Team Deadlock - Test 3 Technical Report

March 2019

1 Summary

Our autonomous flight pipeline implemented in FlightGoggles is broken down into two main aspects: **Flight planning/controls**, and **Perception**. These consisted of classical control theory algorithms and vision-based navigation and state estimation. More specifically for Test 3,

1. VINS-Mono as a full-state visual-inertial odometry state estimator.
2. PBVS + PID for position and attitude control.
3. Mueller's computationally efficient trajectory planner.
4. Classical camera triangulation methods for gate location estimation.

We also want to be transparent and say that we were **not** able to perfectly fly through the racetrack from start to finish. We were only able to do so when subscribing to the true gate location ROS parameter */uav/Gate/location*. The only issue we encountered was that we were not able to get an accurate enough gate estimation on *some* gates (usually Gate #6). We are still proud of what we have accomplished for Test 3, and we're confident in a smoother flight as we work on this more.

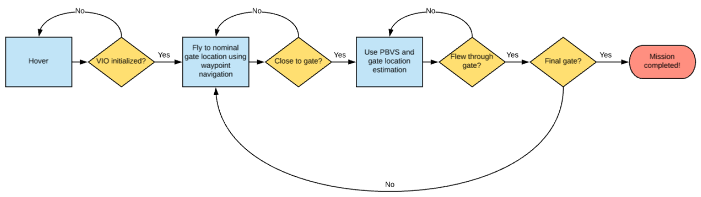

Figure 1: Simplified state machine implemented in FlightGoggles.

2 Flight planning & Flight controls

2.1 Controllers

For position control, we use a simple hybrid between waypoint navigation and position based visual servo (PBVS) controller. Nominal gate locations are fed into the trajectory generator (explained in Path planning section) and are treated akin to waypoint navigation, and switches to PBVS when near a gate for precision to fly through the center (explained in Gate estimation section). Then, the cross-track error is fed into a manually-tuned PD controller.

Finally, attitude stabilization is done via a PID controller which takes in attitude commands and outputs angular rate commands which are then fed into FlightGoggle's lower level controller as a *mav_msgs/RateThrust* ROS topic. We decided that a PID attitude stabilization controller is sufficient for Test 3 because due to the camera not being angled, we were flying between 25°-35°to maintain view of the gate, which is not too far from the linear dynamics regime (max of 4% deviation).

Note on controller design

We are unsure on the restrictions of the controller flexibility due to the fact that PID controllers are standardly used on racing drones. However, this isn't a big issue moving forward.

If for whatever reason we are restricted to PID controllers in the finals, we will design optimized controllers based on system identification models (we can conduct frequency domain based system identification methods ourselves if a plant model will not be provided). We will then design a schedule of controller gains for two main things: (1) feasible aggressive maneuvers, (2) robustness to disturbance (i.e optimal disturbance rejection bandwidth).

If we are free to implement non-linear controllers, we would implement an incremental non-linear dynamic inversion controller [6,7] due to the fact that they are sensor-based. This provides an extremely powerful capability of robust disturbance rejection and tracking of aggressive trajectories.

2.2 Path planner

A computationally efficient trajectory planner that minimizes input aggressiveness and time while rapidly evaluating feasibility of output trajectories based on system constrains was developed based on the work of Mueller M.W. et al.[9]. The idea is to rapidly search over a large number of possible motions in order to identify a trajectory that best achieves a given high level goal. To achieve computational efficiency, the algorithm does not encode trajectory constraints explicitly in the planning phase. Rather, it verifies trajectory feasibility a posteriori.

The goal of the trajectory planner is to generate a thrice differentiable trajectory that guides the drone from the current state (i.e. current position and attitude) to a final state (i.e. desired state at the next nominal gate location) in time T, while minimizing a cost function that encodes jerk j and serves as an upper bound of the inputs of thrust f and body rates w

$$J_\Sigma = \frac{1}{T}\int_0^T ||j(t)||^2 dt \tag{1}$$

$$\frac{1}{T}\int_0^T ||f \begin{bmatrix} w_2 \\ -w_1 \\ 0 \end{bmatrix}||^2 dt = \frac{1}{T}\int_0^T || \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} R^{-1} j||^2 dt \leq J_\Sigma \tag{2}$$

where R is the rotation matrix.

The nonlinear trajectory generation problem is first simplified by decoupling the dynamics into three orthogonal axes and treating each axis as a triple integrator. Each spatial axis is solved for independently using Pontryagin's minimum principle. The decoupled axes are then recombined after the optimization process in order to evaluate feasibility of the whole trajectory, that is, whether the particular trajectory is a collision course or whether it requires unrealistic inputs in a form of thrust and angular velocities. This approach is shown to have closed-form solutions, which would avoid complexities associated with convex optimization based methods and allow the algorithm to be relatively light-weight.

The inherent low computational cost is exploited by rapid search and evaluation of feasible trajectories, allowing planning flexibility. A search extracts and outputs the trajectory, along with the associated states, that has shortest travel time with feasible system inputs. Such an approach is then integrated in a complex high level autonomous architecture.

