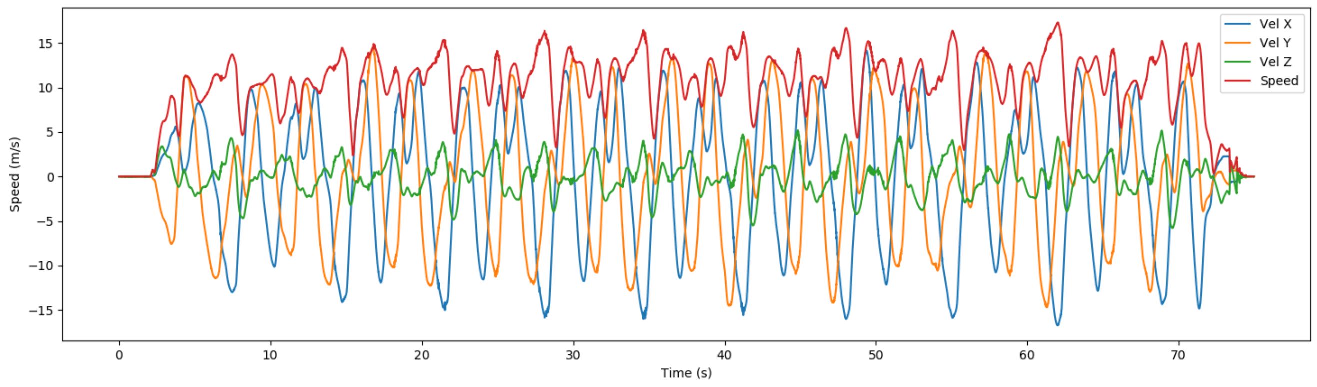

Fig. 3. Velocity values obtained by position measurements from an external motion capture system.

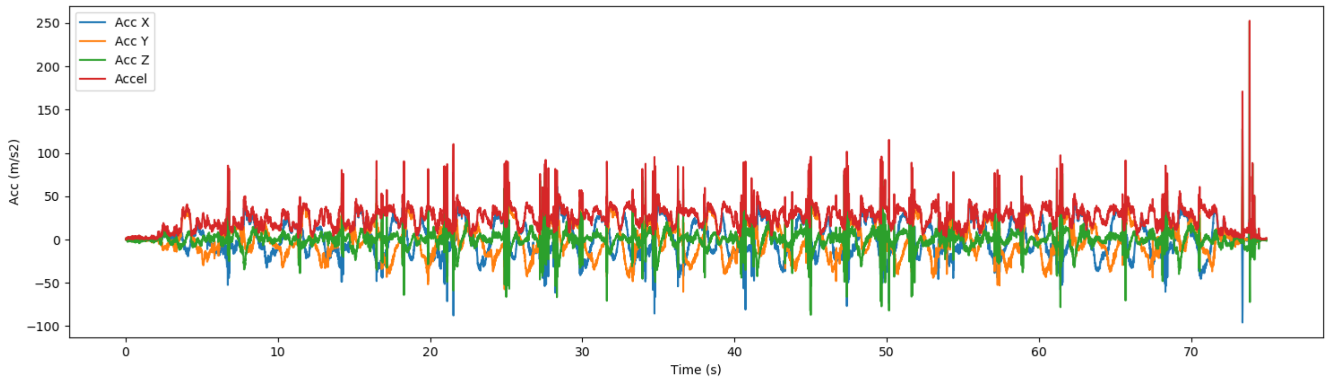

Fig. 4. Acceleration values obtained by position measurements from an external motion capture system.

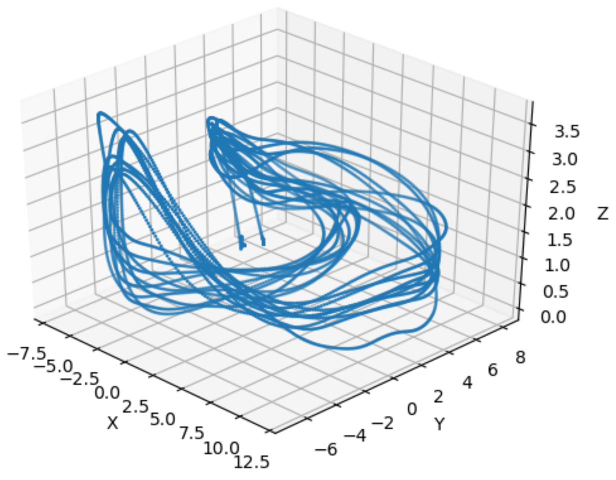

Fig. 5. Visualisation of the drone position data captured via external motion capture system.

