Carnegie Mellon University

Developing and Testing Custom Controllers for Crazyflie Drones for Wind Disturbances

Team 4 - WindBreakers

Denis Alpay
Nikolaj Hindsbo
Will Kraus
Kaustabh Paul

Agenda

- Motivation
- Dynamics Derivation
- Simulation Implementation
  - Introducing Gazebo simulation
- Hardware Implementation
  - How we made our own controllers
  - PID, LQR, SMC Results
- Conclusion & Closing Thoughts
- Q&A

Motivations: Why Drones? Why Wind?

- Goal: Balance performance with robustness to wind

If a drone can be robust, what can it do?

- Surveying
- Payload delivery
- Disaster relief (hurricanes, earthquakes, etc.)

Existing approaches for Robustness with Wind

Existing Solutions

- LQR (Princeton, Crazyflie Thesis Paper)
- PID (In Crazyflie Development)

Sliding Mode Control

- **Pros:**
  - Works online in real time
  - Very robust to model inaccuracies and measurement noise
  - Lyapunov stability provable
  - Convergence in finite time also provable
- **Cons:**
  - Chattering

Other approaches:

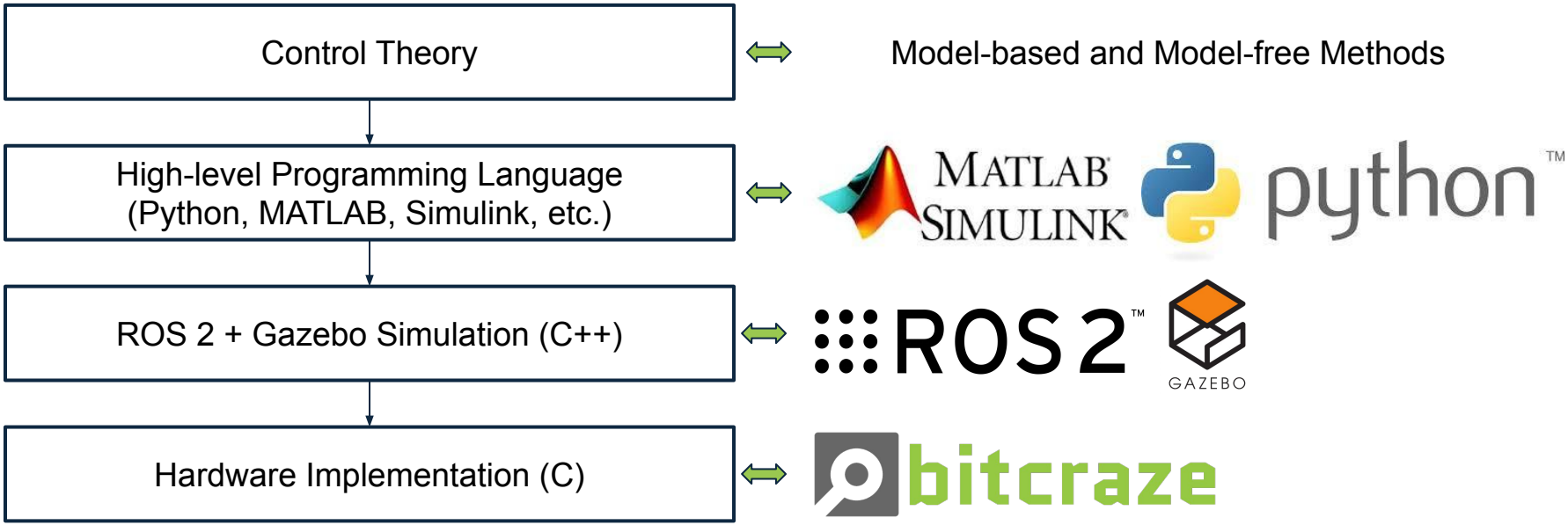
MPC: ⇒ Depends on offboard computation and communication speed

Backstepping: ⇒Recursive and less robust, more adaptive and stable

L1 adaptive controller: ⇒ Different robust control approach

G. Perozzi, D. Efimov, J.-M. Biannic, L. Planckaert, and P. Coton, "Wind rejection via quasi-continuous sliding mode technique to control safely a mini drone," in *7th European Conference for Aeronautics and Space Sciences (EUCASS)*, Milan, Italy, Jul. 2017.

