Development of a Software Solution for Precision Landing Drones

Joel Happ, Jerome Gobeil, Taylor Noel, Maral Rasuli Arasi, Niyousha Raeesinejad
Department of Electrical and Software Engineering, University of Calgary

Connect with us!

Motivation

As flying drones grow in popularity and usage, there is increasing demand for technology that enables drones to perform intelligent landings. GPS can guide drones to a general area, but it lacks enough precision to land on a specific platform like a launchpad, a car, or a boat. High-precision landings are desirable as they would allow for faster turn-around time when launching and recovering a drone, allow for landing on moving platforms and reduce the risk of collisions when landing. The objective of our project is to solve the problem of manually having to land drones on stationary and moving targets by creating a precision landing program. We have created a software solution that helps a drone land using distance detection and precision landing on stationary targets. To validate our solution, we have conducted tests inside an industry-standard simulation program and assembled a physical drone as well.

Drone Assembly

Several hardware components were used to assemble the drone: **Pixhawk**; **Raspberry Pi 3; Raspberry Pi Camera**; **RC Controller**; **Radio Receiver**; **Mamba Board F50 MK22** to connect the Pixhawk and the Raspberry Pi to the battery and; 2 **batteries** (one used to power the drone and the other a backup).

The drone kit used includes the body of the drone and motors. For additional structural support, we designed and 3D printed the following parts:

- A **gimbal** to hold the camera perpendicular to the drone.
- 4 **landing gears** for stable landing.
- 2 **mounts** for the Pixhawk and Raspberry Pi.

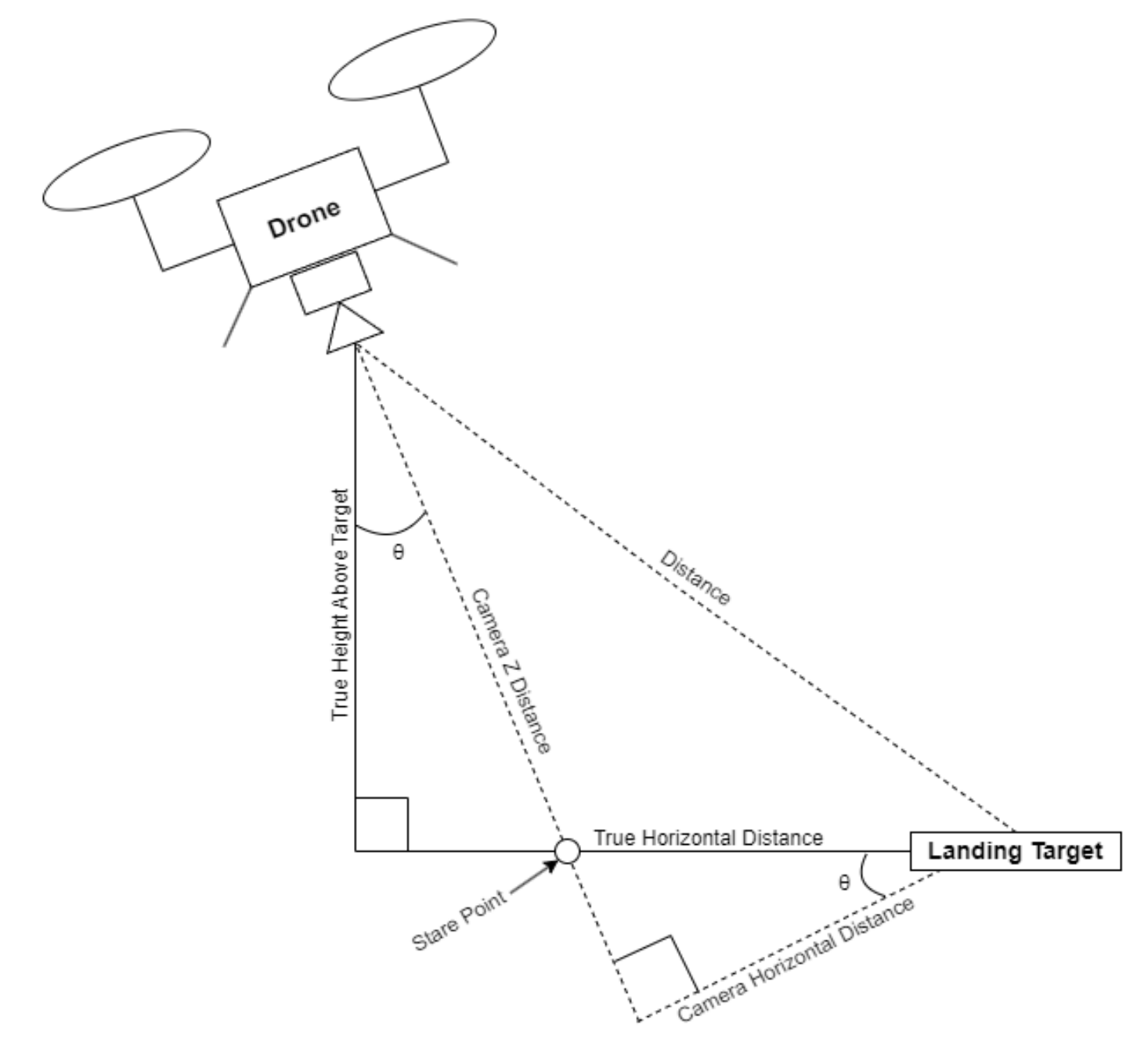
Due to time constraints and challenges of assembling a fightable drone, an **IRIS drone** provided by our sponsor – Lockheed Martin – has been used to run test flights with our integrated solution.

