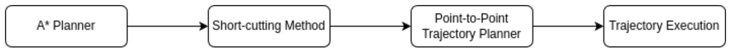
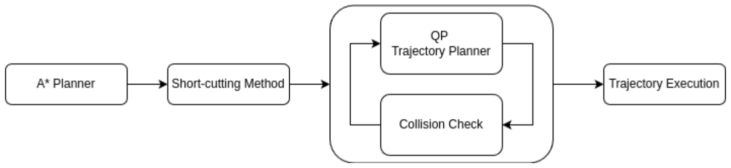
Project Report: An Autonomous VIO-based Quadcopter

Thomas Stephen Felix



(a) Project 1.3



(b) Project 3

Fig. 1: (a) and (b) represent the trajectory generation workflow for Project 1.3 and Project 3 respectively.

I. Trajectory Tracking with State Estimator

A. Overview

This project aims to achieve autonomous quadcopter flight by integrating state estimation, specifically using Visual Inertial Odometry (VIO), with trajectory planning and control modules developed in previous course stages. The core challenge lies in adapting the planning and control algorithms to rely solely on the robot's VIO-derived state estimate, rather than ground-truth data. All development and testing occur within the flightsim simulation environment, which provides simulated VIO sensor input through pre-calculated feature projections, enabling closed-loop autonomous navigation based on onboard sensing.

B. An breakdown of Project 1.3

To provide context for the system improvements implemented in Project 3, this section outlines the baseline trajectory planning system, conceptually illustrated in Figure 1a. The system incorporates the following key components:

1) **A* Planner**: The A* search algorithm is employed to generate a feasible path from a designated start location to a goal location. Path generation is guided by the following cost functions:

$$g(v) = \text{Number of steps from the start location}$$
$$h(v) = \text{Euclidean distance to the goal location}$$
$$f(v) = g(v) + h(v)$$

The environment is represented by a map discretized with a uniform resolution of 0.25 m per dimension. This resolution was selected such that the quadrotor is fully encompassed within a grid cell when positioned at its center. A safety margin of 0.25 m is maintained to ensure adequate clearance from obstacles.

2) **Path Short-cutting**: This post-processing step refines the initial A* path. It identifies pairs of waypoints on the trajectory that can be connected by a collision-free straight line, subsequently removing any intermediate waypoints between them. This process reduces the trajectory representation to a sequence of essential waypoints, as depicted in Figure 2.

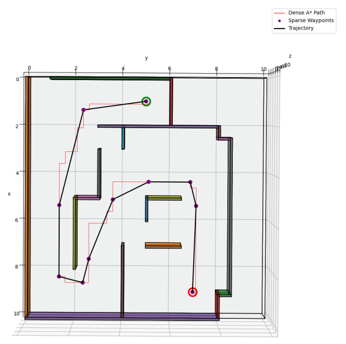
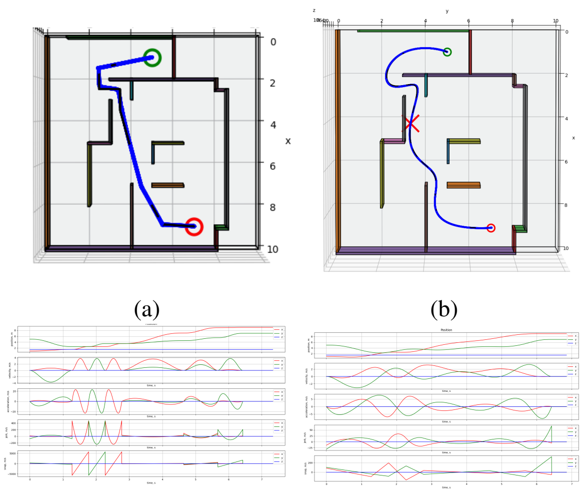


Fig. 2: Illustration of the short-cutting algorithm selecting essential waypoints from an A* generated trajectory.