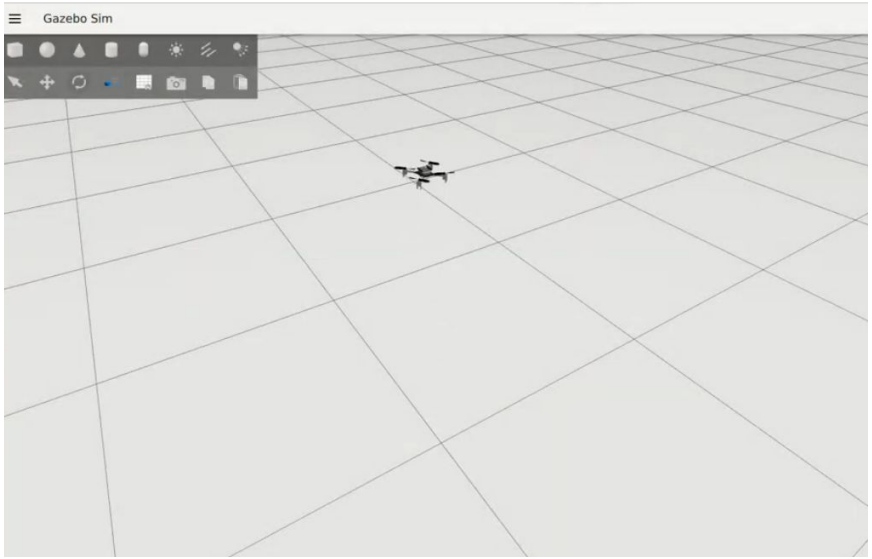
Controller Development Pipeline



Simulation Modeling Overview

Why Gazebo?

- Testing controllers in C++ (vs Python)
  - Makes it easier to convert controller logic into C for hardware implementation
- Open source implementations
- Wind simulation potential
- Crazyflie2.0 modeled by Bitcraze



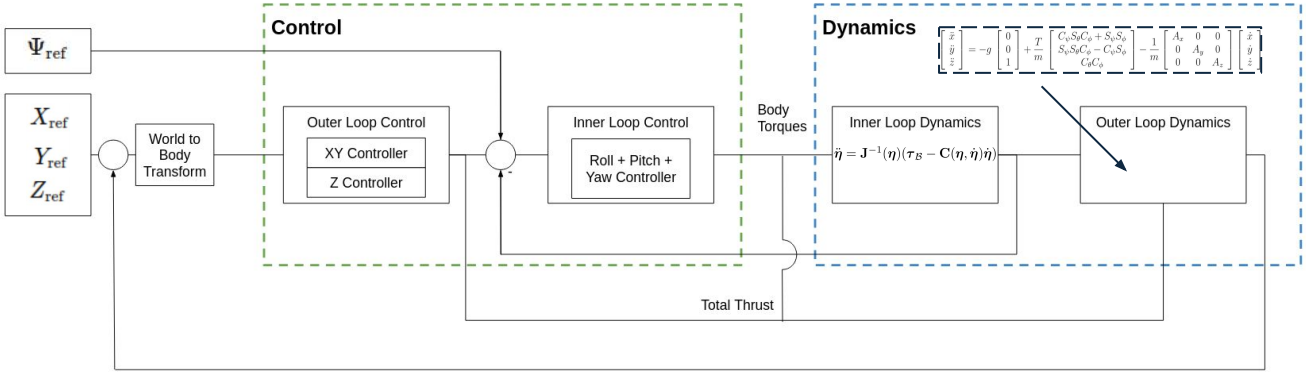
Cascaded vs Model-based Dynamics

Full state dynamics: Derived system dynamics form control output (LQR)

Cascaded: Several controllers fed into each other w/ dynamics (PID, SMC)

XYZ controllers feed into attitude controller → control output

Cascaded:

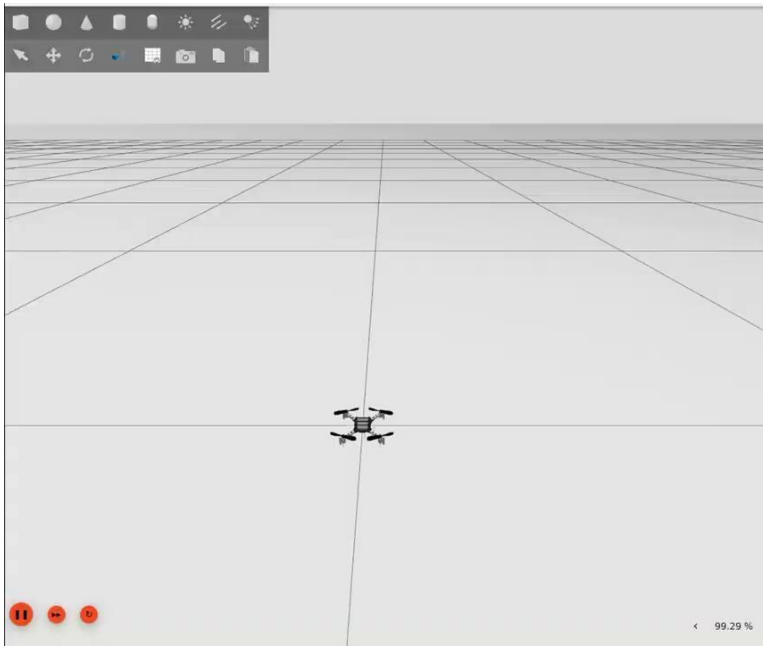


Model–based (State Space):

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Simulation Modeling PID

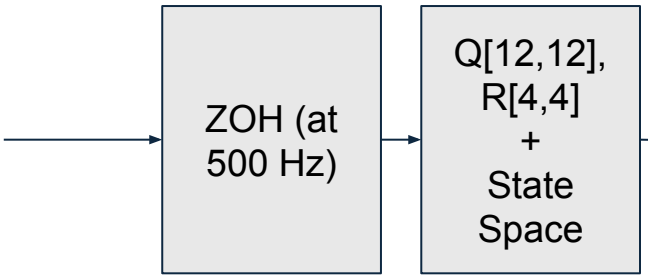


- **Validated Controller Feasibility:** The simulation confirmed that our controller was feasible and ready for hardware implementation.
- **Optimized Tuning Process:** Provided initial PID values as a strong foundation for further hardware tuning.
- **Cost Benefit:** Able to test and develop controllers without risking breaking the hardware.

IH-LQR Model Dynamics

Linearized Model → Discretization → Q,R → K_inf matrix using DT Algebraic Riccati

$$x' = x'$$
$$y' = y'$$
$$z' = z'$$
$$x'' = -g\theta$$
$$y'' = g\phi$$
$$z'' = -\frac{U_1}{m}$$
$$\phi' = \phi'$$
$$\theta' = \theta'$$
$$\psi' = \psi'$$
$$\phi'' = \frac{U_2}{I_x}$$
$$\theta'' = \frac{U_3}{I_y}$$
$$\psi'' = \frac{U_4}{I_z}$$