single optimization problem for numerous laps, using multiple shooting with constraints on passing through waypoints and progress variables. The generated trajectory is tracked using a nonlinear MPC and it consistently beat the expert human pilots in similar conditions. But, this work uses states estimation and gate poses provided by a external motion capture system. Moreover, the solver time is in the order of hours on a normal computer, making it infeasible for real-time computation. This can serve as a baseline for other methods for comparison though.

Authors in [8] use hierarchical approach to generate the final trajectory. Additionally, multiple static obstacles are considered along with waypoint requirements. Initially, multiple diverse paths are found using Probabilistic Roadmaps and Dijkstra's algorithm. These paths are then further used to find point mass trajectory using gradient descent optimization and velocity search for multiple waypoints. Final trajectory is obtained using kinodynamics sampling-based method SST [15] and is tracked using MPC.

The work [14] extends the traditional MPC framework to combinely solve the planning and control problem. In addition to the reference error cost, the authors propose an additional cost for maximising the progress along the path and minimizing the contour distance from the path. It has been shown to effectively track dynamically infeasible trajectories obtained from point mass model. Compared to methods like [2], [7] which take long time to compute global trajectories, MPCC [14] tracks trajectories generated in real-time (but dynamically infeasible) to give comparable lap times.

A very recent extension of the previous work [9] leverages the MPCC [14] to track trajectories based on velocity sampling search approach to efficiently obtain time-optimal

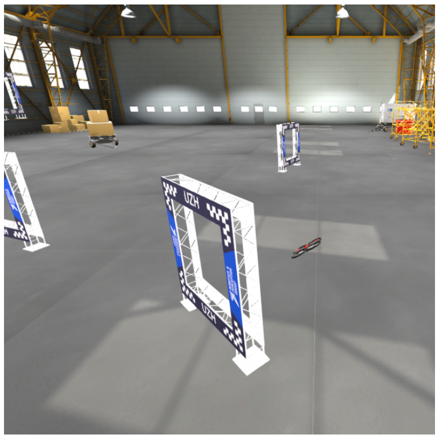
Fig. 6. High-fidelity graphics from Flightmare.

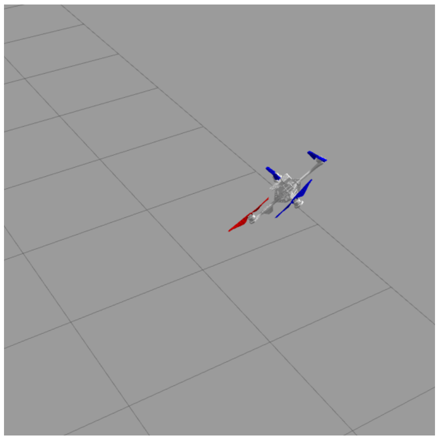
Fig. 7. Dynamics simulation using Gazebo.

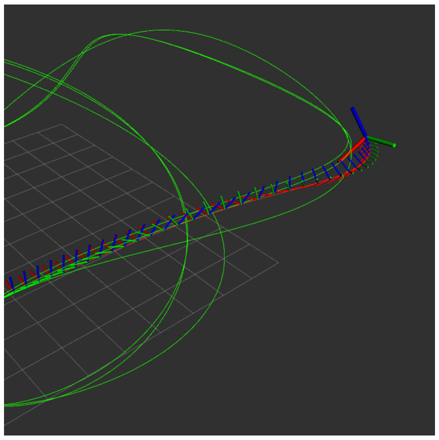
Fig. 8. Trajectory visualisation in Rviz.

point-mass model trajectory. Additionally, the use of a more recent solver 'HPIPM' with an updated 'acados' tool leads to extremely fast computation. This is one of the most recent work in this area.

Building on the advances in MPC, authors in [13] propose a real-time approach that considers both waypoint constraints, spatial bounds and time optimality simultaneously. The MPC problem minimizes the time with pose constraints according to polyhedral tunnels joining the gates.

