TRAJECTORY GENERATION

1. **Preloaded from drone waypoints**
2. **Preloaded from Bezier Trajectories (vis.)**
3. **Realtime using spline Trajectories (vis.) + RPG Drone Racing Sim**
4. **Integration into Crazyflie**

This tutorial is meant to cover the **generation of trajectories** for autonomous drone flight. Each usecase has different requirements, whether simply to **reach a point**, or to do so in a way that takes into account the **environment and the constraints** we put on the drone. The approach here will therefore start with demo-able results in constrained environments, and then form a more **modular framework** that allow **more responsive**, **and more stable** drone usecases.

First we get a Crazyflie drone up in the air using a list of waypoints. Most applications require this alone. The Crazyflie framework is a **Python library** that can perform automated behaviours, given a set of points. This approach has been successfully coupled to **an optic flow system** that you can learn about in the drone maintenance section.

Waypoint-following however, can only guarantee, well, reaching the waypoints. In the interest of stability, we might require a constant velocity, or velocity/acceleration threshholds. Using Bezier trajectories, we can optimise a list of waypoints to a clearer route. This approach has been used with **an optic flow system** as well as **external localization technology** (Optitrack).

This enriches the types of trajectories that can be flown. But aside from **startling demoes**, the trajectory is set and the drone is not responsive to its environment. Can a drone avoid crashes easier? An approach has been derived from self-driving cars called **spline trajectories**, and in short, this gives the robot options on changing its trajectory on the go. As you can imagine, this adds a complex layer of decision-making which can be incorporated into a **behaviour planner**. This theory is explored first with a self-driving car simulator.

Algorithms aside, the trajectory behaviour planner needs to work in conjunction with the drone's localization process. We choose to do this **via the swarm PC** since **the external localization system** is also obtained on the PC**.** A protocol needs to be developed that sends the drones their trajectories in coordination with input information, and this infrastructure is explored in the crazyswarm tutorial a little later.

Bibliography

Traditionally, robotics has **separated the generation of a trajectory from trajectory following**, and a neat research paper explains the variety of algorithms that can be used for this purpose. W. Hoenig, author of this paper, is incidentally the main contributor on the Crazyswarm github, a **framework for drone swarms** that offers the necessary infrastructure for **following simultaneous trajectories** (I write about it a little later).

The **use of spline trajectories** is commonplace in robotics, and a drone racing platform finds how to exploit **spline trajectories in 6 DOF**. The RPG drone racing framework explores this and other trajectory following algorithms on their github.

Going beyond with trajectory following, there has been a serious shortcoming in the last few years of drone development, and that is the poor localization offered by Bayesian filters. Research laboratories frequently use an external localization system to circumvent this. I used the Optitrack system for this purpose.

A recent paper trains a drone to follow a trajectory **with its VIO-IMU** in **zero-shot transfer learning**: this approach comes to show the progress autonomous drones have made over the last few years.