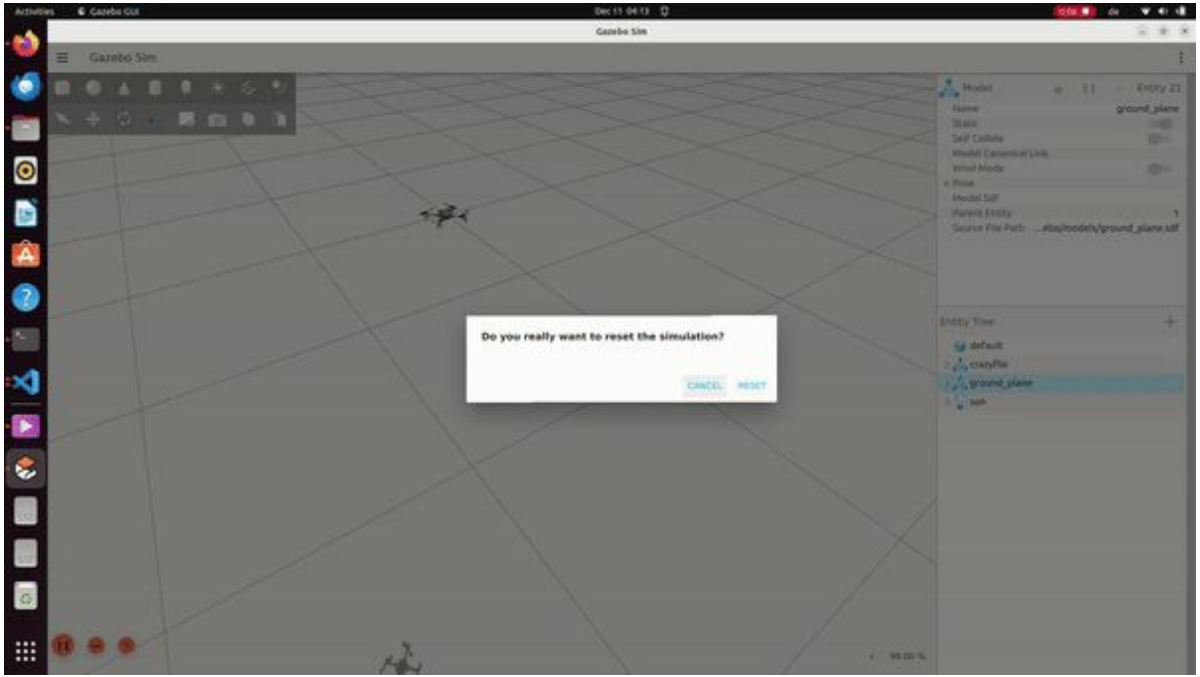


$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k \right)$$

$$P = A_d^T P A_d - A_d^T P B_d (B_d^T P B_d + R)^{-1} B_d^T P A_d + Q$$

$$K = (B_d^T P B_d + R)^{-1} B_d^T P A_d$$

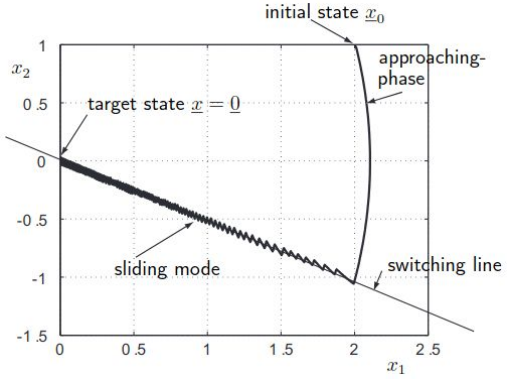
Simulation Modeling LQR



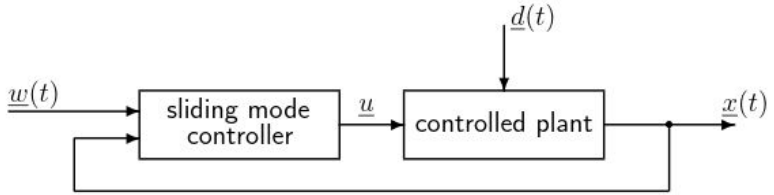
Sliding mode control design

- Given: Nonlinear sys $\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}) + \underline{d}(\underline{x}, t)$

$$\underline{y} = \underline{h}(\underline{x}, \underline{u})$$



- Sliding manifold S(x)



- Maintains system states on a *sliding*
- Uses discontinuous control to drive errors to zero.

SMC controller basic derivation

Lyapunov fcn: $V(s) = \frac{1}{2}s^2$

Positive Definiteness:

$$V(s) > 0 \quad \text{for } s \neq 0, \quad V(s) = 0 \text{ at } s = 0$$

Radial Unboundedness:

$$V(s) \to \infty \quad \text{as } |s| \to \infty$$

Negative Semi-Definiteness of Time Derivative:

$$\dot{V}(s) = s\dot{s} \leq 0$$

SMC controller basic derivation

Lyapunov fcn: $V(s) = \frac{1}{2}s^2$

Sliding manifold: $s(x) = \dot{x} + \alpha x$

- α acts as a tuning parameter that controls the rate of convergence.
- A larger α results in faster convergence to the equilibrium, but it may increase control effort and sensitivity to disturbances.
- A smaller α slows down convergence, which could make the system less responsive but reduces chattering and control effort.

