Minimum-Energy Trajectory Planning for a Planar Drone Using an Augmented Lagrangian Method

Rihan Doshi   Saanvi Choudhary   Veda Chandavolu

1. Problem Statement and Motivation

Autonomous drones operating in real-world environments must compute safe, efficient, and dynamically feasible trajectories. Such trajectories must satisfy the physical constraints of drone motion while avoiding obstacles and minimizing energy consumption.

The objective of this project is to design a trajectory planner for a planar drone that:

- starts at a given initial state,
- reaches a desired final state,
- avoids circular obstacles,
- respects discrete-time motion dynamics,
- minimizes total control effort.

This leads to a constrained nonlinear optimization problem involving equality constraints (dynamics, boundary conditions) and inequality constraints (obstacle avoidance). We solve this using an Augmented Lagrangian-based method implemented entirely in NumPy, without relying on external nonlinear solvers.

2. Mathematical Formulation

2.1. State and Control Variables

We describe the planar drone using the state:

$$x = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}, \qquad u = \begin{bmatrix} a_x \\ a_y \end{bmatrix},$$

where p_x, p_y denote position, v_x, v_y velocity, and a_x, a_y the control accelerations.

2.2. Continuous-Time Dynamics

The drone obeys the point-mass equations:

$$\dot{p}_x = v_x, \quad \dot{p}_y = v_y,$$

$$\dot{v}_x = a_x, \quad \dot{v}_y = a_y.$$

This can be written compactly as:

$$\dot{x} = A_c x + B_c u,$$

with

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

2.3. Derivation of Discrete-Time Matrices A and B

Using zero-order hold with timestep Δt, we integrate the double integrator:

$$p_{k+1} = p_k + v_k \Delta t + \tfrac{1}{2} a_k \Delta t^2, \qquad v_{k+1} = v_k + a_k \Delta t.$$

Thus we obtain:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}.$$

2.4. Objective Function

We minimize total control effort:

$$J(U) = \sum_{k=0}^{N-1} \|u_k\|^2 = \sum_{k=0}^{N-1} (a_{x,k}^2 + a_{y,k}^2).$$

2.5. Constraints

Dynamics Constraints

$$x_{k+1} - A x_k - B u_k = 0.$$

Initial and Final Constraints

$$x_0 = x_{\text{start}}, \qquad x_N = x_{\text{goal}}.$$

Obstacle Avoidance (Soft) For each obstacle with center c and radius r:

$$h_k = \|p_k - c\| - (r + \text{margin}) \geq 0.$$

2.6. Augmented Lagrangian

The full augmented Lagrangian is:

$$\begin{aligned}\mathcal{L}(X,U,\lambda) = &\sum_{k=0}^{N-1}\|u_k\|^2 + \kappa\sum_{k=0}^{N}(\max(0,-h_k))^2 \\ &+ \sum_{k=0}^{N-1}\left[\lambda_k^\top(x_{k+1}-Ax_k-Bu_k) + \rho\|x_{k+1}-Ax_k-Bu_k\|^2\right] \\ &+ \lambda_{\text{start}}^\top(x_0-x_{\text{start}}) + \rho\|x_0-x_{\text{start}}\|^2 \\ &+ \lambda_{\text{goal}}^\top(x_N-x_{\text{goal}}) + \rho\|x_N-x_{\text{goal}}\|^2.\end{aligned}$$

3. Methodology and Solver Design

The optimization is solved using two nested loops.

3.1. Outer Loop: Multiplier and Penalty Updates

After obtaining X and U:

$$\lambda^{(m+1)} = \lambda^{(m)} + 2\rho\, c(X,U),$$

and ρ is increased if constraints remain violated.

3.2. Inner Loop: Unconstrained Minimization

We minimize $\mathcal{L}(X,U,\lambda)$ with respect to X and U using:

- numerical gradients (finite difference),
- normalized gradient-descent steps,
- flattened variable representation.

3.3. Obstacle Penalty

Penalty added for violations:

$$\kappa\cdot(\max(0,-h_k))^2.$$

This gives smooth avoidance behavior.

4. Results, Analysis, and Discussion

4.1. Trajectory Characteristics

The solver produces trajectories that:

- smoothly curve around obstacles,
- minimize control effort,
- satisfy start and goal boundary conditions,
- respect discrete-time dynamics.

4.2. Convergence Behavior

Dynamics and boundary constraints converge rapidly. Obstacle avoidance depends on tuning κ; larger values yield stricter avoidance.

4.3. Parameter Sensitivity

- High κ improves avoidance but may slow convergence.
- Higher ρ enforces constraints more strictly.
- More inner iterations improve minimization accuracy.
- Step size controls stability of descent.

4.4. Limitations

- Numerical gradients are computationally expensive for large N.
- Soft obstacle penalties do not guarantee strict separation.
- Model is a simplified double integrator.

5. Conclusion

We developed a complete trajectory optimization framework for a planar drone using the Augmented Lagrangian method. The solver produces smooth, energy-efficient, dynamically feasible trajectories that avoid obstacles while satisfying all constraints.
The implementation is modular, transparent, and educational, relying only on NumPy. This framework can be extended to more realistic drone dynamics, 3D motion, or real-time MPC.