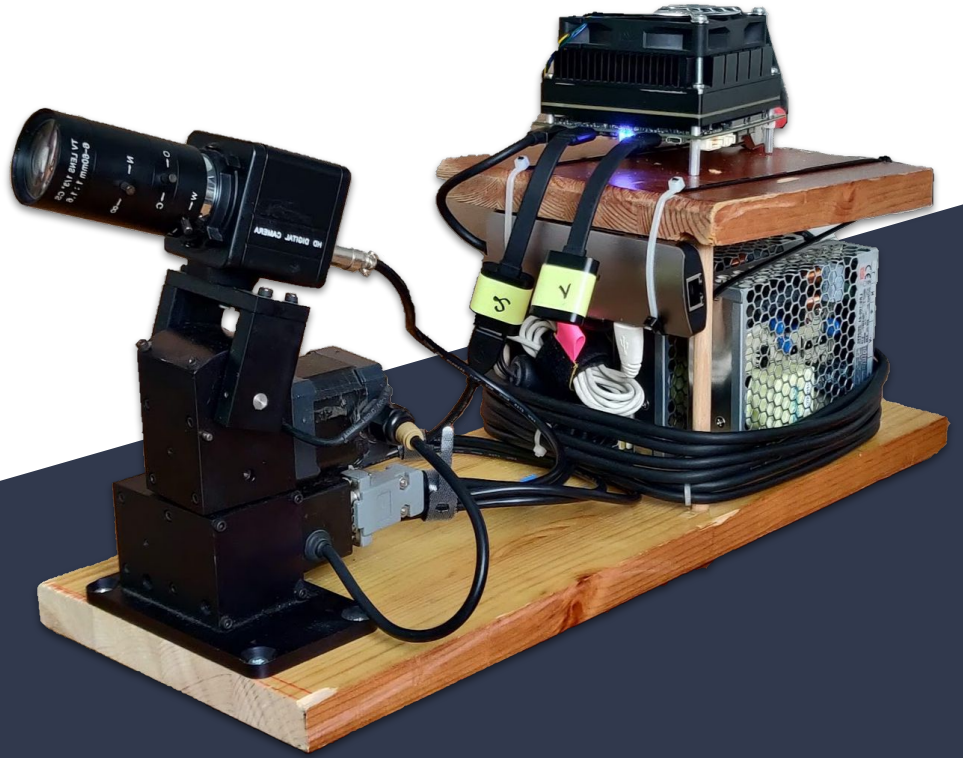
Controlling A Gimbal to Follow A Drone, Handling Input Delay with Prediction

Amir Sarig & Idan Anner
Supervised by Ilan Rusnak

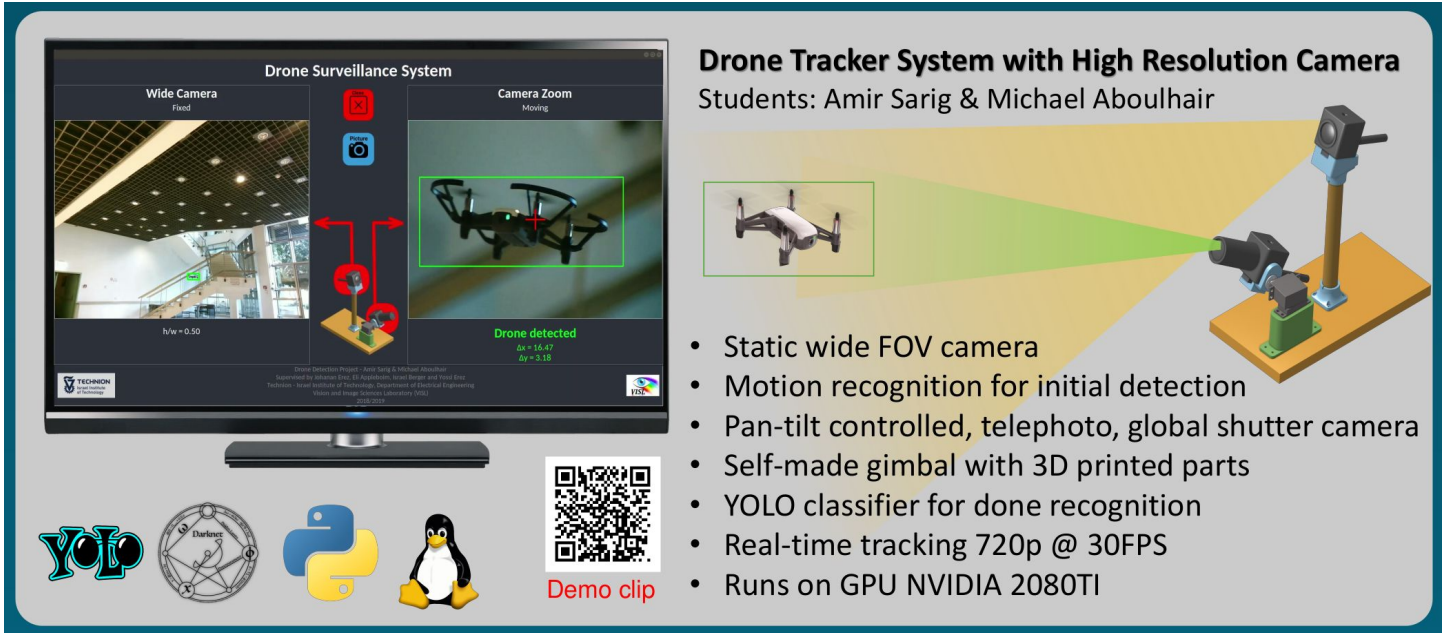


Outline

- Introduction
  - Background & Motivation
  - Objectives
  - Academic Sources
- Hardware Overview
- Block Diagrams
- Integration
  - Driver
  - Arduino
- In-Depth Control Review
  - System Modeling
  - Kalman
  - Controller
  - Forward Prediction
- Performance
- Summary & Conclusion
- Questions

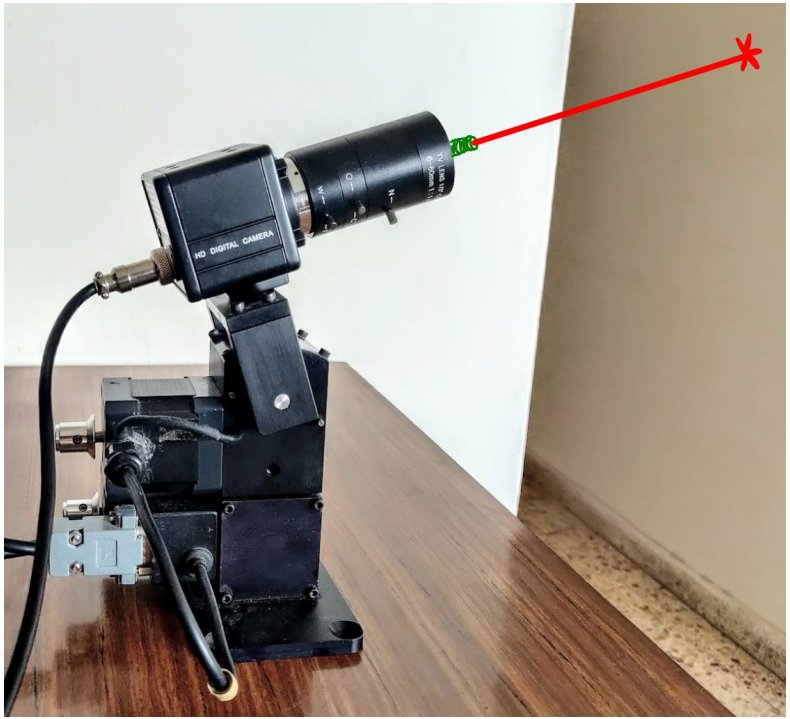
Introduction

Introduction - Background & Motivation



Introduction - Background & Motivation

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



Introduction – Objectives

Operational Objective:

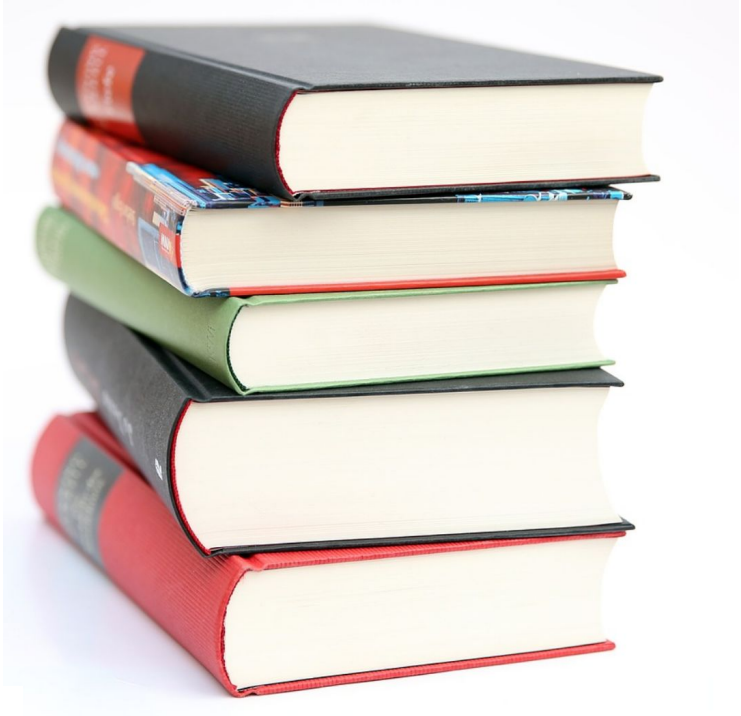
- Point a laser beam to this drone as accurately as possible.