Most of the works assume the gates to be of sufficiently large. Authors in [6] address the autonomous racing in cluttered environments with narrow and oblique gaps. Full trajectory optimization is used to plan trajectories in SE(3), considering a polyhedral representation of the drone. Moreover, GPU-based computation is used for faster solver time. Main focus of this work is safety in tracks with very narrow and constrained gaps.

As an alternative to classic methods, the work [7] use deep reinforcement learning to compute minimum-time trajectories. A infinite-horizon Markov Decision Process defines the problem with reward for path progress and distance from waypoint centre. The action space consists of the directly the rotor thrusts, without any additional controller. The policy trained for any particular track has been showed to be close to the times obtained using [2].

Another work [16] use CNN to map the ram image data to a look-ahead waypoint. The CNN is trained using expert policy to follow a global constrained minimum snap trajectory [17]. During the final run, the next waypoint output of the CNN is transformed into a smooth trajectory. The time optimality of the trajectory is not considered, but is capable of planning using onboard data. Reinforcement Learning is increasingly used for control of systems with complex dynamics or environment. Agile flights in cluttered environments is a perfect use-case for such approach. The work [18] combines the sample-based methods and reinforcement learning to generate minimum-time aggressive trajectories. First, topological paths connecting the waypoint and avoiding obstacles is obtained using a novel sampling based method. Next, learning algorithm with proxy reward for path progress maximization and obstacles avoidance is deployed to generate the control policy.

As seen in [16], neural network is used to process high-dimensional image data to low-dimensional outputs for MPC. This uses both the advantages of learning interfaced with optimization. In [19], author interface policy search (episodic RL) with the MPC. Authors design algorithms to learn Gaussian as well as neural network policies. Both simulations and experimental results have been discussed for drones passing through highly dynamic gates.

In terms of control, MPC and differential-flatness-based control remain the most popular choice. Reference [20] compares the performance of both these controllers. It is found that both the methods have similar performance for feasible trajectories, whereas MPC has an edge with model uncertainty and external disturbances. Moreover, the authors stress heavily on the choice of low-level controller affecting system performance.

It can be clearly seen that the works with extremely aggressive flights touching the limits of the system are mostly based on external state estimation. The ones using onboard sensor data only are limited in terms of aggressiveness. Offline path planning is possible only when prior information on the gate positions is known. In case it is unavailable, learning based algorithms are usually used to detect the gates and constructs a map. It is of primary importance that, for a real-world experiments in uncontrolled environments, a fast and iterative method must be used for flight robust to disturbances, uncertain dynamics and state estimation drift.

For comparison with expert human pilots, authors in [2] capture data from professional drone racing pilots using a motion capture system for multiple laps. Open-source dataset for flight data from two pilots has been made available. Figures 3, 4 and 5 represent the captured data. The velocities and accelerations have been obtained by discrete numerical differentiation from the position data. The spike denote the instances when the drone passes a gate, as human pilots perform a high-jerk maneuver to point towards next gate. The speed of the drone hovers around 10 ms^{-1}, and acceleration around 25 ms^{-2}.

III. Simulation Overview

In this project, a realistic simulation framework is set up that can be utilized to focus on all four aspects on drone racing. For implementing the computer vision techniques accurately, a photo-realistic simulator is necessary. Microsoft Airsim [21] is one of the most popular simulators that uses Unreal Game Engine to generate graphics. Despite its high-fidelity graphics, it fails to match accuracy of the Gazebo Physics simulator in terms of dynamics. On the other hand, Gazebo fails to provide high quality graphics for computer vision and training AI. Therefore, a hybrid approach is used here, that simulates the dynamics on a physics simulator and generates graphics using photo-realistic simulator. Flightmare [22], based on Unity game engine (see Fig. 6) is use on top of the Gazebo simulator (see Fig. 7). The entire control framework uses the high and low-level controller, discussed in [11], [23]. Robot Operating System (ROS) is used for communication between Flightmare, Gazebo and autopilot controller. Figure 8 shows real-time visualization of the global and predicted trajectory of the drone at an instance.

