THE ANDREW & ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

TECHNION Israel Institute of Technology

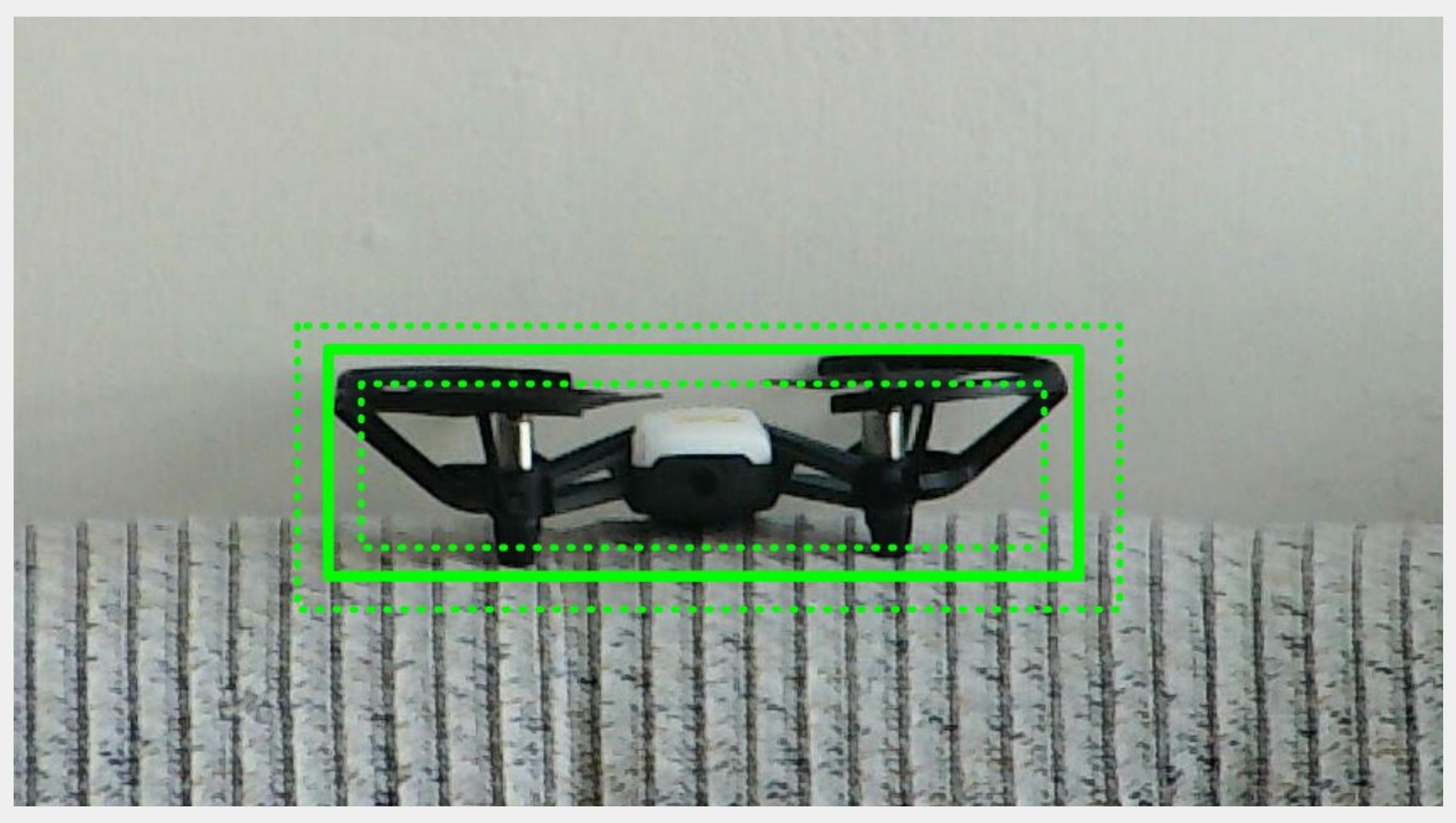
Control Robotics and Machine Learning Laboratory

Controlling A Gimbal to Follow A Drone, Handling Input Delay with Prediction

A work by Amir Sarig, Idan Anner, supervised by Ilan Rusnak at CRML Laboratory, Technion Institute of Technology, October 2020