Educational Objective:

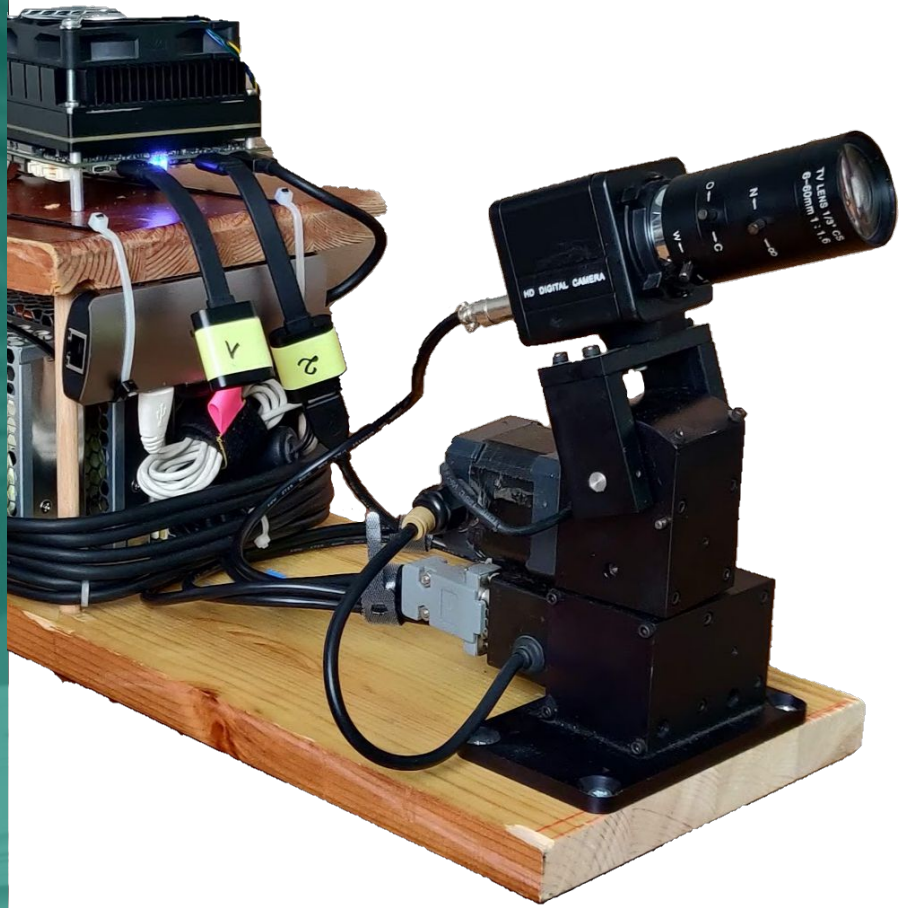
- Understand the main consideration of designing a discrete controlled following system
- Learn the right development workflow of such system
- Feel the main bottlenecks of such a system including noises, delays, amplifications etc.
- Get a sense of the limitations of such a cheap setup

Introduction - Academic Sources

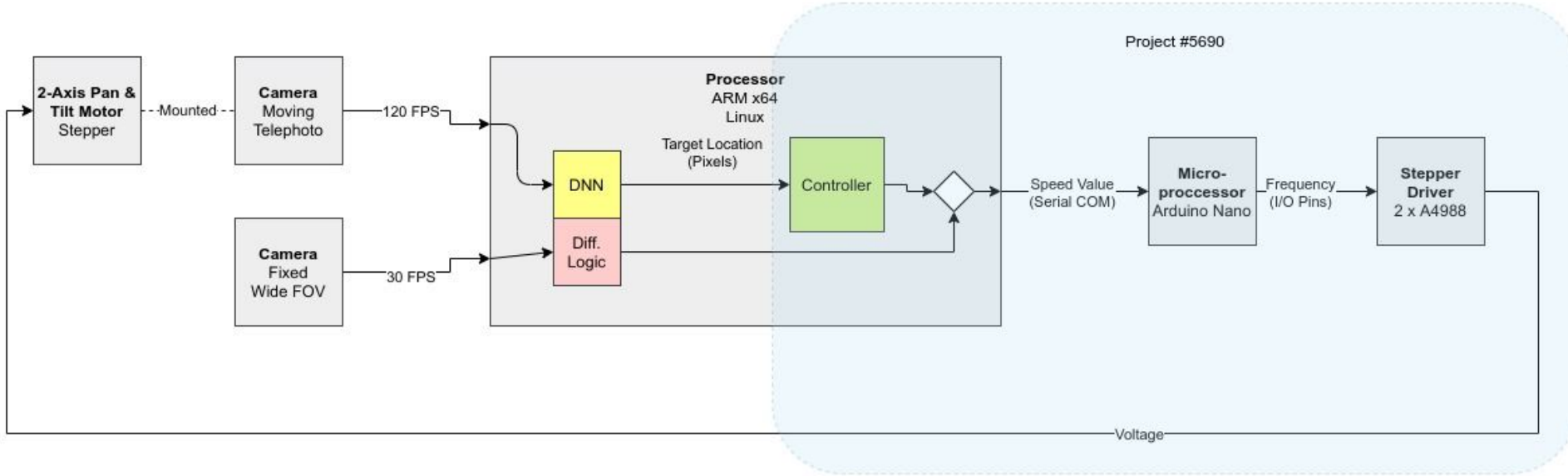
- Direct Versus Indirect Line of Sight (LOS) Stabilization (Article)
- Feedback Control of Dynamic Systems, 8th Edition (Book)
- Technion Control Theory 2 course Tutorials



Hardware Overview



Hardware Overview – Hardware Diagram



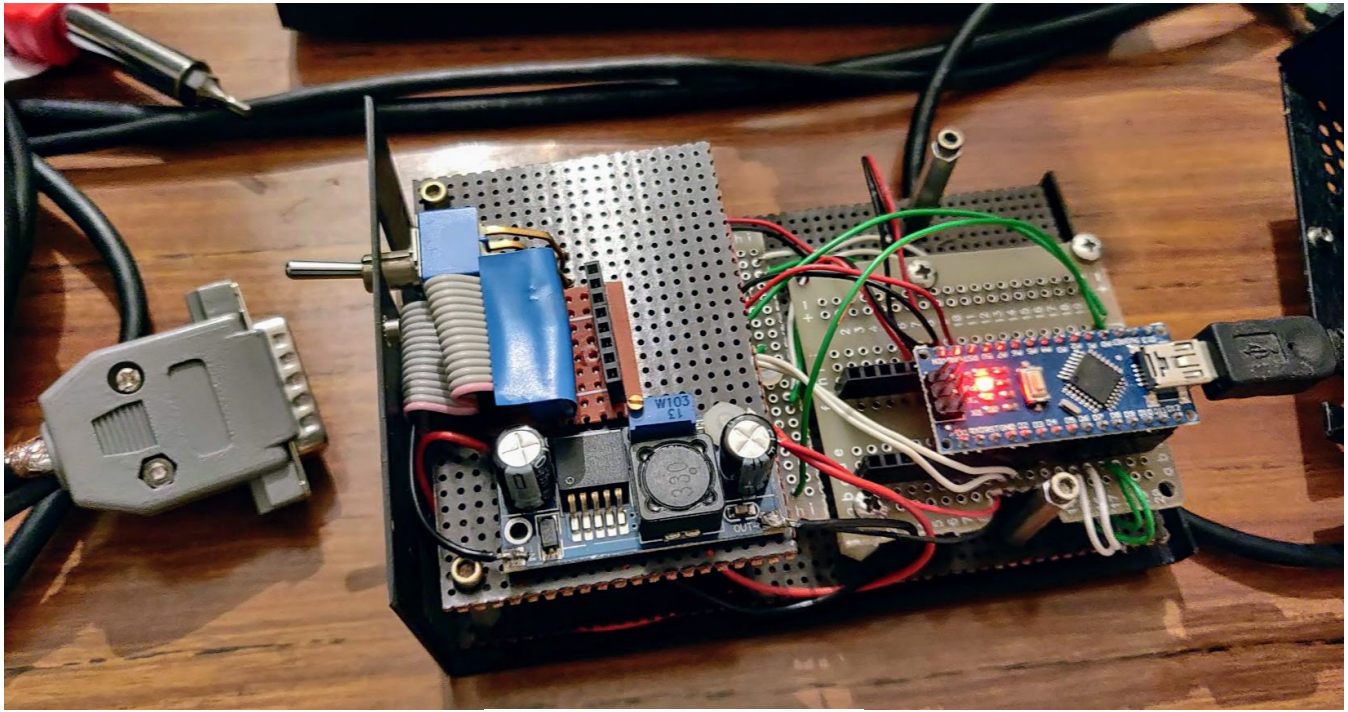
Hardware Overview – The Gimbal

- Model: Flir PTU-46-17.5
- Motors Type: Stepper, open loop (NEMA 17)
- Step size: 1.8 [deg]
- Resolution: 0.05 [deg]
- Max. Speed: 300 [deg/sec]

Hardware Overview – The Camera

- Model: Kayeton 6-60mm High Speed
- Zoom Type: Fixed, Tunable
- Frame Rate: 120 [FPS]
- Resolution: 1280x720 [pixels]
- E2E Delay: [sec] (as measured by us)

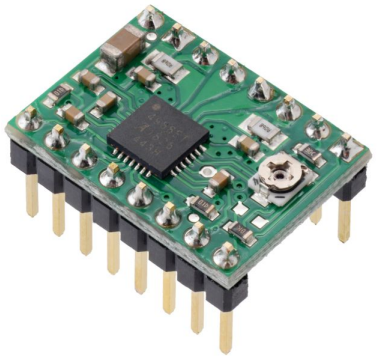
Hardware Overview – The Stepper Driver



Hardware Overview – The Stepper Driver

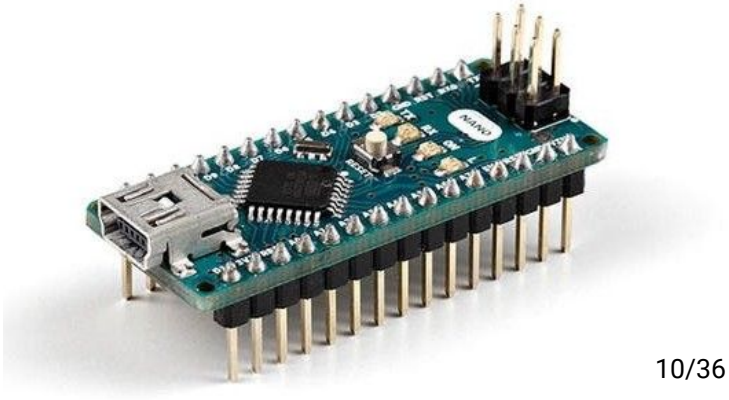
Driver:

- Model: A4988
- Input Voltage: 0 to 35 [V]
- Output Voltage: -2 to 37 [V]
- Microstepping: ½, ¼ , ⅛, 1/16



Microcontroller:

- Model: Arduino Nano
- Clock Speed: 16 [MHz]

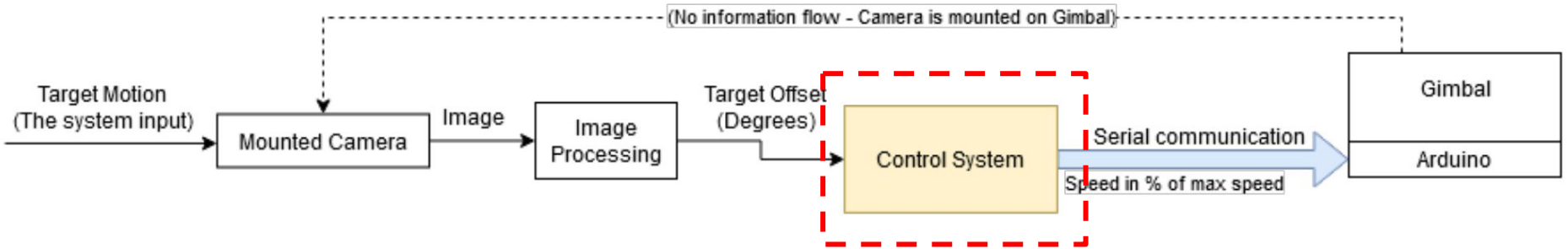


Hardware Overview – Computer

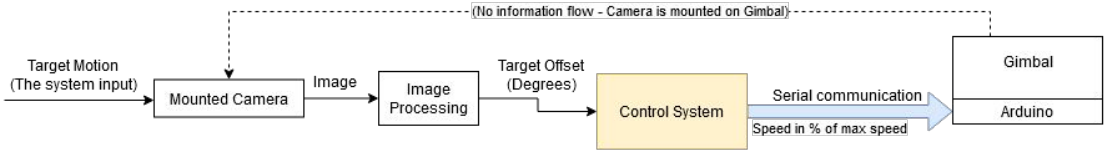
- Model: Nvidia Jetson AGX Xavier
- OS: Linux, Ubuntu (Jetpack)
- CPU: 8-core ARM x64
- GPU: 512-core Volta

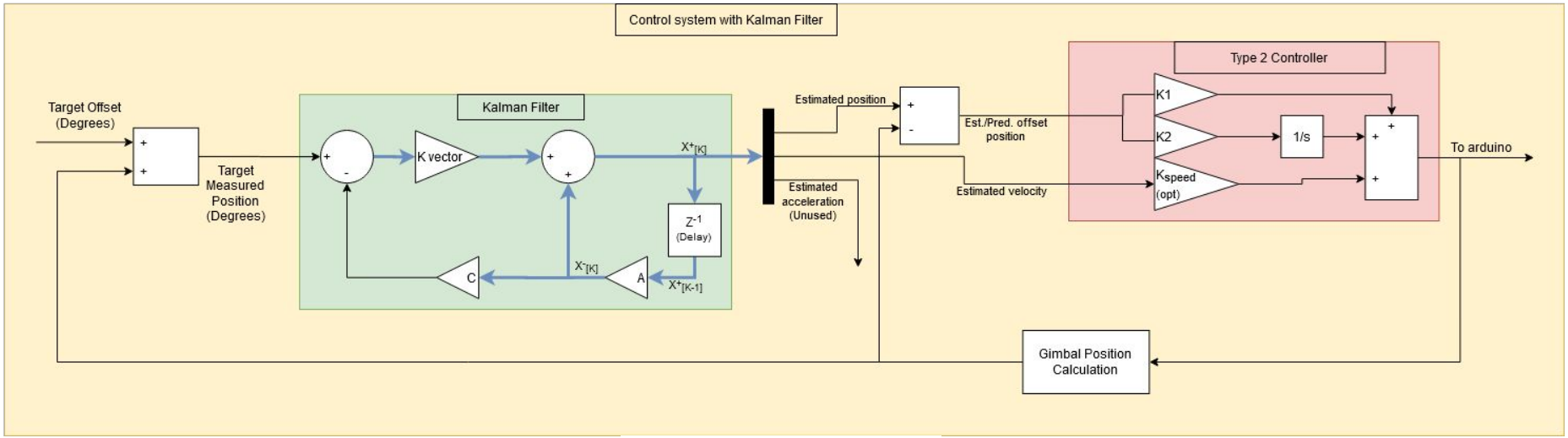
Block Diagrams

Block Diagrams – Top Level



Block Diagrams – Control System



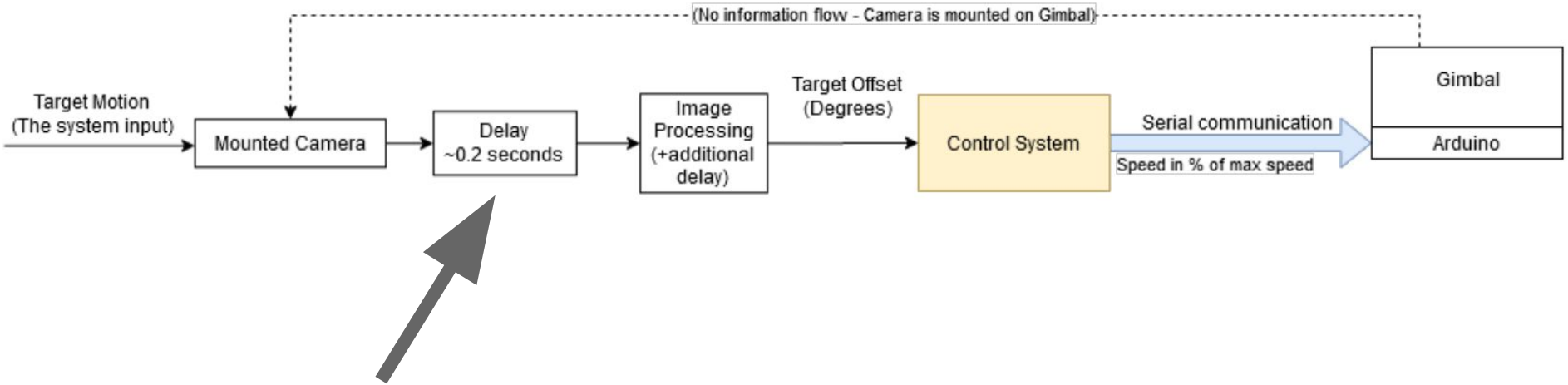


Block Diagrams – E2E Delay

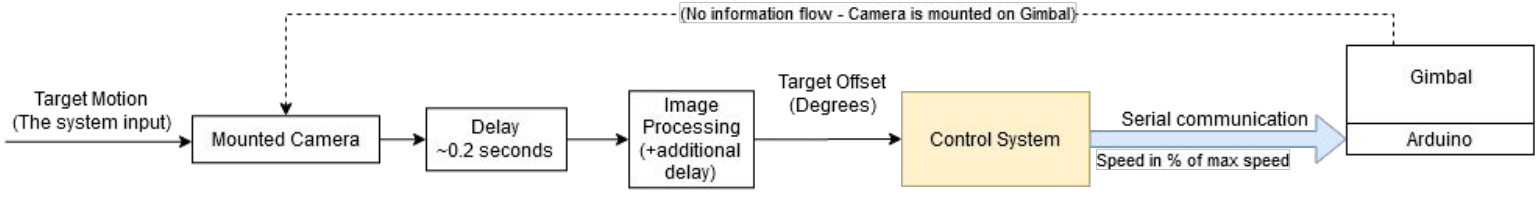
"Glass-to-glass video latency, also called end-to-end (E2E) video delay, is indicating the period of time from when the photons of a visible event pass through the lens glass of the camera till the moment when the nal image is delivered to the glass of the monitor."

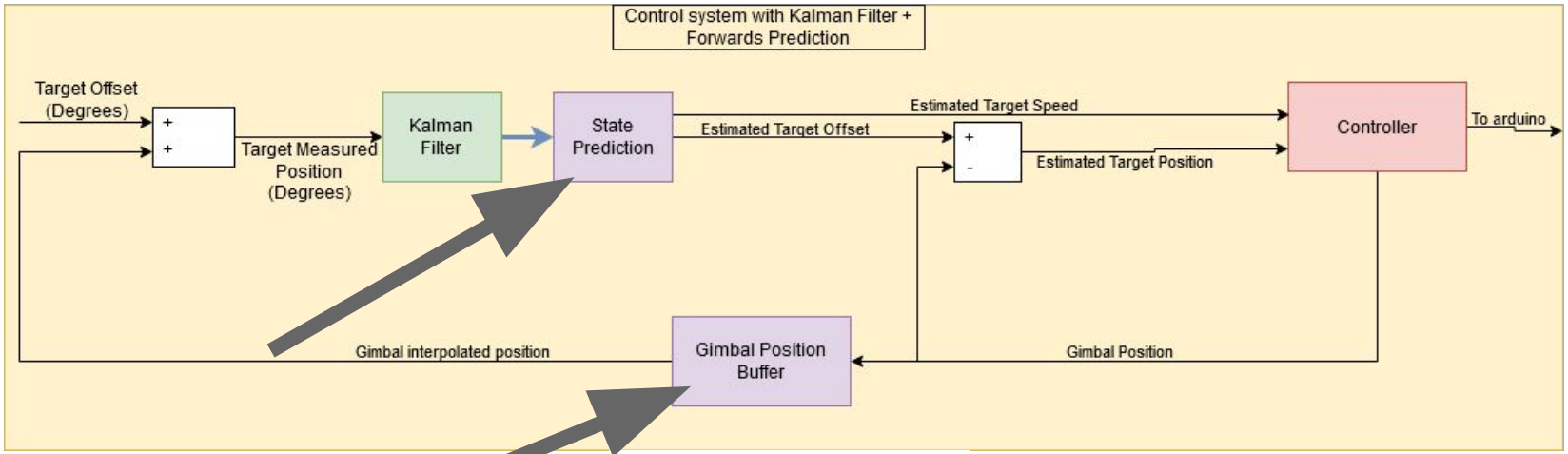
Source: ximea

Block Diagrams – Control System + E2E Delay



Block Diagrams – Control System + E2E Delay





CRM!...
Control Robotics and Machine Learnin

Integration

Integration – Optimizing for the A4988 Driver

- Remindier: The controller has changed
- The PTU-46-17.5 specs are the upper bound for the performance we can get
- The true performance will be determined by the **input voltage** and **microstepping** configuration