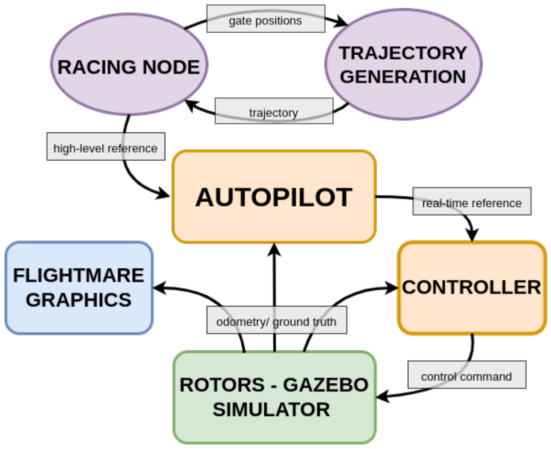

Fig. 9. Flowchart representing the flow of information in the code base.

Figure 9 depicts how various nodes in the code interact with each other. Each node is denotes as a enclosed shape with arrows showing the exchange of information. The nodes in a circular shape (in violet) are run offline before starting to execute trajectory. Initially, the user can specify the track data on gate positions, or desired time segments to the racing node (highlighted in violet). It obtains the complete trajectory from the trajectory generation node. In this work, minimum snap trajectory [17] method is used to generate polynomial trajectories. The gate positions are expressed as orderly sequence of waypoints. In total, these two nodes act as high-level planner. The Autopilot node is the primary node responsible commanding the quadrotor in real-time. It consists of a finite-state machines with modes such as, TAKEOFF, LAND, GO TO POSE, EXECUTE TRAJECTORY, etc., and ensures smooth transition between different states. As per the current state of autopilot, it communicates the real-time reference with the controller that finally computes the control command. The use of two of the most commonly used controllers viz., differential flatness-based controller (DFBC) and nonlinear model predictive control (NMPC) have been demonstrated further. For simulations, RotorS interface [24] to the Gazebo loads the quadrotor model and publish the real-time ground truth as obtained from Gazebo simulation, which is used for feedback and Flightmare simulator to generate graphics.

A. Differential Flatness-based Controller

Quadrotors have a unique (and useful) property of differential flatness. This means that all the states $\mathbf{x}$ (that include position, velocity, orientation and angular rates) can be written as smooth functions of flat outputs and their derivatives. For quadrotors, these flat outputs are position $\mathbf{r}$ and yaw angle ψ (or heading direction). Hence, all the state references can be calculated if a smooth trajectory is known in terms of positions and heading angle,

$$\mathbf{x}_{ref} \leftarrow \mathrm{DF}([\mathbf{r}, \psi]),$$

where DF() represents function to inversely calculate states from differential flatness property. These references can be tracked using commonly used PD controllers with feedforward terms for different components of the state vector. This method is detailed in [11]. Additionally, a low-level controller is required to get individual rotor speeds.

B. Model Predictive Control

Model Predictive Control (MPC) is a recursive finite-horizon optimal control problem. Figure 10 depicts the solution obtained for finite time horizon with a discrete time sample. The predicted states using the dynamics model aim to track the reference trajectory. Due to the use of numerical optimization techniques, various constraints on states and controls including the dynamics and limits can be applied.

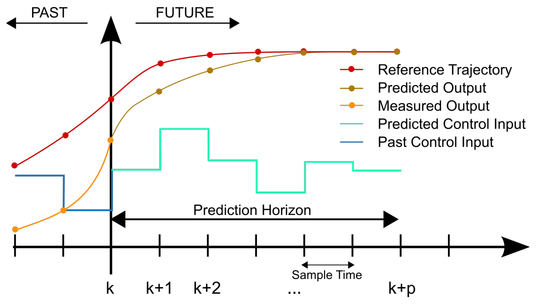

Fig. 10. Visual representation of receding-horizon control or MPC. Image obtained from Wikipedia.

The objectives function $J(\mathbf{x}, \mathbf{u}, \mathbf{x}_{ref})$ corresponds to the difference between current and the reference state, combined with the control cost. The optimal control problem at time step k given by

$$\begin{aligned}\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=1}^{N} J^{k+i|k}(\mathbf{x}, \mathbf{u}, \mathbf{x}_{ref}) \\ \text{s.t.} \quad & \mathbf{x}(k+i+1|k) = f(\mathbf{x}(k+i|k), \mathbf{u}(k+i|k)), \\ & x \in \mathcal{X},\ u \in \mathcal{U}, \\ & \text{for } i \in \{1, ..., N\},\end{aligned}$$

where $f(\mathbf{x}, \mathbf{u})$ describes the discrete-time nonlinear dynamics of the systems, and $\mathcal{X}$ and $\mathcal{U}$ being the limits on states and control inputs. This optimization problem is solved using qpOASES [25] solver. More details can be obtained in [26].