Introduction

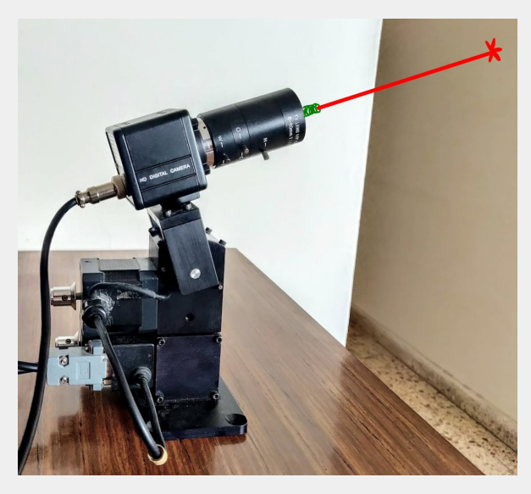
Our system consists of a camera mounted on an open-loop stepper pan & tilt unit. The camera captures a frame and feeds it to a DNN, which gives us the pixel coordinates of a drone within it. These coordinates are the **input** to our system, and our goal is to control the motors so that the target is centered in the camera.



Implementing a good target following system may be useful for various applications such as:

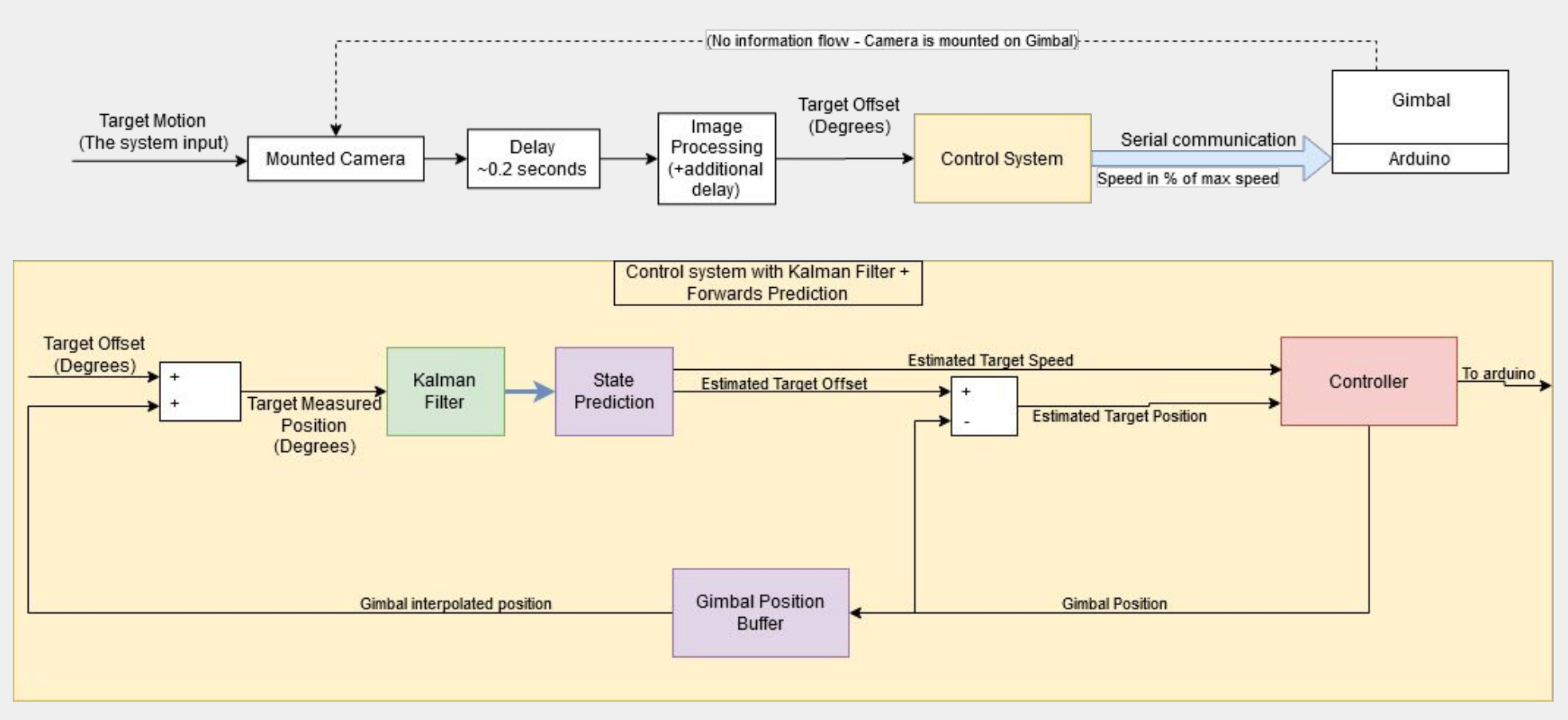
1. recording fast moving object like birds, bats or insects.
2. Shooting down military invasive targets
3. Maintaining beam based communication with a moving system