SMC controller basic derivation

Lyapunov fcn: $V(s) = \frac{1}{2}s^2$

Sliding manifold: $s(x) = \dot{x} + \alpha x$

System dynamics $\ddot{x} = f(x, \dot{x}) + bu$

$\rightarrow \quad \dot{s}(x) = f(x, \dot{x}) + bu + \alpha \dot{x}$

Since $\dot{V}(s) = s\dot{s} \leq 0 \quad \rightarrow \quad sf(x, \dot{x}) + bsu + \alpha s\dot{x} \leq 0$

Equivalent Control (u_{eq}): Cancels the nominal system dynamics.

Switching Control (u_{sw}): Drives the system toward the sliding manifold and compensates for uncertainties/disturbances.

SMC controller basic derivation

Requirement: $sf(x,\dot{x}) + bsu + \alpha s\dot{x} \leq 0$

Decompose control law into 2 parts: $u = u_{\text{dyn}} + u_{\text{sm}}$

- Dynamics Compensation: Cancels out system dynamics

$$u_{\text{dyn}} = -\frac{1}{b}\left(f(x,\dot{x}) + \alpha\dot{x}\right)$$

- Sliding mode term: Ensures convergence to sliding manifold

$$u_{\text{sw}} = -k\,\text{sign}(s)$$

SMC controller basic derivation

$$u_{\mathrm{sw}} = -k \cdot \mathrm{sign}(s)$$

→Prone to high frequency chattering

$$u_{\mathrm{sw}} = -f(x) \cdot \mathrm{sign}(s)$$

→Chattering reduction through state dependent gain

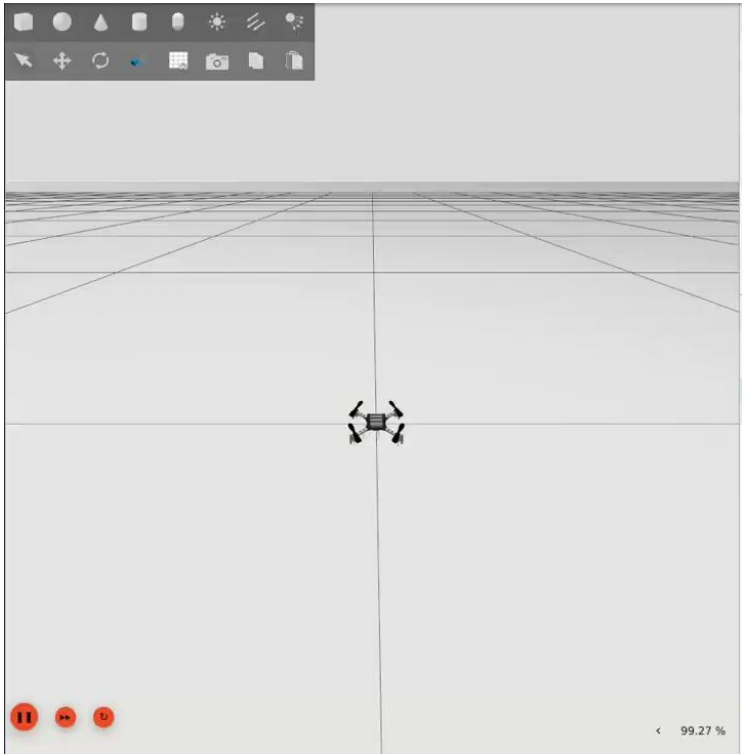
→Chattering still exists due to discontinuity