3) **Point-to-Point Trajectory Planner**: As implemented in Project 1.3, this component generates smooth trajectories between consecutive essential waypoints obtained from the short-cutting process. It operates under the assumption that the quadrotor starts from rest and comes to rest at the end of each segment. The trajectory for each individual segment is computed by solving a least-squares optimization problem formulated with endpoint constraints for position, zero velocity and zero acceleration in the form

$$Ax = b$$

C. Improvements



(a) (b)

(c) (d)

Fig. 3: (a) refers to the path generated after reducing the resolution. (c) is a plot of the kinematics vs time of the trajectory followed. (b) and (d) show the 3D path and kinematics of the path generated

1) Reduced Resolution: Initial investigations revealed that the map resolution employed previously (as discussed in the context of Figure 2) was insufficient to identify narrow passages potentially navigable by the quadrotor. Consequently, the map resolution was refined to 0.2 m per dimension. This finer discretization enabled the planner to discover shorter, feasible paths through confined spaces, as illustrated in Figure 3a.

2) Trajectory Planning as a Quadratic Program (QP): The point-to-point trajectory planner, while functional, exhibited limitations related to dynamic feasibility, evident in analyses such as Figure 3c. The requirement for the quadrotor to achieve zero velocity and acceleration at each intermediate waypoint resulted in discontinuous velocity/acceleration profiles and inefficient flight, characterized by excessive time spent on acceleration and deceleration cycles.

To address these shortcomings, the trajectory generation problem was reformulated as a Quadratic Program (QP). This approach aims to find a dynamically smoother path by optimizing the entire trajectory simultaneously. The specific formulation involved the following objectives and constraints (detailed equations are provided in the Appendix):

a) Trajectory endpoints must coincide with the designated waypoints.
b) Ensure continuity of velocity and acceleration profiles across segment junctions.
c) Minimize the integral of squared jerk along the entire trajectory.

However, optimizing solely based on these criteria frequently produced trajectories that violated environmental constraints, leading to collisions (illustrated in Figure 3b). To mitigate collisions while retaining the benefits of the QP formulation, several strategies were implemented:

- **Waypoint Insertion**: Intermediate waypoints were strategically inserted between consecutive waypoints derived from the path short-cutting phase. This aimed to constrain the optimized trajectory closer to the initial, geometrically feasible path.
- **QP Inequality Constraints**: The QP formulation was augmented with inequality constraints to enforce dynamic limits and safety. These included: (a) maximum velocity limits, (b) maximum acceleration limits, (c) constraints penalizing deviation from the initial shortcut path segments, and (d) constraints enforcing a minimum distance from detected obstacles near the trajectory.
- **Sharp Turn Handling**: The angle between successive trajectory segments was monitored. If a turn exceeded a predefined threshold (e.g., 45 degrees), the time allocation for the relevant segments could be adjusted, implicitly allowing the optimizer to find a lower-acceleration maneuver.

While combinations of these methods yielded improvements in specific scenarios, they often lacked robustness and computational efficiency, particularly when higher quadrotor speeds were required.

3) Iterative Collision Check and QP Refinement: Further experimentation led to a more robust two-phase approach, detailed in Algorithm 1. This method integrates trajectory optimization with iterative collision checking and refinement.

The process begins with the waypoints generated by the short-cutting algorithm. First, an initial minimum-jerk trajectory connecting these points is computed via the QP formulation. Second, this trajectory is densely sampled and checked for collisions against the occupancy map. If the trajectory is collision-free, it is accepted. However, if a collision is detected within a particular segment, a new waypoint is inserted at the midpoint of that colliding segment. The QP optimization is then re-solved with this updated set of waypoints. This generate-check-refine cycle repeats until a fully collision-free trajectory is obtained or a maximum iteration limit is reached.

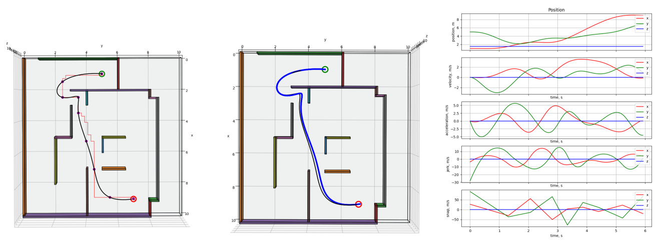
Map	Maze	OverUnder	Window	Slalom	Stairwell	Switchback
P13 Time (s)	12.01	16.35	13.1	27.5	19.14	36.42
IQP Time (s)	5.39	13.57	7.36	16.24	14.31	23.31

TABLE I: The table shows the performance of the iterative QP (IQP) algorithm vs Project 1.3 (P13) as measured in Gradescope