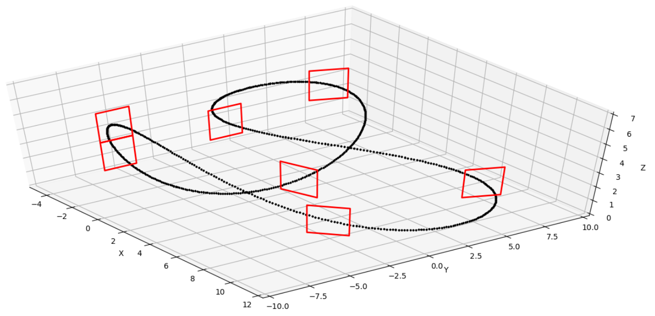

Fig. 11. 3D visuals of the trajectory and gate positions.

C. Trajectory Generation

The high-level trajectory planner uses minimum snap trajectory generation [17]. Snap is the second derivative of acceleration, denoted by $\frac{d^4\mathbf{r}}{dt}$. When working with quadrotors, we want to find a trajectory that minimizes the snap cost function denoted by

$$\mathbf{r}^\star(t) = \underset{\mathbf{r}(t)}{\text{argmin}} \int_0^t \|\mathbf{r}^{(4)}\|^2 dt.$$

For this task, higher degree polynomials are used as basis functions for trajectory segment between each waypoint. After solving for the minimization problem, we get coefficients for each segment, which can be used to calculate references, smooth at least up to fourth order of derivative, at any given time. This trajectory containing reference positions, velocities, jerks and snaps, is sent to the autopilot that executes the trajectory. Figure 11 shows generated trajectory for the particular gate positions of the racing track. The racing track environment is made using gate positions obtained from a recent reference [14].

D. Performance

Same track and trajectory is used to demonstrate both the controllers described in previous subsections. Figures 12 and 13 show the X-Y plane projection of the trajectories tracked by each of the controller. Both the controllers give very good tracking performance even at high speeds. The advantages of MPC are more observed when the high-level trajectory violates systems limits, as MPC gives stable flights for all scenarios. Certainly, MPC is a much more computationally expensive, hence cannot be implemented on smaller hardware. Also, the performance is sensitive to tuning of the cost function. Hence, DFBC remains popular for onboard implementation. Reference [20] analyses the differences is great detail. Additional videos can be accessed on the project Github page.

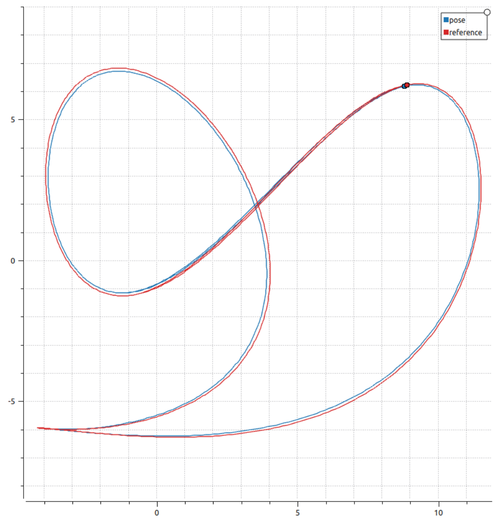

Fig. 12. 2D view of trajectory tracking using DFBC.

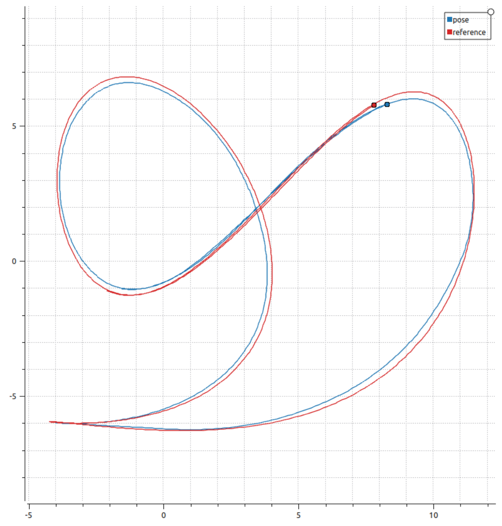

Fig. 13. 2D view of trajectory tracking using NMPC.