Hardware Overview - The Gimbal

- Model: Flir PTU-46-17.5
- Motors Type: Stepper, open loop (NEMA 17)
- Step size: 1.8 [deg]
- Resolution: 0.05 [deg]
- Max. Speed: 300 [deg/sec]

Call out key parts of the UI

Integration – Optimizing for the A4988 Driver

- Remindier: The controller has changed
- The PTU-46-17.5 specs are the upper bound for the performance we can get
- The true performance will be determined by the **input voltage** and **microstepping** configuration

Hardware Overview - The Gimbal

- Model: Flir PTU-46-17.5
- Motors Type: Stepper, open loop (NEMA 17)
- Step size: 1.8 [deg]
- Resolution: 0.05 [deg]
- Max. Speed: 300 [deg/sec]

Call out key parts of the UI

Integration – Optimizing for the A4988 Driver

- Raising input voltage
- Enabling Micro-stepping

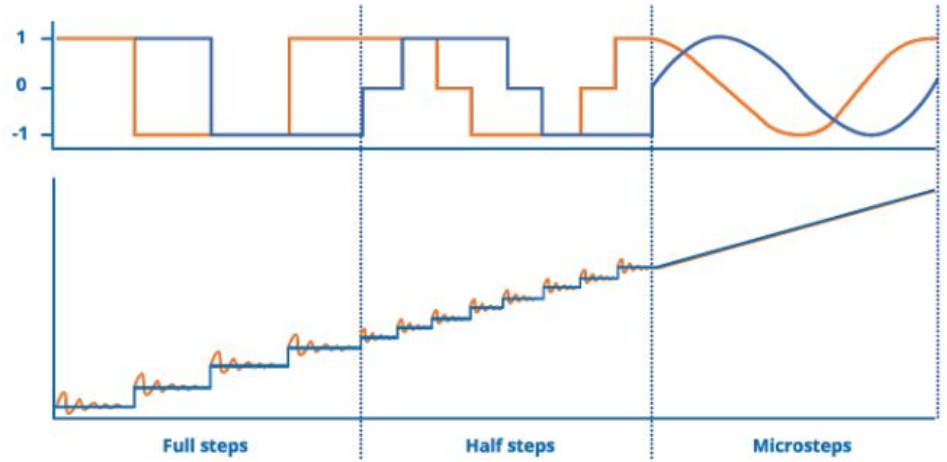


Figure 18: Demonstration of Sine signal following using different levels of micro-stepping

Integration – Optimizing for the A4988 Driver

Enabling Micro-stepping

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Figure 17: Micro-stepping programming table. The configuration is done by connecting the digital pins according to this table

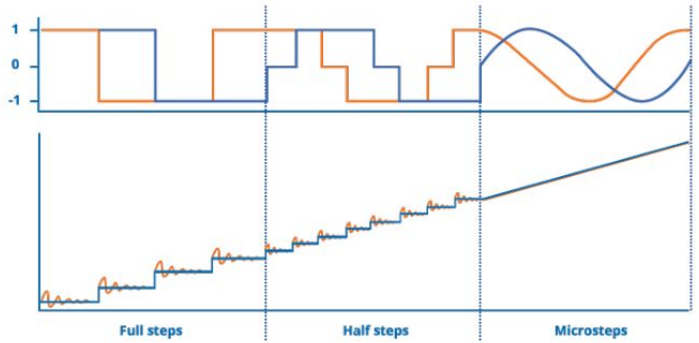


Figure 18: Demonstration of Sine signal following using different levels of micro-stepping

Integration – Optimizing for the A4988 Driver

Supplying Higher Voltage

- Problems with high frequency
  - The torque become too low → slipping
- The relation to micro-stepping
  - Microstepping → slower rotation → compensate with higher frequency

Integration – Optimizing for the A4988 Driver

Our current working point:

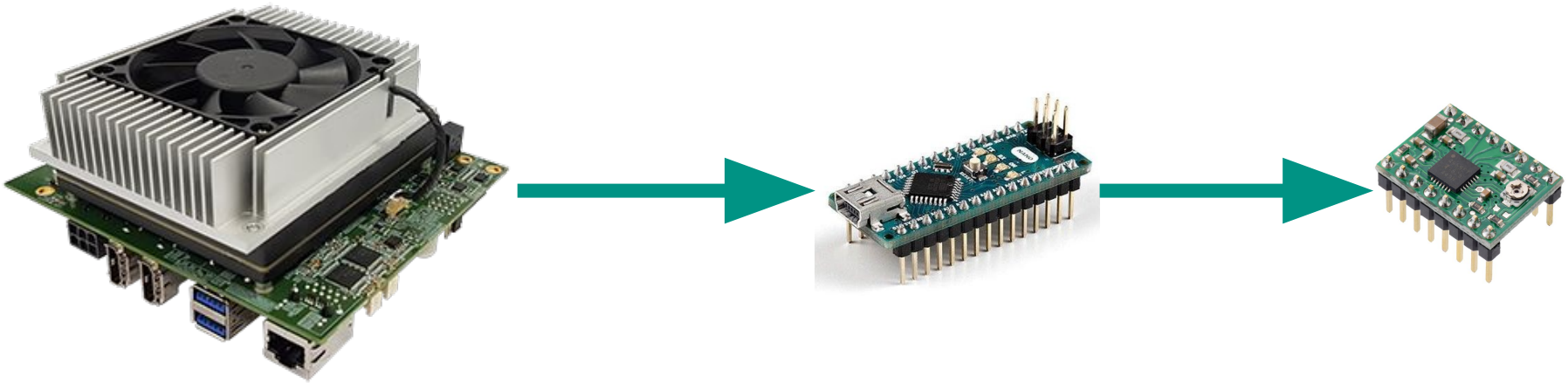
MS config.	Input Voltage $[V]$	Max. Freq. $[Hz]$	Calculated V_θ $[deg \cdot s^{-1}]$	1-Pulse size $[deg/pulse]$
LHL	19	3571	89.28	$\frac{1.8^\circ}{2}$

Table 4: Motors working point #2 specification

* LHL = quarter step

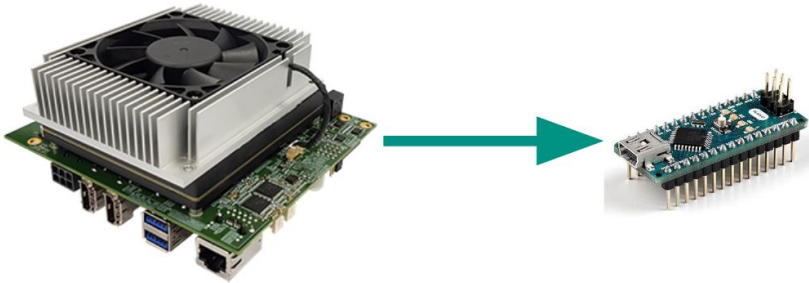
Integration - Arduino Code

Discussion: Controlling the A4988 Driver



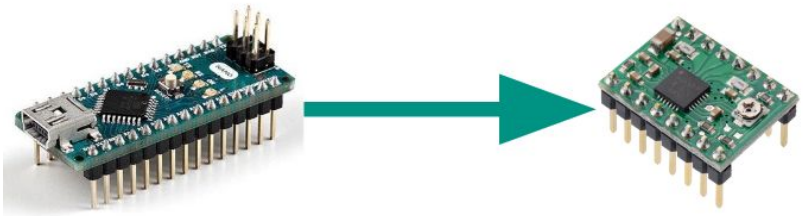
Integration - Arduino Code

- **Send Arduino commands**
- Moving the gimbal
- Implement Parallelism between 2-Axes



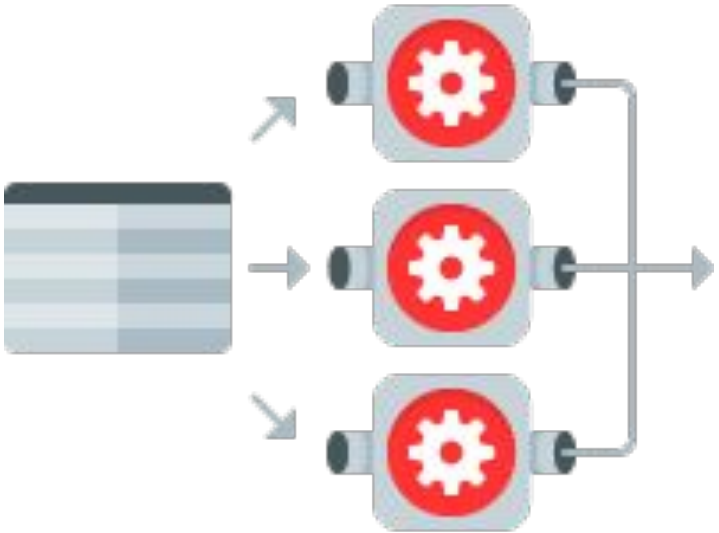
Integration - Arduino Code

- Send Arduino commands
- **Moving the gimbal**
- Implement Parallelism between 2-Axes



Integration - Arduino Code

- Send Arduino commands
- Moving the gimbal
- **Implement Parallelism between 2-Axes**