Algorithm 1 Iterative Spline Refinement with Collision Checking

```
procedure MAKESPLINES(initialPoints)
    $points \leftarrow initialPoints$
    for $i \leftarrow 1$ to 10 do
        $trajectory \leftarrow$ GETQPTRAJECTORY($points$)
        for all $segment$ in $trajectory$ do
            if COLLISIONDETECTED(segment) then
                Add $midPoint$ to $points$
            end if
        end for
        if no trajectory collisions detected then
            return $trajectory$
        end if
    end for
    return $trajectory$
end procedure
```

D. Results



(a) (b) (c)

Fig. 4: (a) path generated by A* and the final way points used to generate the trajectory. (b) path followed by quadrotor. (c) smooth kinematics of the quadrotor when following the path

The performance of the iterative planning method proved highly effective, as illustrated in Figure 4. The planner consistently generated feasible, collision-free trajectories, even at significant operational speeds. This robust planning capability was a key factor in achieving a final score of **97.52** on Gradescope. The performance improvement as compared to Project 1.3 can be analyzed in table I.

E. Discussion

Further effort was also directed towards tuning the PID control parameters. Throughout the project's duration, the existing state estimation pipeline functioned effectively, necessitating no additional modifications to that system component.

II. Local Trajectory Re-planning

A. Implementation

The local trajectory planner included the following modifications :

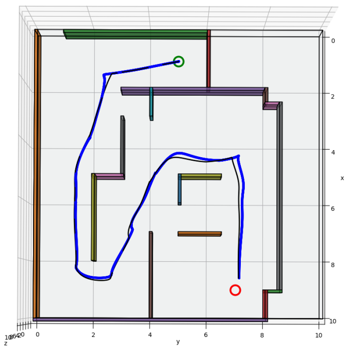
- **Modified A* Goal Condition**: To handle scenarios where the precise goal coordinates might be inside an obstacle or otherwise inaccessible, the A* search termination condition was relaxed. Instead of requiring the path to reach the exact goal, the search concludes successfully upon finding a valid, accessible node within a specified tolerance radius (e.g., $0.25\,\mathrm{m}$) of the target goal.
- **QP Trajectory Planner**: A Quadratic Programming (QP) solver, as previously described in Section I-C.2, was employed to generate smooth trajectories between the waypoints identified by A*. This QP planner incorporated the following enhancements:
  - Generation of intermediate waypoints for trajectory segments exceeding a certain length, ensuring better path control.
  - Inclusion of an inequality constraint to restrict the maximum velocity along the trajectory (e.g., to $3\,\mathrm{m/s}$).
  - Adaptive time allocation for segments involving sharp turns (determined by angle thresholds), allowing the quadrotor more time to execute demanding maneuvers smoothly.
- **Spline Timing Representation**: Each trajectory segment is represented by an instance of a dedicated **Spline** class. This object stores all relevant information for the segment, including its polynomial coefficients, duration, and its start and end times referenced to a global clock. While internal calculations might use a relative time frame (starting from zero for each segment), this is consistently mapped to absolute time for overall trajectory execution and synchronization.

The `world_traj.py` module incorporates a reactive replanning mechanism to handle potential collisions detected along the current trajectory. Within the `replan` method, the currently planned trajectory is periodically checked for future collisions using the local occupancy map. If an impending collision is detected within a defined horizon, the system triggers a trajectory regeneration sequence. This process involves capturing the quadrotor's current state as the new starting point, defining a new local target goal, and re-invoking the full trajectory planning pipeline (`plan_traj`) including graph search, path shortcutting, and QP-based spline optimization to generate a new, feasible trajectory originating from the current state.

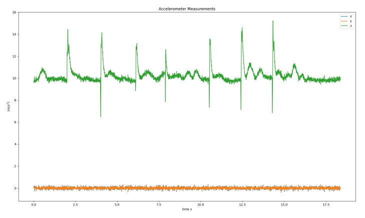
B. Performance

The system exhibited the following key performance limitations:

- The planning system failed to identify available narrow corridors, thus preventing the quadrotor from utilizing potentially shorter or more efficient paths to the goal location.
- Trajectory execution involved significant acceleration phases at the beginning of segments and deceleration phases at the end (or at each waypoint), leading to discontinuous velocity/acceleration profiles and reducing overall flight efficiency.
- These frequent and potentially large changes in acceleration adversely affected the accuracy of the state estimation pipeline, a problem anticipated to be exacerbated at higher flight speeds.