3 Perception

3.1 State estimation

We use visual-inertial odometry (VIO) to provide full state estimation throughout the flight. Although stereo vision is allowed in Test 3, we decided to go for a monocular based VIO algorithm in the case that the actual racing drone is restricted to monocular vision. Hence, VINS-Mono[1] was chosen due to its high accuracy and robustness under high speed motion. VINS-Mono is a graph-optimization-based solution to non-linear visual inertial system. Eq.3 is the objective function of VINS-Mono, which features a tightly coupled and sliding window-based formulation.

$$\boldsymbol{X}^* = \underset{X}{\operatorname{argmin}} \left\{ \|\boldsymbol{r}_p - \boldsymbol{H}_p X\|^2 + \sum_{k\in\beta} \left\|\boldsymbol{r}_\beta(\hat{\boldsymbol{z}}^{b_k}_{b_{k+1}}, X)\right\|^2_{\boldsymbol{P}^{b_k}_{b_{k+1}}} + \sum_{(l,j)\in C} \rho(\left\|\boldsymbol{r}_C(\hat{\boldsymbol{z}}^{c_j}_l, X)\right\|^2_{\boldsymbol{P}^{c_j}_l}) \right\} \tag{3}$$

For brevity, we refer the reader to [1] for detailed definition and derivation of the variables and symbols in the function. Eq.3 is linearized and solved iteratively using Ceres[2], an incremental linear solver. The key to the robustness of VINS-Mono lies in its on-the-fly estimator initialization and loop closure detection. Estimator initialization ensures the above nonlinear optimization always converge[3]. Loop detection utilizes DBoW2[4], an implementation of bag-of-words (BOW) place recognition, which runs in the background and constantly check the loop closure to reduce drift[5].

3.2 Gate location estimation

Since we know the camera intrinsic parameters, gate widths, and IR marker locations in the pixel space, we are able to obtain the 3D location of the gate by solving the following mapping relationship between the pixel space and inertial frame [9]:

$$\begin{bmatrix} p_{x_c} \\ p_{y_c} \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^c_e & \mathbf{T} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} p_{x_e} \\ p_{y_e} \\ p_{z_e} \\ 1 \end{bmatrix} \tag{4}$$

where f_x and f_y are the focal length in the horizontal and vertical axis respectively, α is the camera scale factor, $\mathbf{R}^c_e$ denotes the rotation matrix from the inertial frame to camera frame, and $\mathbf{T}$ denotes the translation vector. This 3D location is fed into the controller and path planner.

4 Next steps

We understand that the implementation in FlightGoggles only serves as a baseline to the actual due to the fact that many parameters were given in FlightGoggles may not be available to use in actual flight (i.e IR markers, noise variances, gate width, etc). Our next steps include:

1. More thorough state-estimation validation of VIO using UZH-FPV dataset.
2. A more robust and aggressive vision based planning pipeline using simulated moving gates.
3. Learning-based methods for aggressive maneuvers.
4. More thorough controller design as mentioned in Section **2.1**
5. Implementation of Structure from Motion for more robust environment estimation.

References

[1]T. Qin and S. Shen, "Robust initialization of monocular visual-inertial estimation on aerial robots," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 4225-4232.

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org/

[3] D. Galvez-López and J. D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," in IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1188-1197, Oct. 2012.

[4] P. Li, T. Qin, B. Hu, F. Zhu and S. Shen, "Monocular Visual-Inertial State Estimation for Mobile Augmented Reality," 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Nantes, 2017, pp. 11-21.

[5] W. Wei, and Tischler, M. "System Identification and Controller Optimization of a Quadrotor UAV". AHS 2017. https://nams.usra.edu/NAMS/assets/AFDD/AHS_2015_Wei.pdf

[6] E. Smeur, et al. "Cascaded Incremental Nonlinear Dynamic InversionControl for MAV Disturbance Rejection" Jan 2017 preprint, https://arxiv.org/pdf/1701.07254.pdf

[7] E. Tal, and S. Karaman, "Accurate Tracking of Aggressive Quadrotor Trajectories usingIncremental Nonlinear Dynamic Inversion and Differential Flatness". Sep 2018 preprint, https://arxiv.org/pdf/1809.04048.pdf.

[8] Mueller, M.K., Hehn, M., and D'Andrea, R. "A computationally efficient motion primitive for quadrocopter trajectory generation", 2015, IEEE Transactions on Robotics, Volume 31, no.8, pages 1294-1310.

[9] Chaumette, F. and Hutchinson, S. "Visual servo control" 2007, IEEE

[10] J. Delmerico, et al. "Are We Ready for Autonomous Drone Racing?The UZH-FPV Drone Racing Dataset" ICRA 2019.

[10] E. Kauffman, et al. "Deep Drone Racing" https://arxiv.org/pdf/1806.08548.pdf