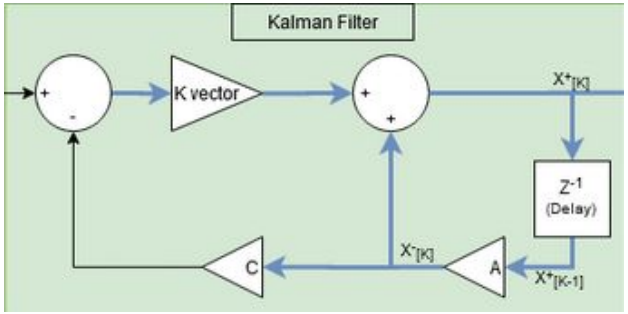
In-Depth Control Review

Control Review – System Modeling

- Model the system as a linear system
- Model the target motion as constant acceleration with noise
- Measurements are the target's relative position to the gimbal
- Use a Kalman filter to get information about targets:
  - Position
  - Speed
  - acceleration

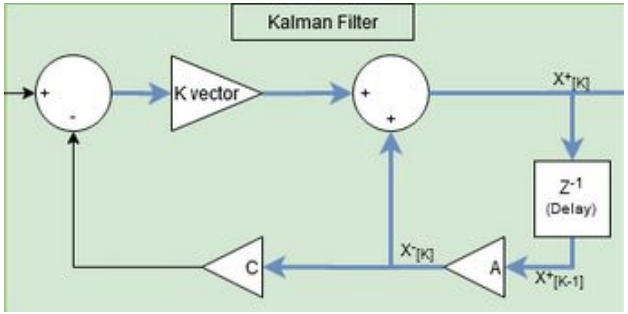
Control Review – Kalman Filter

- Implemented as a standard discrete Kalman filter
- Iterative process. For each processed image:
  - Calculate a K vector
  - If stationary, K vector is a constant
  - Predict state vector according to model
  - Correct state vector according to measurement and K vector



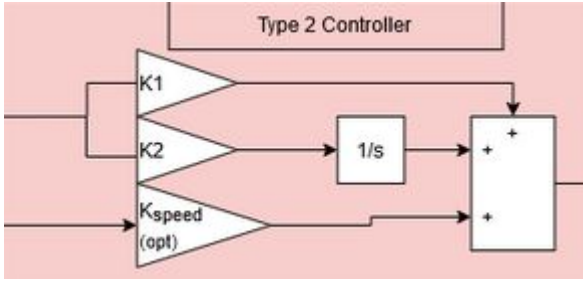
Control Review – Kalman Filter

- Code has options for stationary and non stationary filter, default is non stationary as:
  - Performance is not adversely affected by it
  - Better if time between each image processed varies
- However, results are usually the same as K vector eventually converges to the stationary value



Control Review – Controller

- Kalman filter outputs position, speed and acceleration
- Acceleration output is noisy and inaccurate so is not used
- Add discrete approximation of integral on position
- Result is a standard PID controller



Control Review – Camera Delay

- Delay from camera output to controller output is around 0.2-0.3 seconds.
- It is mostly due to limitations in camera hardware
- This creates problem with position calculation and tracking
- Best solution is to get a better camera
- Using the current gear, best solution is forward prediction

Control Review – Forward Prediction

- Add buffer to gimbal position to simulate delay
- Linear prediction according to:
  - Time delay value (Input to the system)
  - Average time between images (delta)
- Predict forward to correct amount of steps

$$\underrightarrow{x}_{pred} = \underrightarrow{x}_{delayed} A^{\left\lfloor \frac{t_{E2E}}{t_{\Delta}} \right\rfloor}$$

Control Review – Forward Prediction

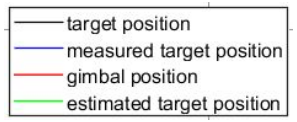
- The Good: Good tracking when target is in constant acceleration

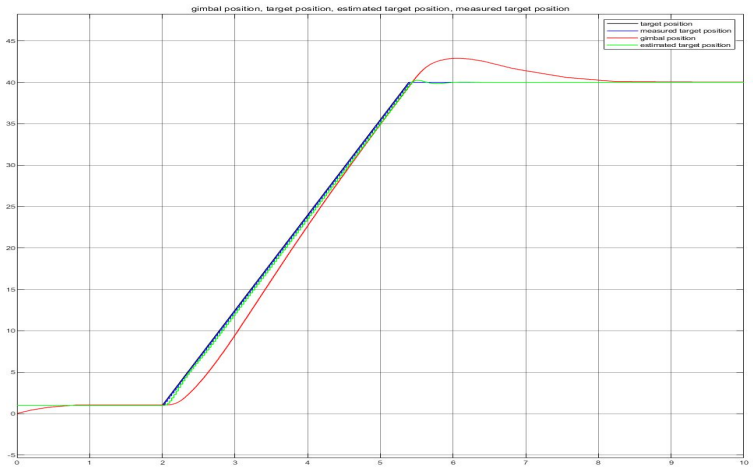
- The Bad: Performs poorly when target has sharp changes in acceleration:
  - Example: Gimbal overshoots when target abruptly changes direction

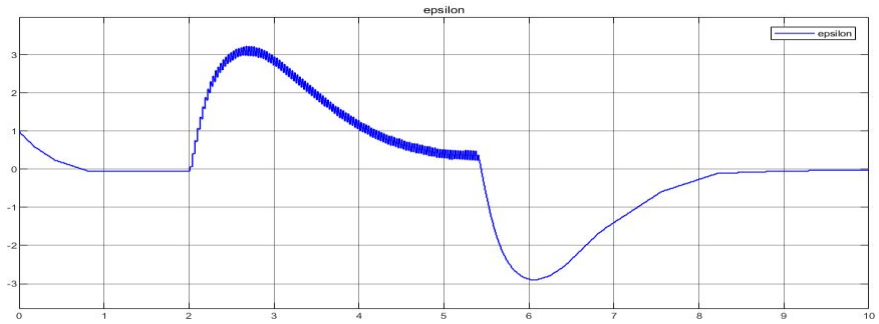
Performance

Performance – Simulation

Baseline simulation:

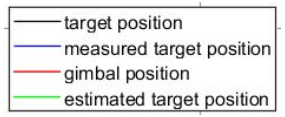


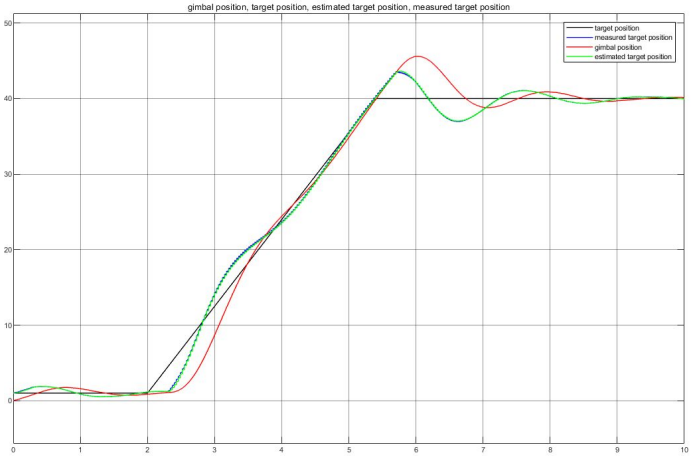


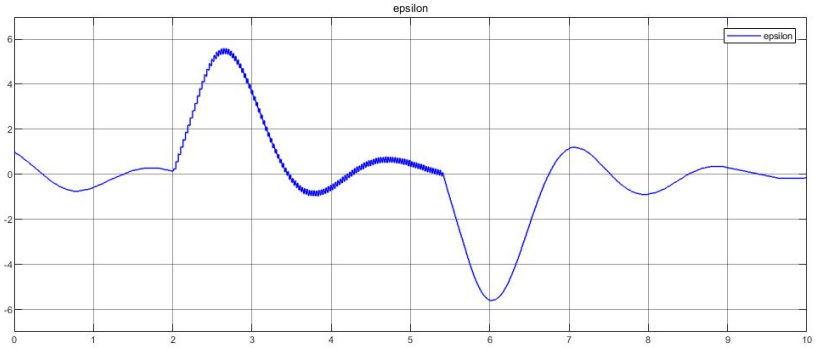


Performance – Simulation

The effects of E2E delay:

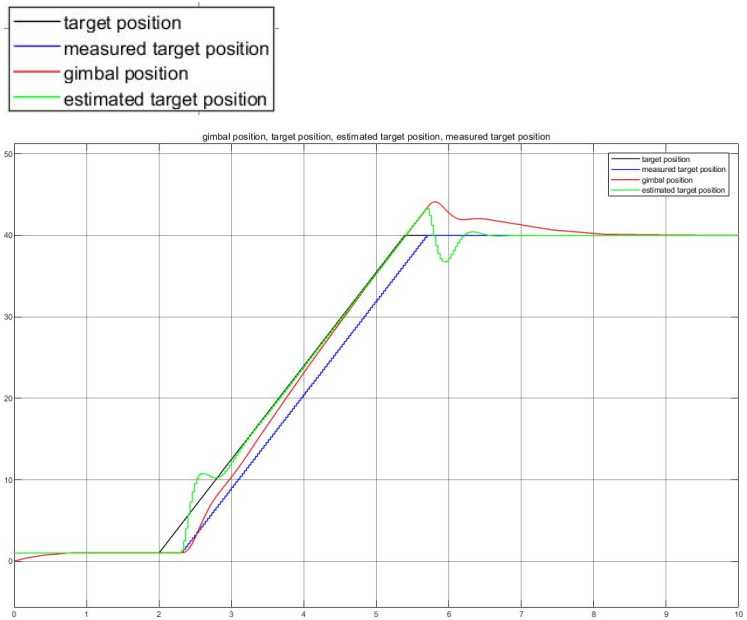


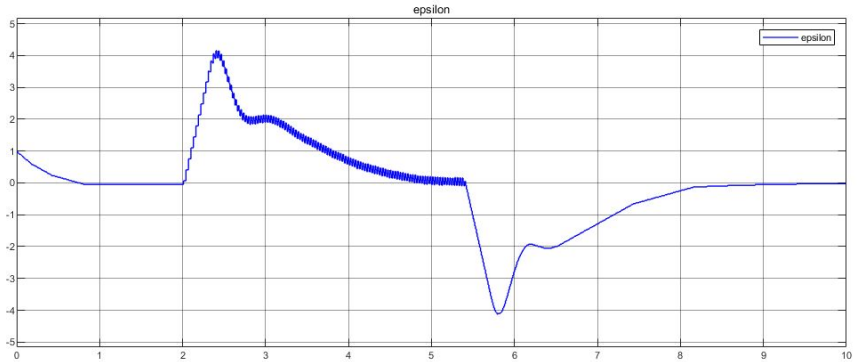




Performance – Simulation

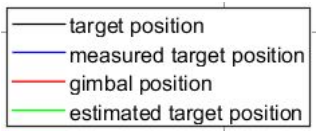
E2E delay with correction:

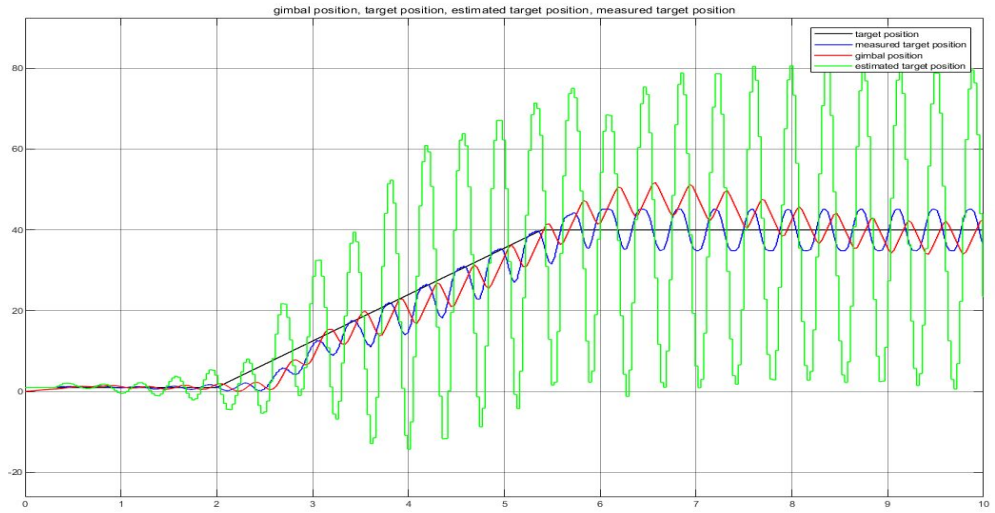




Performance – Simulation

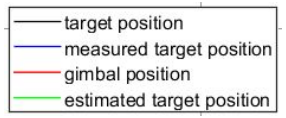
When predicting using the wrong E2E delay:

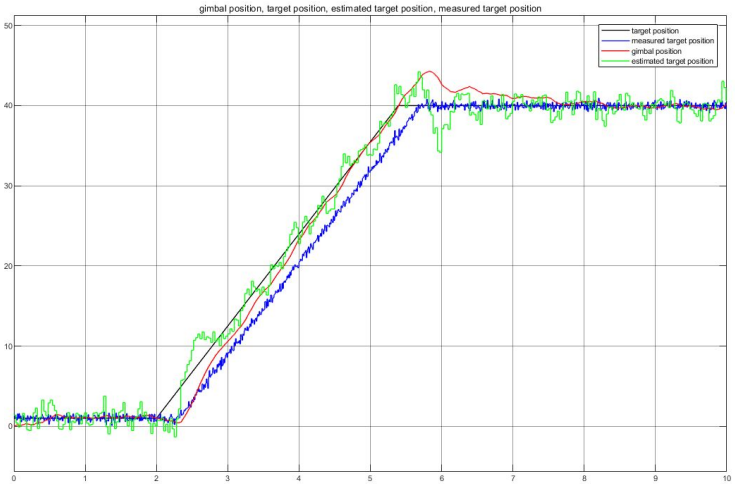


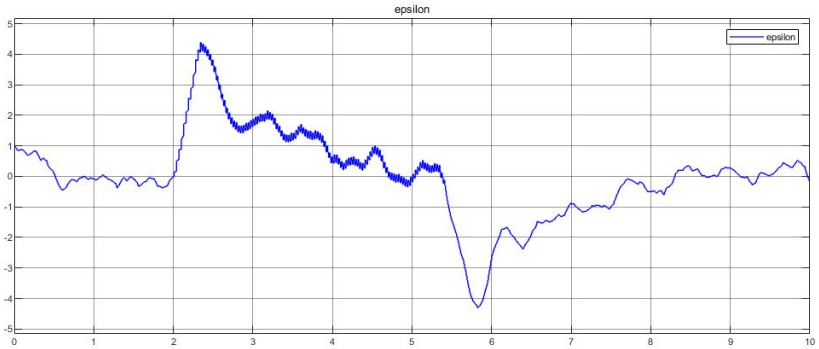


Performance – Simulation

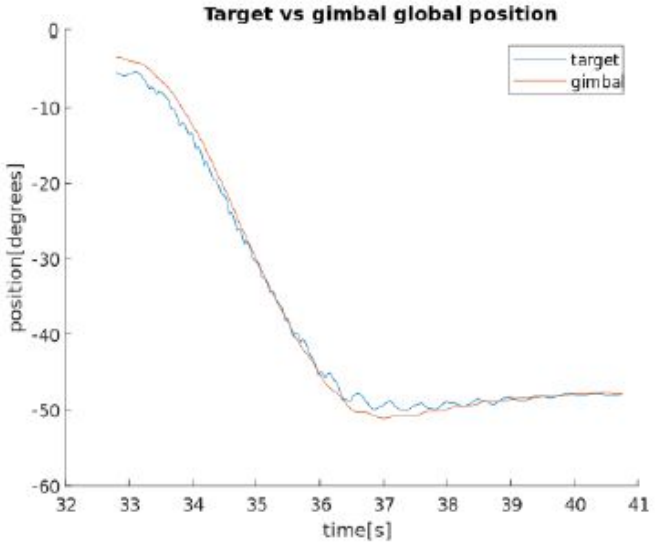
Add noise:

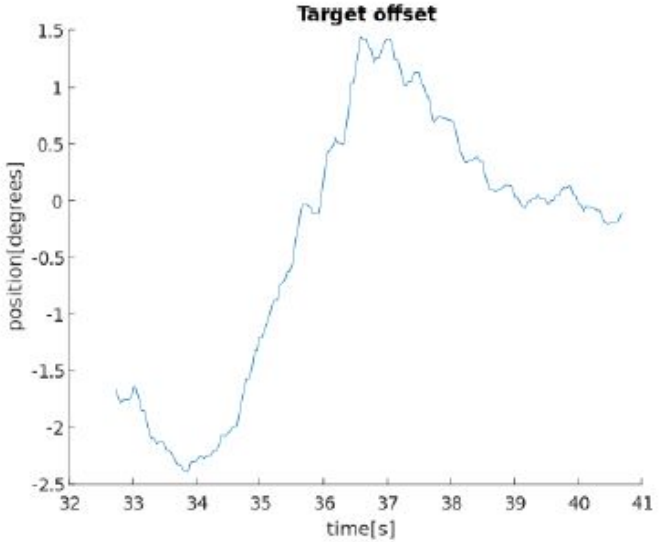




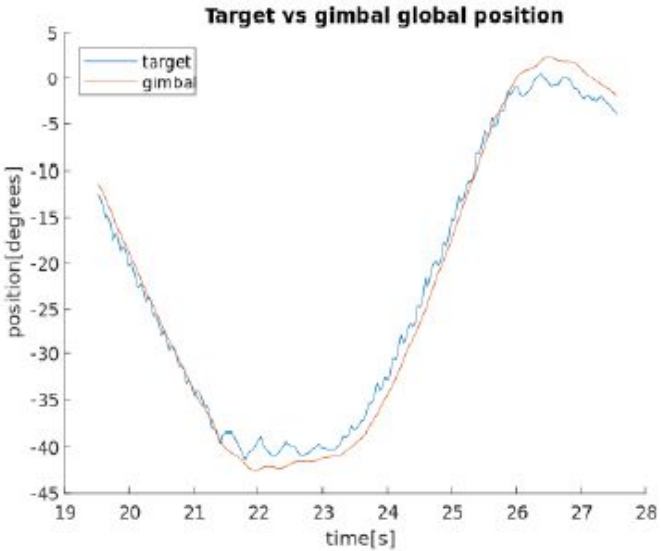


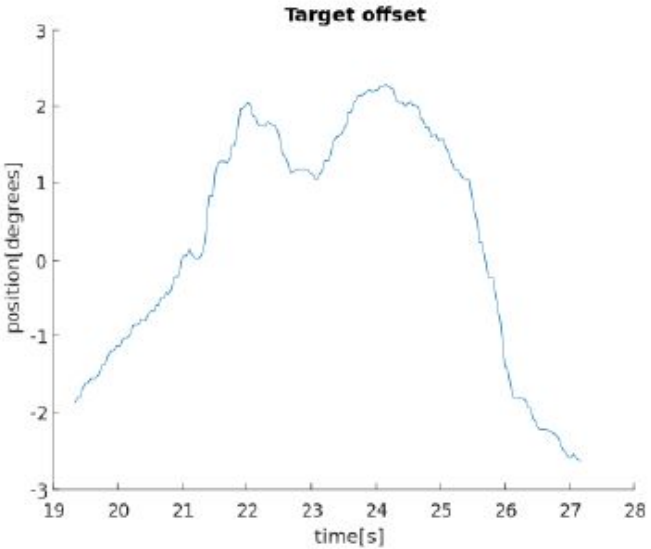
Performance - Real Life - PID = 4, 6, 0.1