The Control



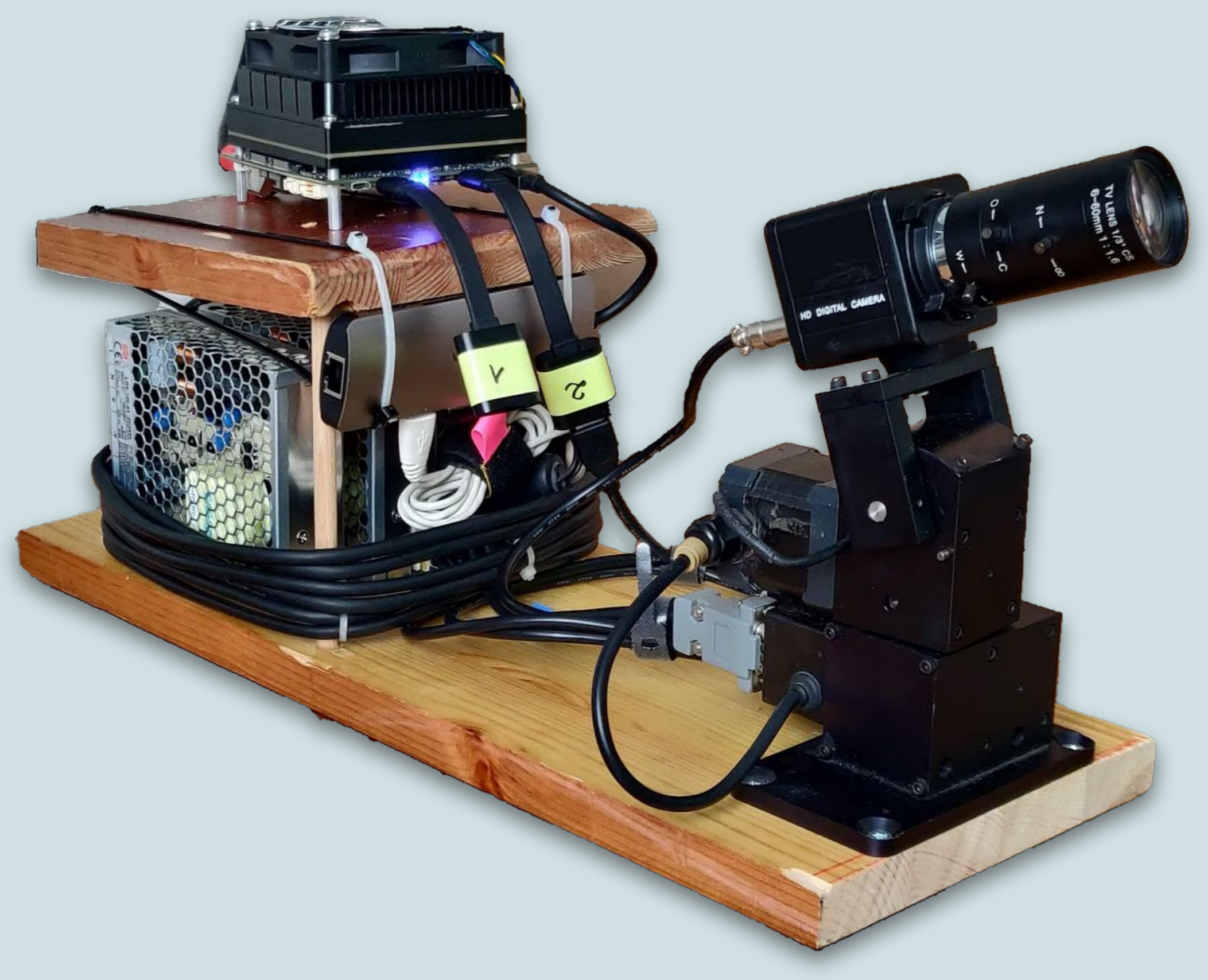
Our control loop, written in Python includes:

- Kalman Filter
- PID Controller
- Predictor



It takes into account::

- Discrete, variable frame rate
- Discrete motor step size
- Open-Loop Control
- Software & Hardware delays



Handling Long Input Delay

Our camera suffers from ~0.17s E2E (input) delay causes 2 suveer problems:

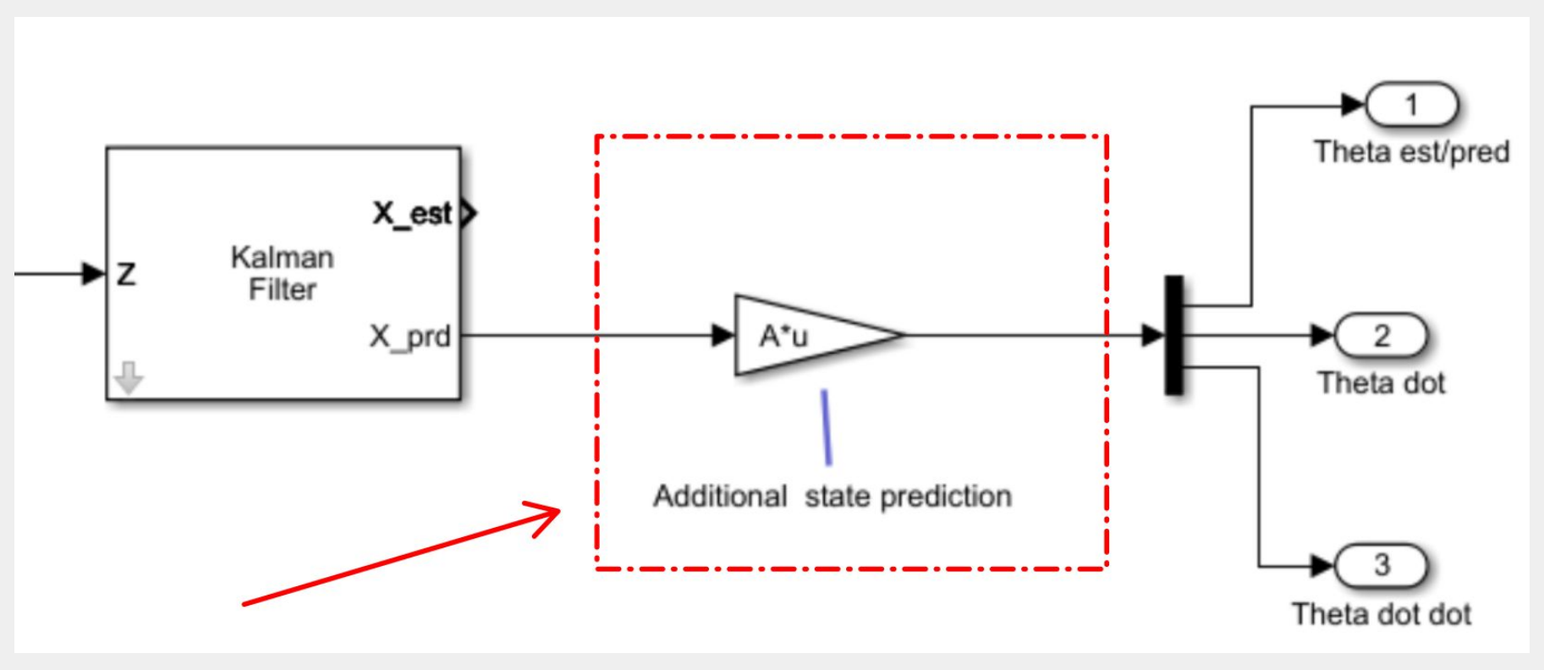
1. Target position is based on target offset and gimbal position. As only the offset is affected by the delay, the target position we calculate is false, and not even delayed as we expect.