IV. Conclusion

To summarise the work, the primary goal and fundamental challenges for autonomous drone racing have been discussed. The existing research can be classified in four directions namely, perception, localisation, trajectory generation and control. Keeping the discussion focused on the later two aspects, the latest state-of-the-art research is discussed. Assuming perfect knowledge of the surrounding, the autonomous drones have been able to match the human potential. Large scope still remains, as professional pilots only rely on real-time vision data. Next, a modular framework has been established that would be useful to conduct research on all four aspects. Commonly used minimum snap trajectory is used, combined with DFBC or NMPC controllers, demonstrate the basic drone racing tasks and effectiveness of the simulation framework.

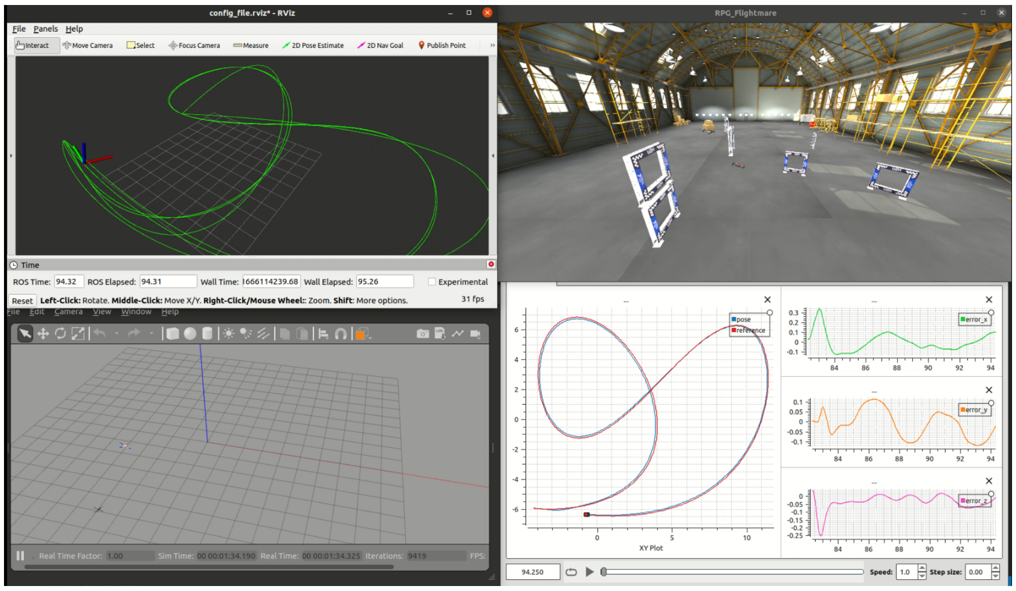

Fig. 14. Screenshot of the real time data and graphics visualization on the computer during a racing mission.

References

[1] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[2] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[3] S. Li, M. M. Ozo, C. De Wagter, and G. C. de Croon, "Autonomous drone race: A computationally efficient vision-based navigation and control strategy," *Robotics and Autonomous Systems*, vol. 133, p. 103621, 2020.

[4] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[5] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[6] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for $SE(3)$ planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.

[7] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[8] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[9] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *arXiv preprint arXiv:2203.09839*, 2022.

[10] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[11] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[12] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, 2020.

[13] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," *arXiv preprint arXiv:2110.01351*, 2021.

[14] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, 2022.

[15] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[16] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 133–145.

[17] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[18] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *arXiv preprint arXiv:2203.15052*, 2022.

[19] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Transactions on Robotics*, 2022.

[20] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, 2022.

[21] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[22] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proceedings of the 2020 Conference on Robot Learning*, 2021, pp. 1147–1157.

[23] M. Faessler, D. Falanga, and D. Scaramuzza, "Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 476–482, 2016.

[24] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot operating system (ROS)*. Springer, 2016, pp. 595–625.

[25] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[26] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.