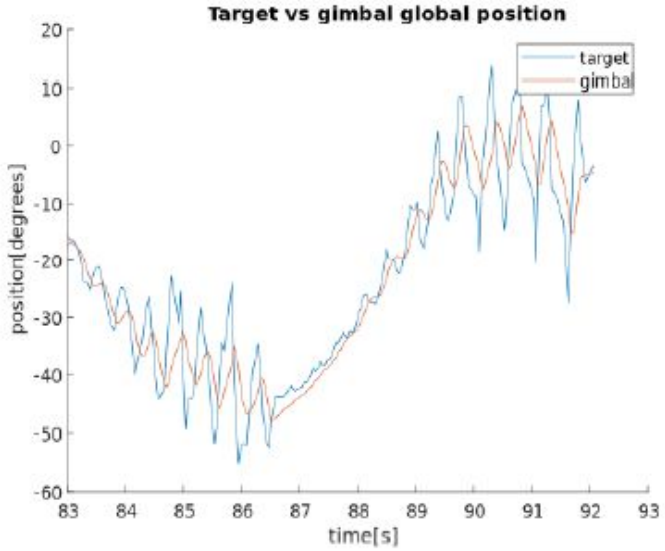


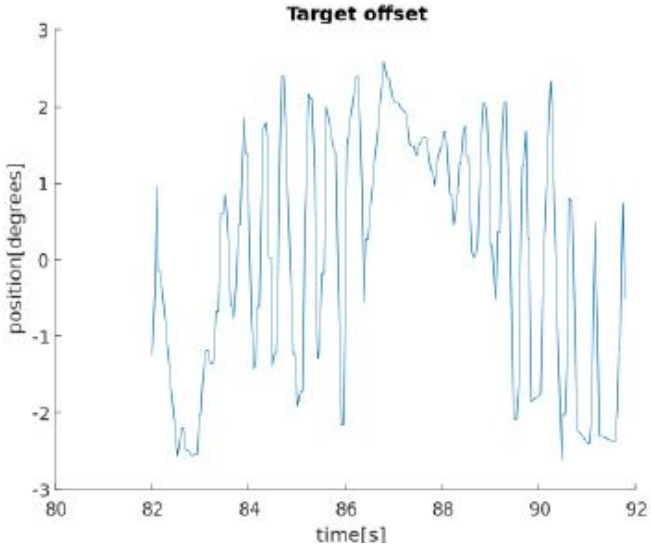
Performance - Real Life - PID = 4, 4, 0.1





Performance – Real Life – PID = 6, 6, 0.1





Summary & Conclusions

Summary & Conclusions

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

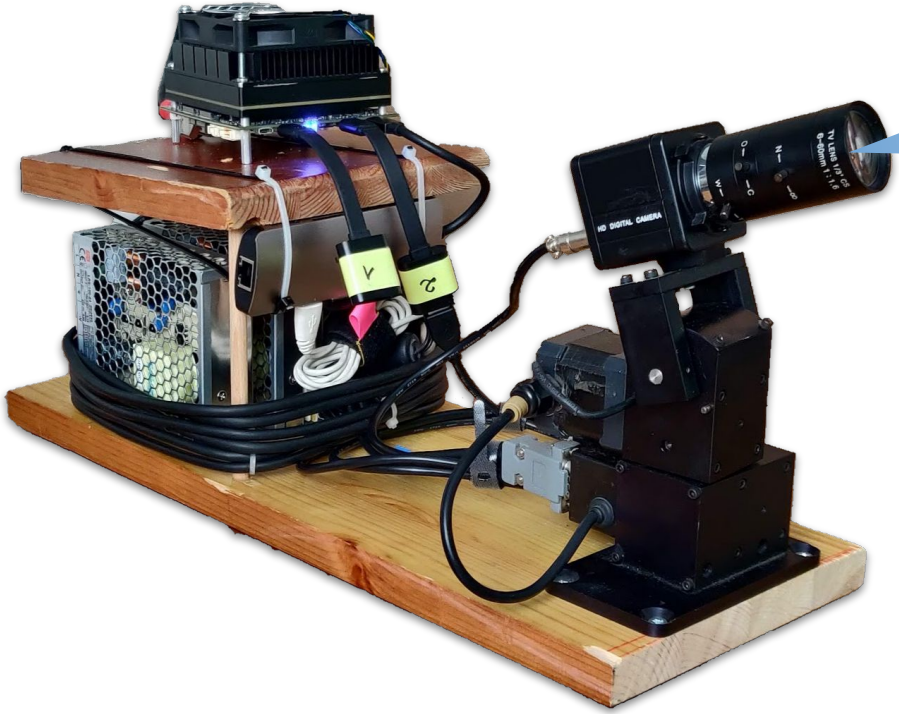
Questions?

Link to video

THE ANDREW & ERNA VITERBI
FACULTY OF
ELECTRICAL
ENGINEERING

TECHNION
Israel Institute of Technology

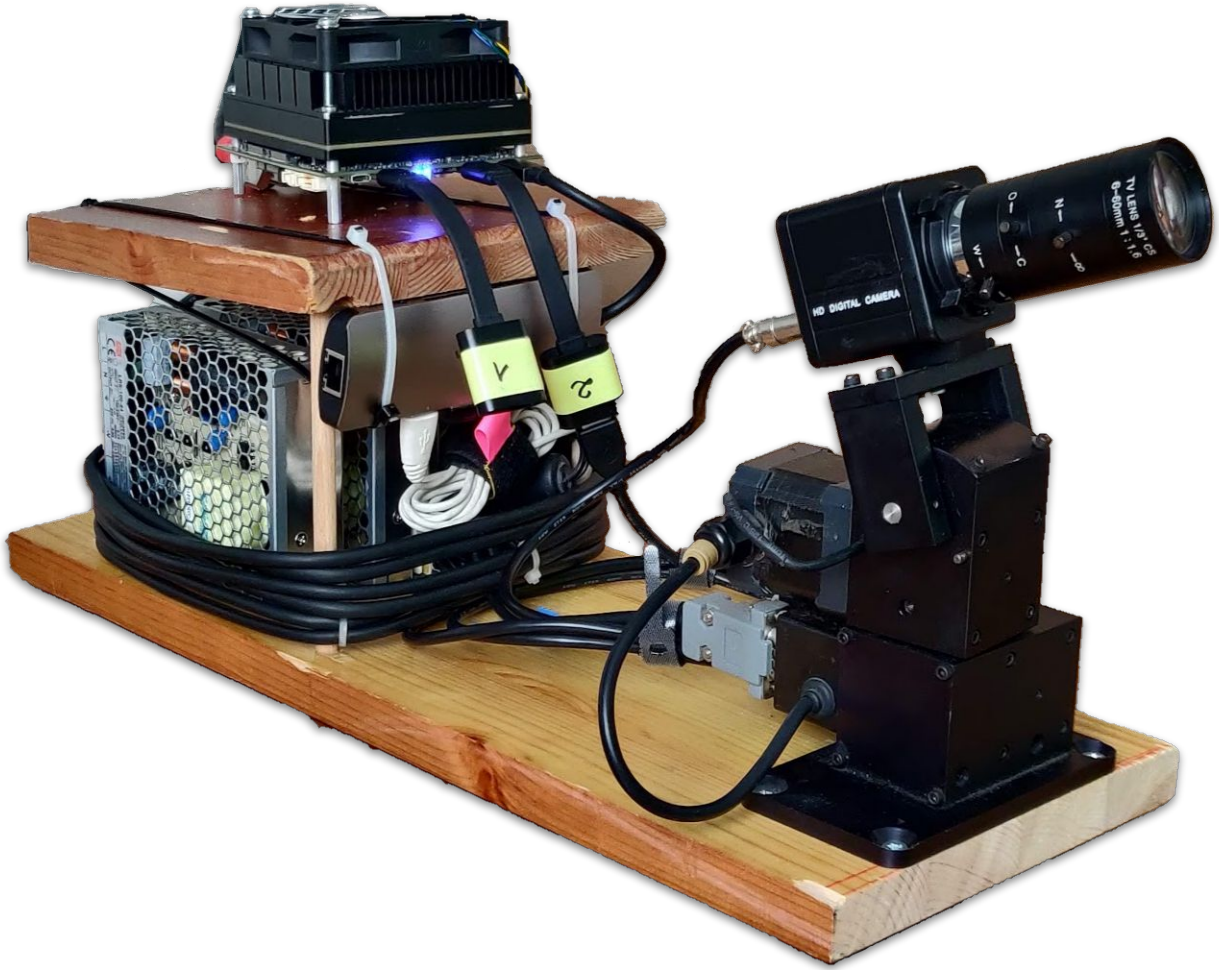
CRML
Control Robotics and Machine Learning Laboratory