2. Even if our target position is known, it is delayed and so the gimbal follows the target in delay

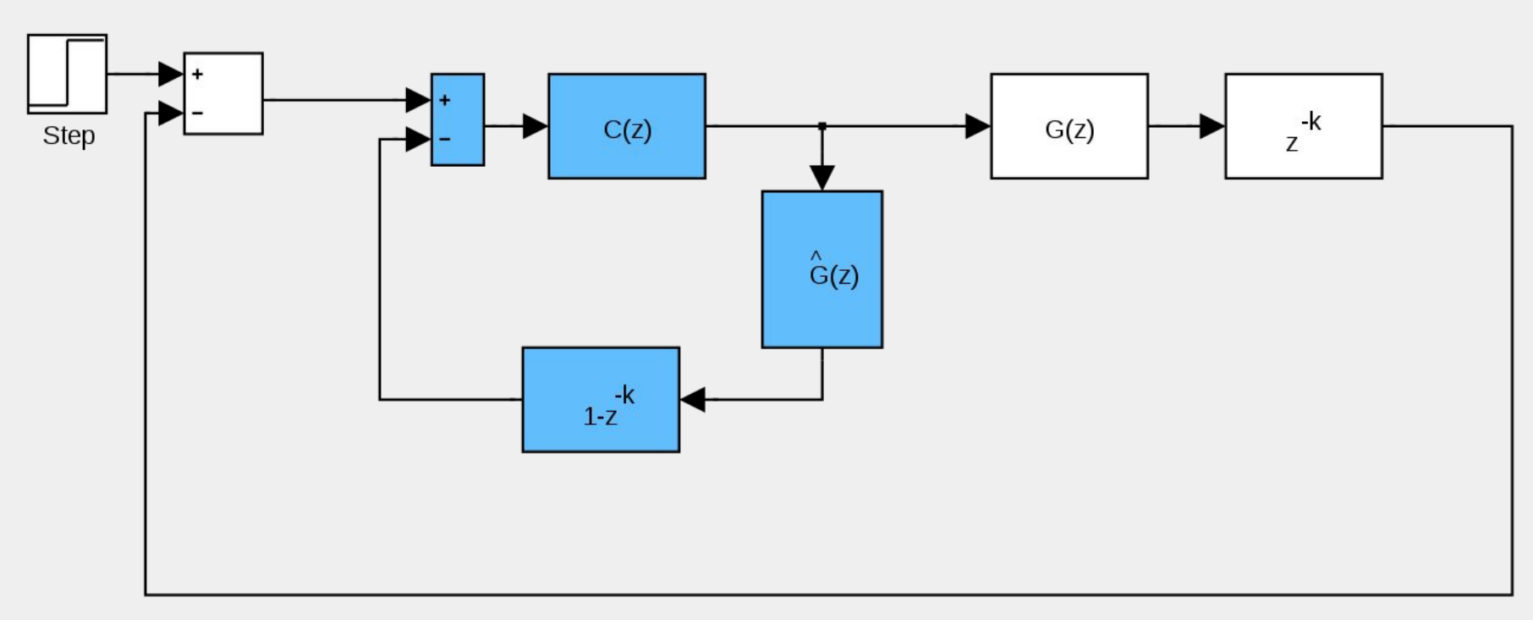
Improve performance using **Forward Prediction:**

1. We add an artificial delay on the gimbal position, so we get a correct target position (but still delayed)

2. We predict the position of the target based on its motion



As an adaptation to what is commonly done with in-system delays (Smith predictor):



Results

Below are some examples of real-life experiments. The next graph logs the following performance of a ramp motion profile.