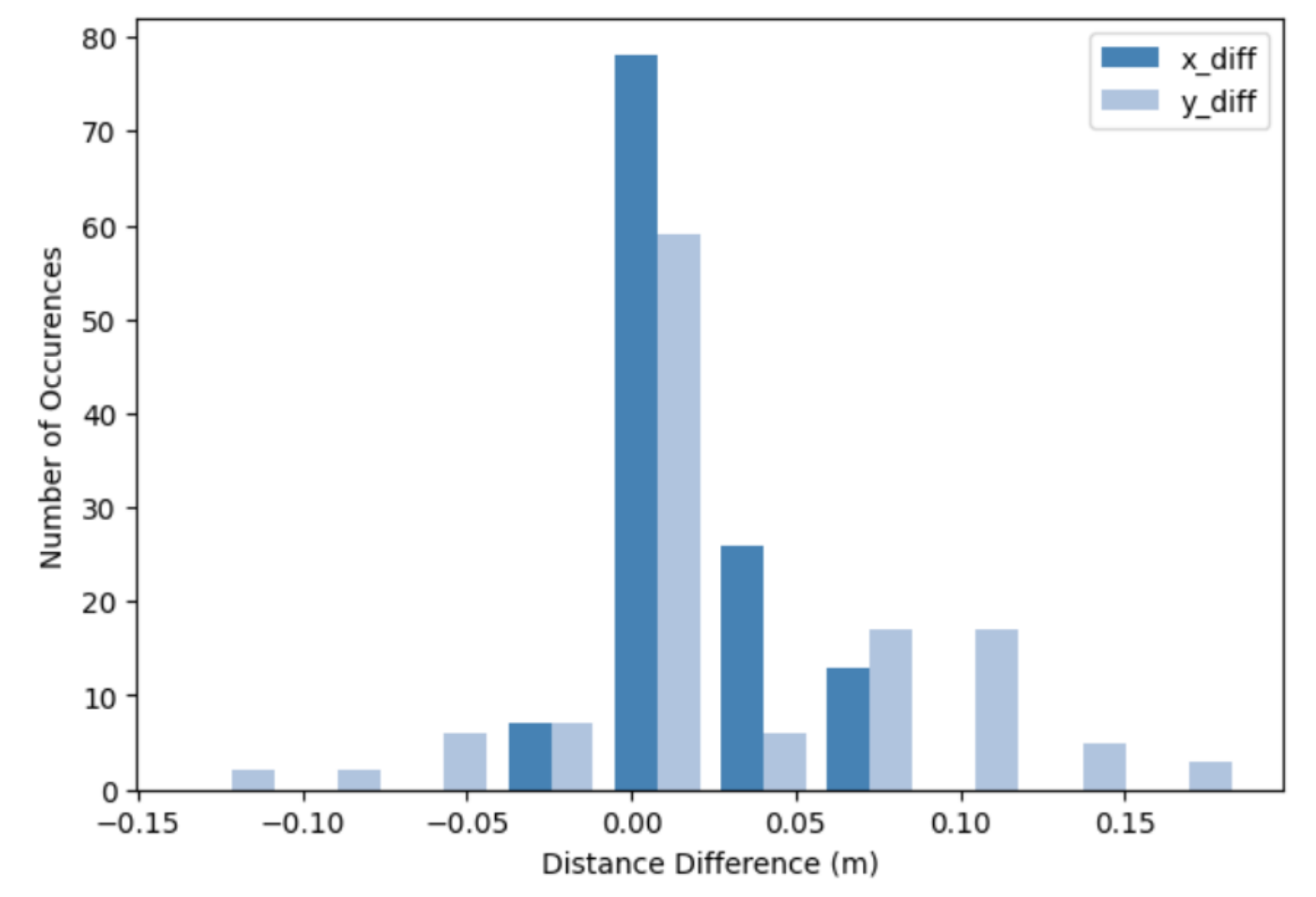
Computer Vision

Our computer vision solution helps the drone detect the correct location to land. The landing platform is demarked by an ArUco code which is detected by our computer vision algorithm. Then, based on the known size of the code and the focal length of the camera, we calculated the distance to the center of the code as well as its relative x and y displacement. Afterwards, we adjust these measurements to consider the rotation of the drone, which gives us a final measurement of the location of the landing platform relative to the drone (as shown in the figure below).

On average, our measurements have a 5% margin of error, and this yields good results in practice as the drone reruns the computer vision algorithm at a high frequency so that the position is continually updated as the drone moves closer towards its target. The following tools were leveraged for computer vision solution implementation:

- **OpenCV**: A Python library for camera calibration, ArUco Code detection, and functions to get the normal vector of the ArUco Code.
- **ArUco** Codes: Used to detect landing targets.





Difference Between True (2m) and Measured X and Y Distances Histogram - Constant Simulation Distance