$$u_{\mathrm{sm}} = -k \cdot s$$

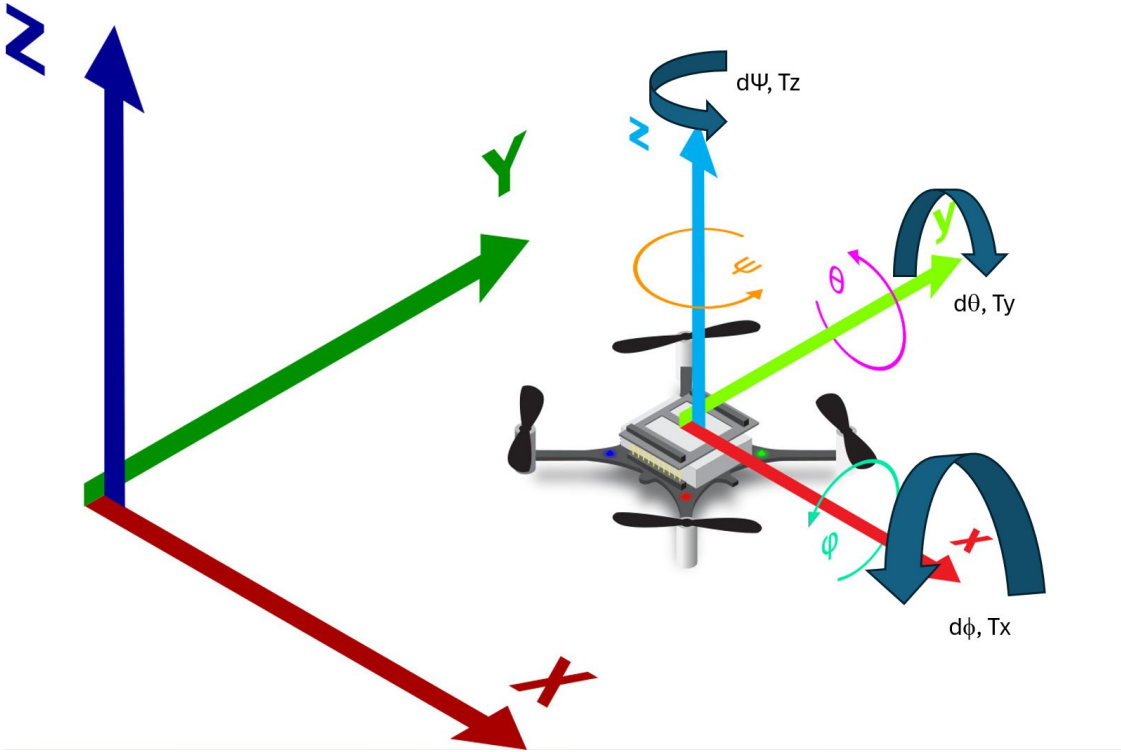
→Smooth but slow convergence

→Our SMC is combining second and third formula

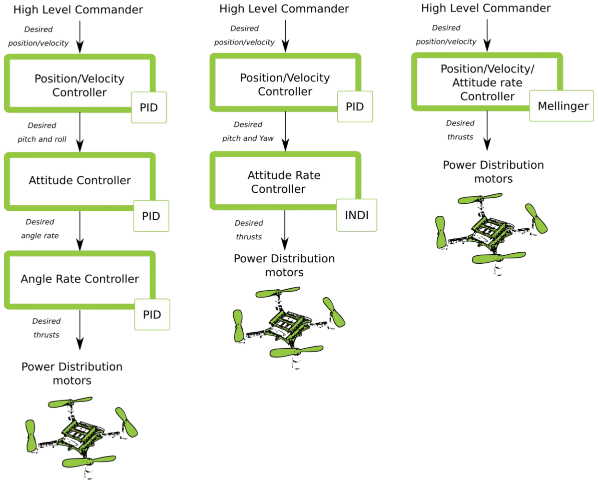
Simulation Modeling SMC

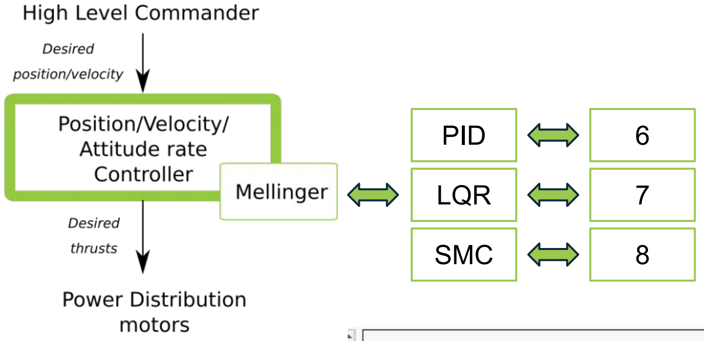


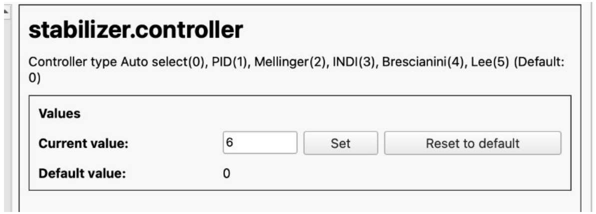
Hardware Implementation Overview



Hardware Implementation Overview





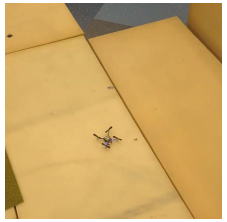


Hardware Implementation - PID

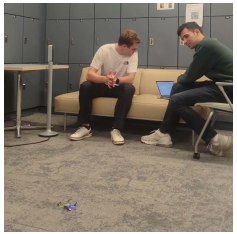
Name	Type	Access	Persistent	Value
ctrlLee				
ctrlMel				
ctrlOurPID				
a_baseline	float	RW	Yes	0.33
imax_p	float	RW	Yes	0.0008
imax_r	float	RW	Yes	0.0008
imax_x	float	RW	Yes	0
imax_y	float	RW	Yes	0
imax_y_t	float	RW	Yes	0