(a)



(b)

Fig. 5: (a) path followed by quadrotor. (b) accelerometer readings when executing the trajectory

Vanilla simulator comparison : The generated trajectories were observed to be sub-optimal, resulting in completion times significantly longer than potentially achievable with a more optimized approach. Furthermore, accurate tracking of these trajectories proved increasingly challenging, particularly at higher operational velocities.

C. Failure Cases

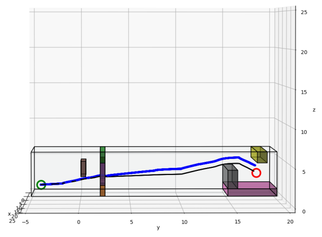


Fig. 6: Local planning for window map

Figure 6 illustrates a potential failure mode where performance degrades over longer trajectories. This may occur due to the accumulation of state estimation error over time, potentially exacerbated by non-smooth characteristics (e.g., high jerk) in the planned path, leading to deviations from the intended trajectory.

Furthermore, Figure 7 highlights challenges associated with high-speed operation. The depicted scenario suggests that at increased velocities, the system exhibits difficulty in reacting sufficiently quickly or accurately to ensure effective obstacle avoidance.

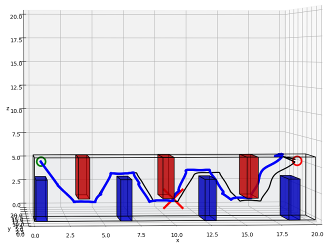


Fig. 7: Local planning for over under at high velocity

D. *Discussion*

Further investigation and development could enhance the local trajectory replanning capabilities. Potential extensions include:

1) Addressing discontinuities (e.g., in velocity or acceleration) observed in the generated trajectory segments, particularly during replanning events, to ensure smoother execution.
2) Implementing or further refining the iterative Quadratic Programming (QP) trajectory generation method, aiming to produce trajectories that enhance both safety margins and overall speed.
3) Investigating the underlying reasons why the current pathfinding or planning approach fails to utilize narrow corridors, potentially examining aspects like map resolution, cost function design, or search heuristics.

E. *Conclusion*

This project successfully integrated VIO-based state estimation with trajectory planning and control modules, achieving the goal of autonomous quadcopter flight predicated on onboard sensor feedback rather than ground truth. An iterative approach combining Quadratic Programming (QP) for smooth, minimum-jerk trajectory generation with collision checking and trajectory regeneration proved particularly effective. This methodology facilitated dynamically feasible flight, enabled successful navigation even at relatively high speeds within the simulated environment, and resulted in strong quantitative performance. While demonstrating significant success for a global planner, opportunities for future enhancement were identified when using a local planner, including improving trajectory time-optimality and ensuring reliable utilization of narrow environmental corridors.

Appendix

QP Formulation

$$C_0^{(k)} = W_k \tag{1}$$

$$C_0^{(k)} + C_1^{(k)}T + C_2^{(k)}T^2 + C_3^{(k)}T^3 \tag{2}$$

$$+C_4^{(k)}T^4 + C_5^{(k)}T^5 = W_{k+1} \tag{3}$$

$$C_1^{(k)} + 2C_2^{(k)}T + 3C_3^{(k)}T^2 \tag{4}$$

$$+4C_4^{(k)}T^3 + 5C_5^{(k)}T^4 = C_1^{(k+1)} \tag{5}$$

$$2C_2^{(k)} + 6C_3^{(k)}T \tag{6}$$

$$+12C_4^{(k)}T^2 + 20C_5^{(k)}T^3 = 2C_2^{(k+1)} \tag{7}$$

$$\tag{8}$$

$$\text{Minimize} \begin{bmatrix} C_5^{(k)} & C_4^{(k)} & C_3^{(k)} \end{bmatrix} \begin{bmatrix} 720T^5 & 360T^4 & 120T^3 \\ 360T^4 & 192T^3 & 72T^2 \\ 120T^3 & 72T^2 & 36T \end{bmatrix} \begin{bmatrix} C_5^{(k)} \\ C_4^{(k)} \\ C_3^{(k)} \end{bmatrix} \tag{9}$$

k refers to segment between two waypoints. W_k is the start position and W_{k+1} is the end position. $C_i^{(k)}$ are the coefficients corresponding to the kth segment.