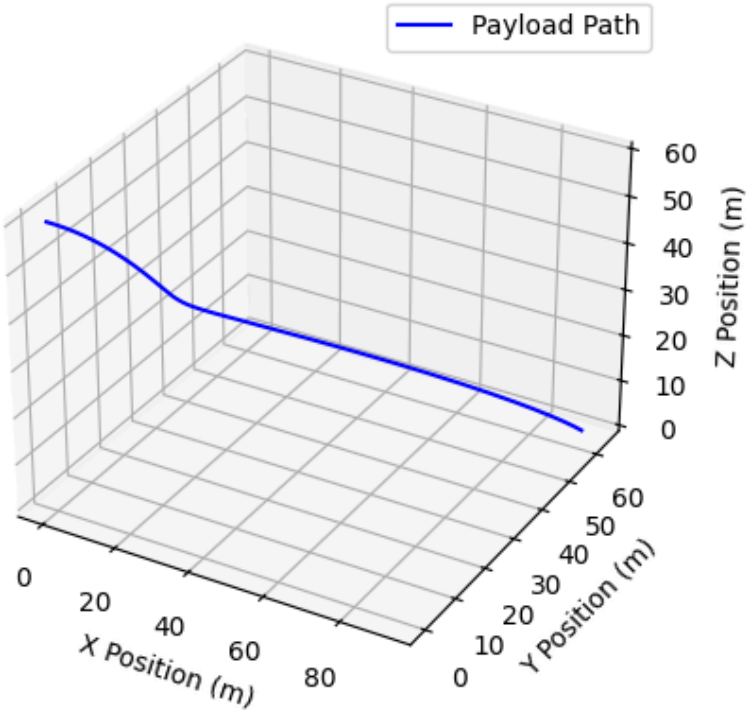
The optimal drop position algorithm

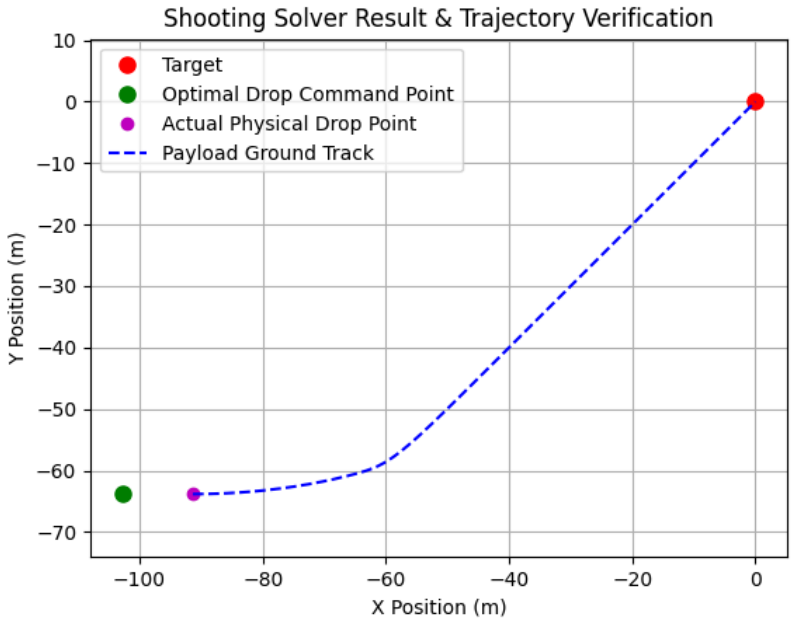
When the airplane computer vision systems detect a target object (e.g. mannequin), the GPS coordinates of the object are calculated based on its relative position to the plane. This requires parameters such as camera resolution, the current plane altitude, and its GPS position. The knowledge of the position still leaves us with the question of how to get the payload to the target with precision. For that part of the mission, an algorithm was created and fine-tuned in field conditions.



Example 3D trajectory plot for payload drop.

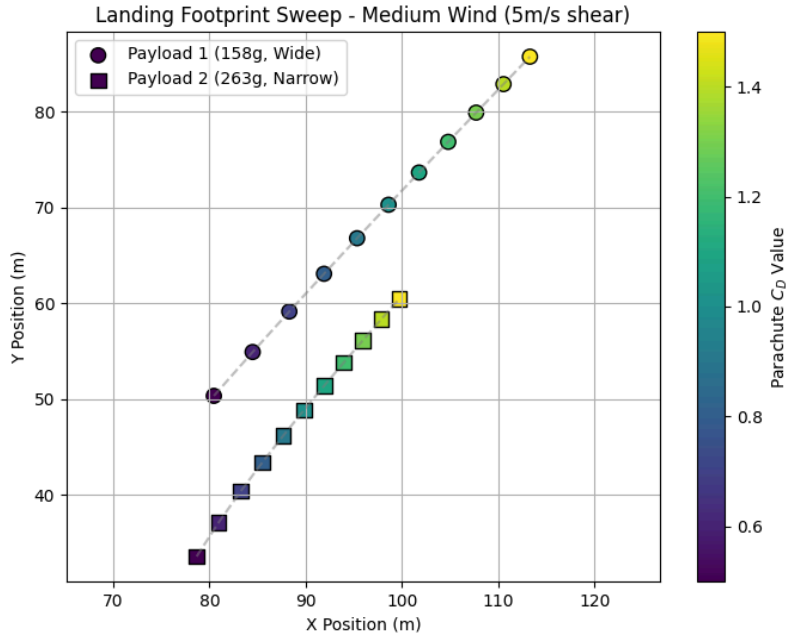
The algorithm is based on the numerical simulation of a payload drop. The solver uses Runge-Kutta method (RK4) for finding the trajectory. The parameters are:

- payload mass,
- surface areas of the parachute and object,
- aerodynamic coefficients,
- servo delay,
- parachute opening time,
- air density,
- wind model,
- initial vector state (position and velocity).



Example of the solver working with side wind. At x=-60m, the wind influence becomes clearly visible as the parachute opens and the velocity is reduced.

The solver, when the wind model assumes uniform wind strength across the terrain x and y coordinates (the wind speed is a function of altitude), uses a simple displacement strategy to find the optimal drop point. If the wind conditions near the ground are significantly affected by the terrain shape (forest, hills etc.) and a more precise model is available, an optimization strategy is used such as the scipy.optimize.minimize function.



A sweep analysis of the parachute aerodynamic drag coefficient and payload type for landing position potential error.

Initial simulations showed that the critical parameters are the aerodynamic coefficient of the parachute used and an accurate wind model. The first one could be found experimentally through practical tests, the latter depends heavily on wind stability at the competition site.

When it comes to the solver implementation, the Python programming language was used with the NumPy library for handling state vector calculations, which were done in a way taking advantage of quick operations that this package allows on vectors.

Apart from parachute aerodynamic coefficient and wind speed, there are important variables to consider. Servo delay, that's time between onboard computer sending signal initiating drop to the payload physically leaving the plane. As the cruising speed can be around 23m/s even slight delay is equivalent to meters of potential mistake. Further the parachute opening time, when the precise estimation of aerodynamic properties is very hard, is simulated by assuming linear drag force increase. Air density is calculated based on temperature, pressure and humidity, or it can be taken from the weather forecast at the competition site. Potential errors also might depend on velocity and altitude sensors precision as well as the correct placement of the target GPS coordinates.