OpenCV

Simulation Testing

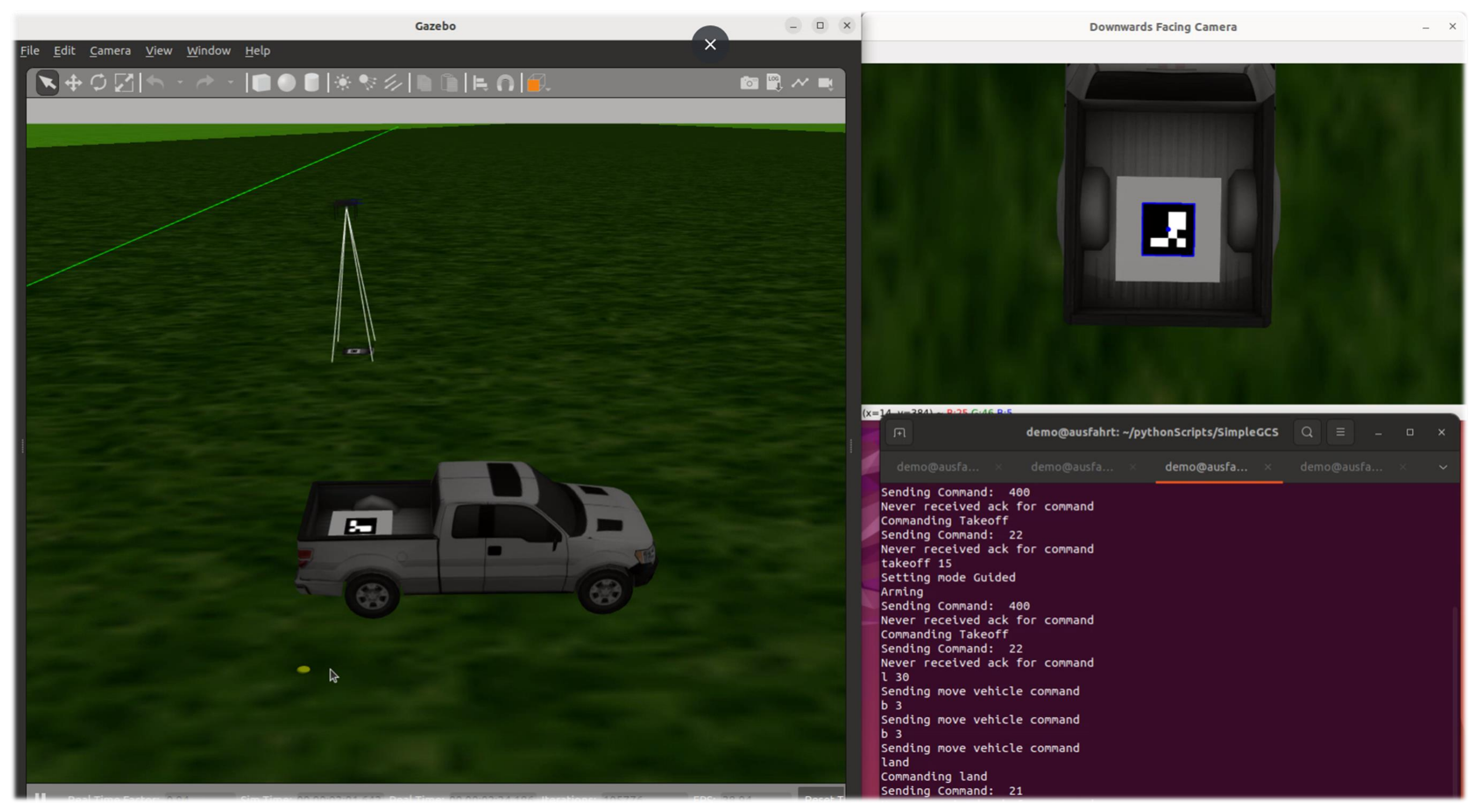
Software simulation was leveraged to validate our computer vision-based precision landing solution. While not a full replacement for real world tests, software simulation has many advantages:

- It is low risk; In the event of a crash, no damage is done to the drone, operators, or the environment.
- It makes performing and repeating tests very easy as the tests are not affected by factors such as weather, drone battery, and space availability. It is also much easier to compare results.
- It allows for much faster prototyping as it is faster to set up than an actual drone.

The following tools were leveraged for simulation testing:

- **Gazebo**: An industry standard simulation software which simulates a virtual environment.
- **Ardupilot** and **SITL**: An open-source autopilot SITL allows for simulating a ArduPilot autopilot.
- **MissionPlanner**: A popular ground control station used to fly ArduPilot drones.
- Python scripts were also used to fly the drone, receive simulated video output, and configure the drone and virtual environments.

The figure below presents the simulated test flights. On the left is the Gazebo window with the simulated drone and a landing target in the back of a pickup truck. The top right window displays our simulated video output displaying the back of the pickup truck and the landing target highlighted in blue showing that it is detected. The bottom right terminal window is running various tools mentioned above.



```
Sending Command:  400
Never received ack for command
Commanding Takeoff
Sending Command:  22
Never received ack for command
takeoff 15
Setting mode Guided
Arming
Sending Command:  400
Never received ack for command
Commanding Takeoff
Sending Command:  22
Never received ack for command
l 30
Sending move vehicle command
b 3
Sending move vehicle command
b 3
Sending move vehicle command
land
Commanding land
Sending Command:  21
```