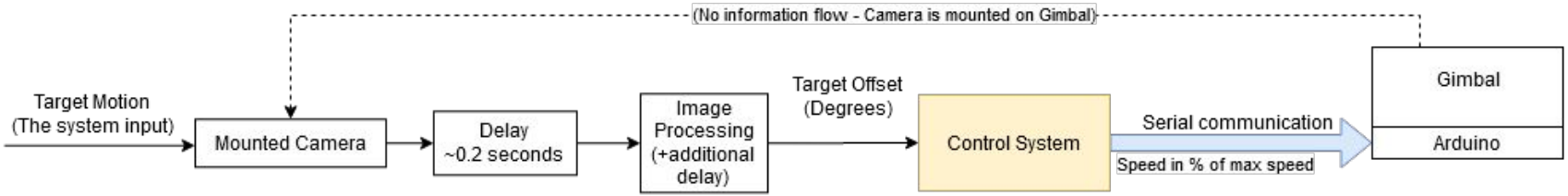
Amir Sarig
&
Idan Anner

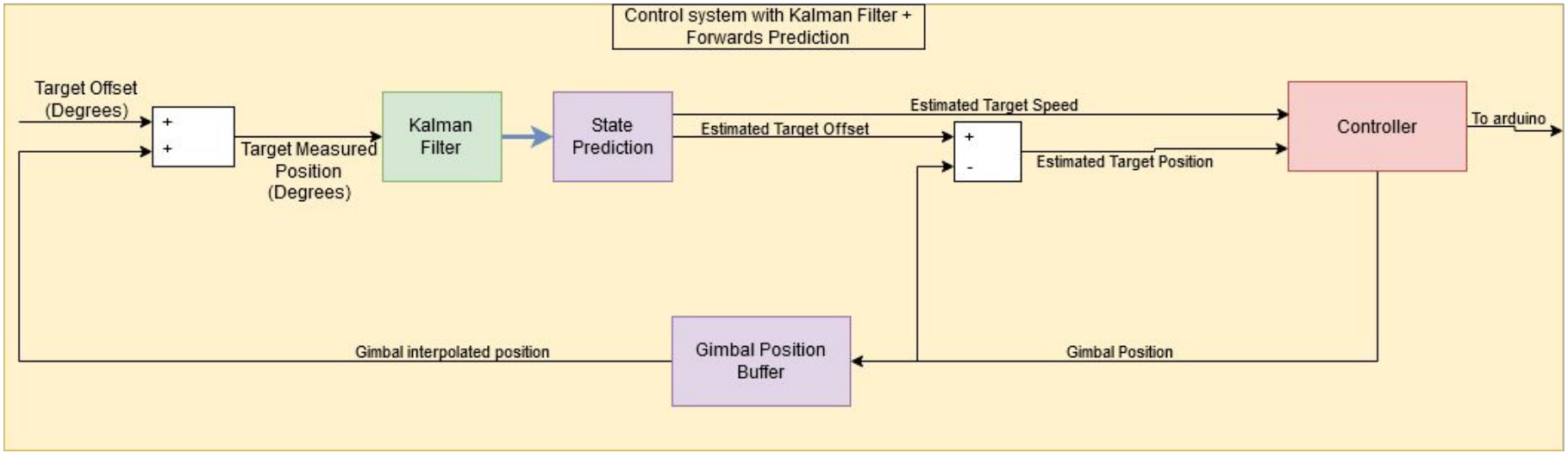
Supervised by Ilan Rusnak

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


HD DIGITAL CAMERA
TV LENS 1/3" CS
6~60mm 1:1.6
N
O
C
W
∞
A
2

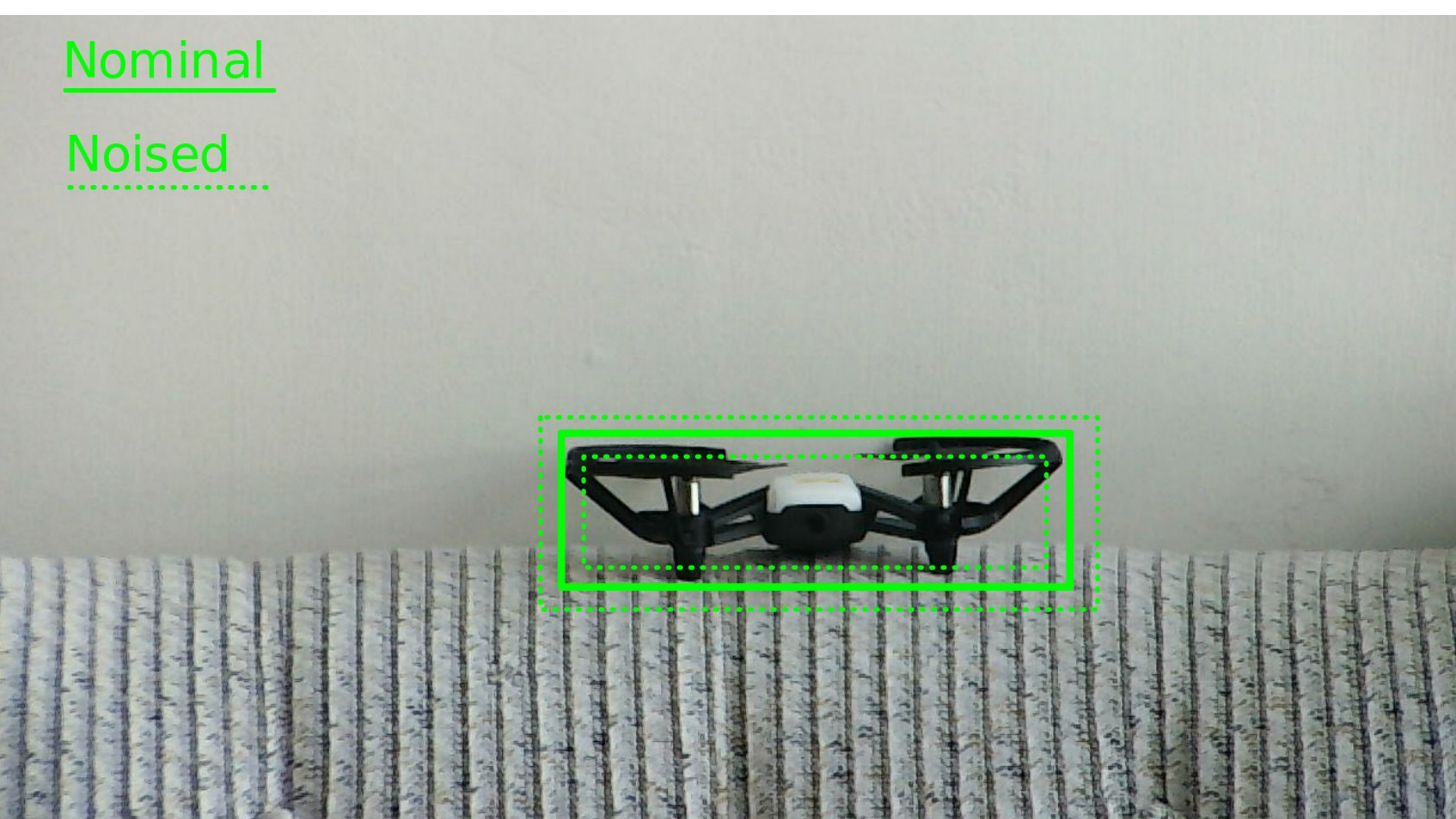
Modeling The System


(No information flow - Camera is mounted on Gimbal)
Target Motion
(The system input)
Mounted Camera
Delay
~0.2 seconds
Image
Processing
(+additional
delay)
Target Offset
(Degrees)
Control System
Serial communication
Speed in % of max speed
Gimbal
Arduino

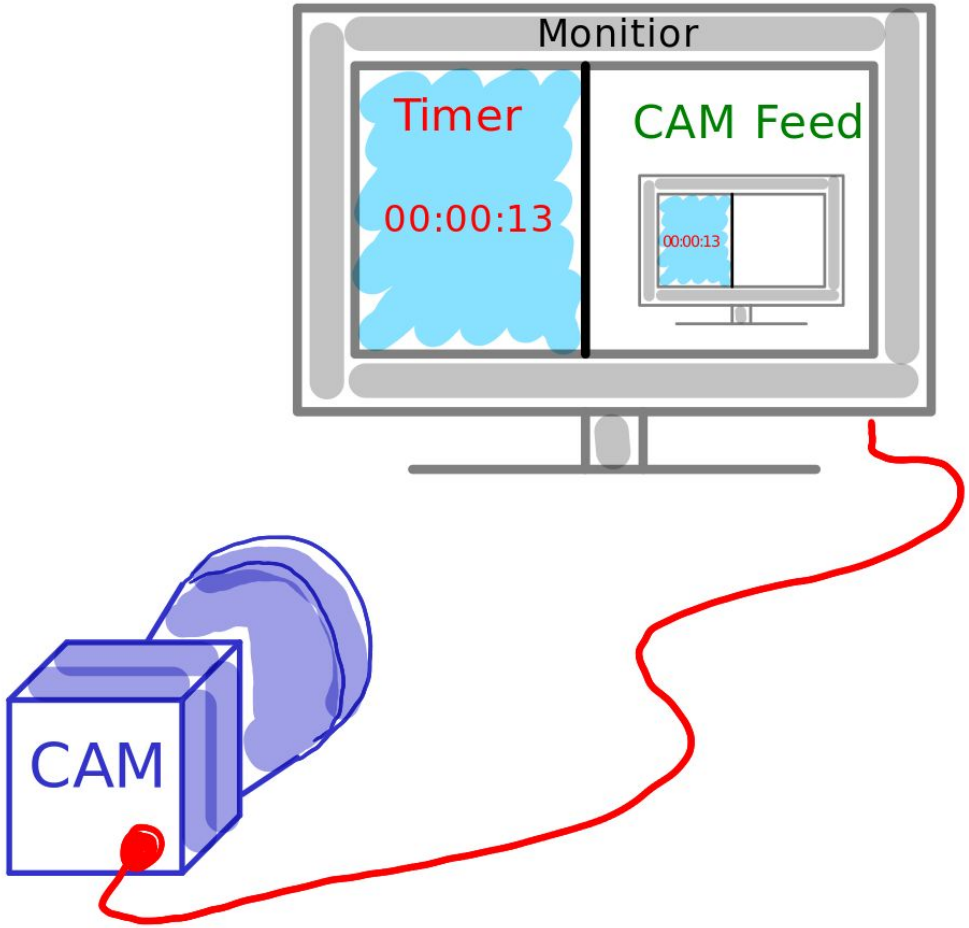

Control system with Kalman Filter +
Forwards Prediction
Target Offset
(Degrees)
+
+
Target Measured
Position
(Degrees)
Kalman
Filter
State
Prediction
Estimated Target Speed
Estimated Target Offset
+
-
Estimated Target Position
Controller
To arduino
Gimbal interpolated position
Gimbal Position
Buffer
Gimbal Position

Simulate and Tune

Drone Coordinates as Input


Nominal
Noised

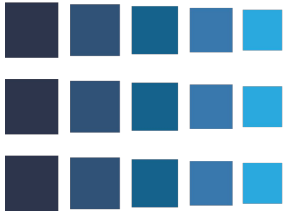
Camera’s Delay Characterized


Monitior
Timer
00:00:13
CAM Feed
00:00:13
CAM

Add Prediction

$$\underrightarrow{x}_{pred} = \underrightarrow{x}_{delayed} A^{\left\lfloor \frac{t_{E2E}}{t_{\Delta}} \right\rfloor}$$

Achieve Great Tracking



THE ANDREW & ERNA VITERBI
FACULTY OF ELECTRICAL ENGINEERING

TECHNION
Israel Institute of Technology

Control Robotics and Machine Learning Laboratory

October 2020