imax_z	float	RW	Yes	0
kd_p	float	RW	Yes	0.0008
kd_r	float	RW	Yes	0.0008
kd_x	float	RW	Yes	0
kd_y	float	RW	Yes	3e-05
kd_y_t	float	RW	Yes	0
kd_z	float	RW	Yes	0.05
ki_p	float	RW	Yes	0.001
ki_r	float	RW	Yes	0.001
ki_x	float	RW	Yes	0
ki_y	float	RW	Yes	0
ki_y_t	float	RW	Yes	0
ki_z	float	RW	Yes	0
kp_p	float	RW	Yes	0.003
kp_r	float	RW	Yes	0.003
kp_x	float	RW	Yes	0.3
kp_y	float	RW	Yes	0.001
kp_y_t	float	RW	Yes	0.3
kp_z	float	RW	Yes	0.1
omax_p	float	RW	Yes	1
omax_r	float	RW	Yes	1
omax_x	float	RW	Yes	0.3
omax_y	float	RW	Yes	1
omax_y_t	float	RW	Yes	0.3

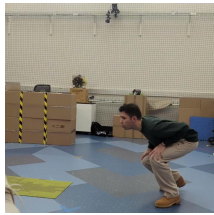
- Motivation:
  - Tunable Implementation
  - Would help us understand codestack
  - Baseline controller is easy to compare