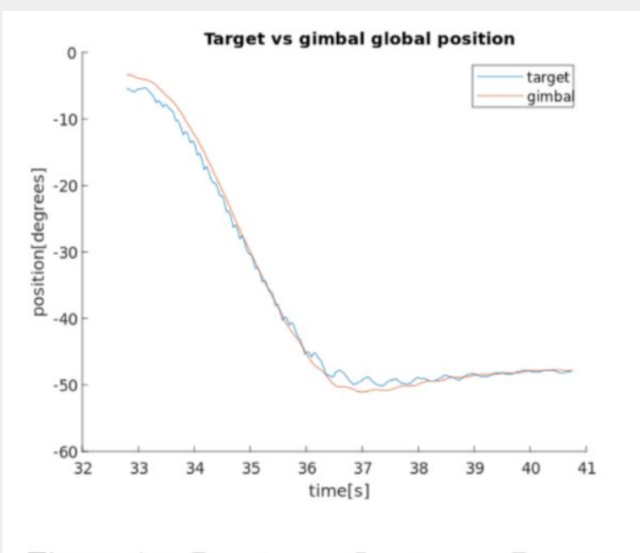


Figure 48: $P = 4, \quad I = 6, \quad D = 0.1$

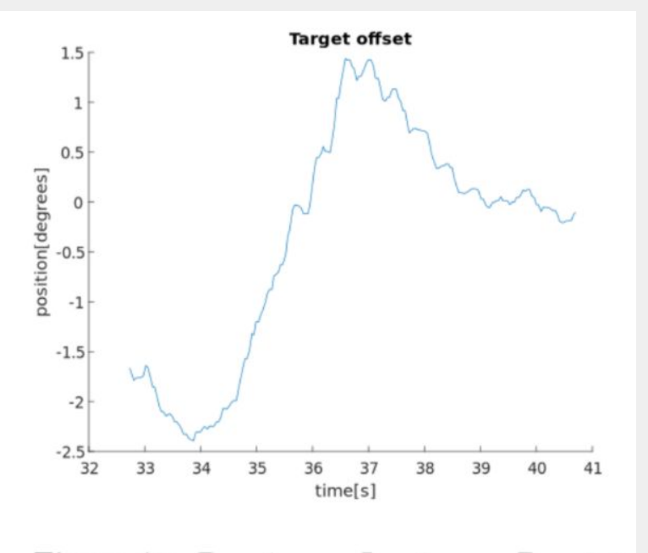


Figure 49: $P = 4, \quad I = 6, \quad D = 0.1$

The next graph describes a back and forth ramp motion profile:

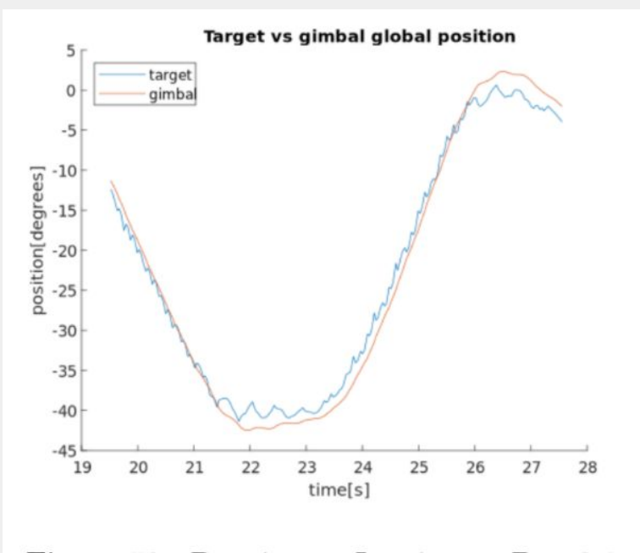


Figure 50: $P = 4, \quad I = 4, \quad D = 0.1$

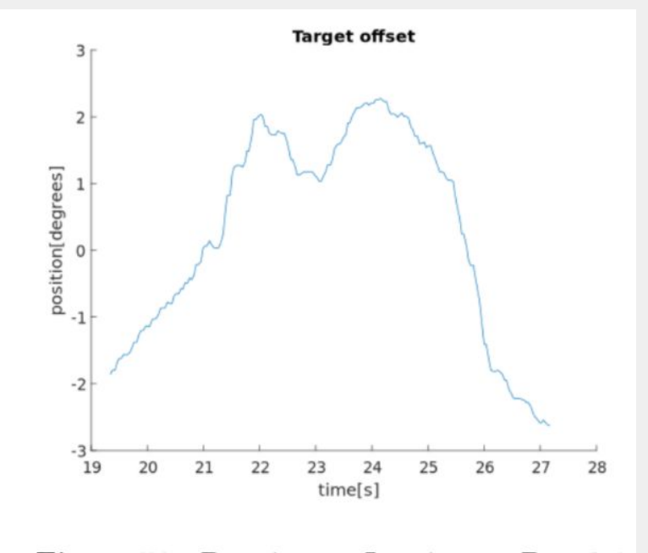


Figure 51: $P = 4, \quad I = 4, \quad D = 0.1$

These graphs were captured by using the best PID values we have found and they present good targeting performance.

Conclusions

Working on this project had lead us to the following insights:

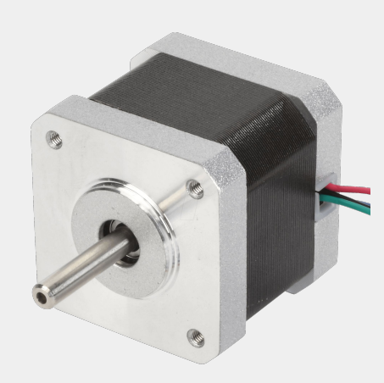
1. Input delay is the bottleneck for good following performance in this project
   a. It cannot be fully overcome using prediction
   b. It's affect is amplified with higher types of the system
2. Working with open-loop system is complex and can cause inaccuracy
3. Relying on relative input value is less stable than absolute input value
4. Simulating a code-based system is not always trivial, at least with Simulink
5. Performing real-life experiments might involve additional challenges and additional noise
6. Traditional Kalman Filter is not suited for compex motion profiles

Video

python™

ARDUINO

NVIDIA



SIMULINK