Initial

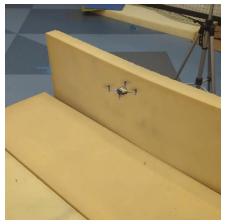


Progress

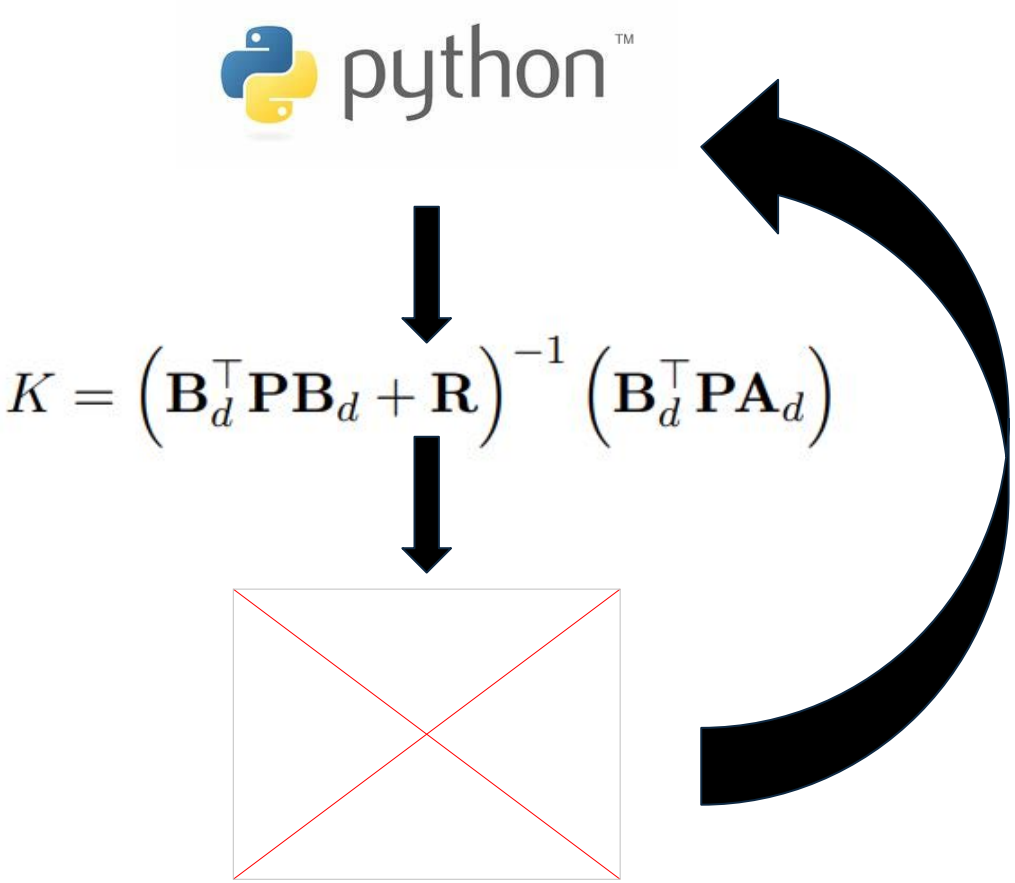




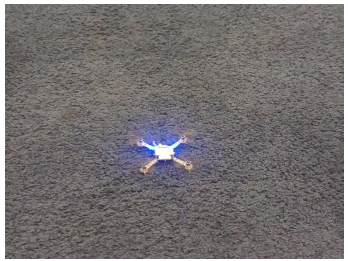
Tuned Results



Hardware Implementation - LQR



Initial



Progress

ITERATION 1 VIDEO

ITERATION 3 VIDEO

Hardware Implementation - SMC

ctrlMel				
ctrlOurPID				
ctrlSMC				
A	float	RW	Yes	0.35
I_xx	float	RW	Yes	1.6572e-05
I_yy	float	RW	Yes	1.6656e-05
I_zz	float	RW	Yes	2.9262e-05
alpha_pitch	float	RW	Yes	0.45
alpha_roll	float	RW	Yes	0.45
alpha_z	float	RW	Yes	25
g	float	RW	Yes	9.81
gain_pitch	float	RW	Yes	0.48
gain_roll	float	RW	Yes	0.48
gain_yaw	float	RW	Yes	1
gain_z	float	RW	Yes	1
kp_x	float	RW	Yes	0.3
kp_y	float	RW	Yes	0.3
m	float	RW	Yes	0.031
nu	float	RW	Yes	3.3
omax_x	float	RW	Yes	0.2
omax_y	float	RW	Yes	0.2
omin_x	float	RW	Yes	-0.2
omin_y	float	RW	Yes	-0.2
ctrlSMC5				
deck				
firmware				

- Key Features
  - Tunable Implementation
  - Chattering is evident when not tuned, confirming SMC implementation
  - Seems to resist large perturbations in roll & yaw, but position control needs tuning

Initial

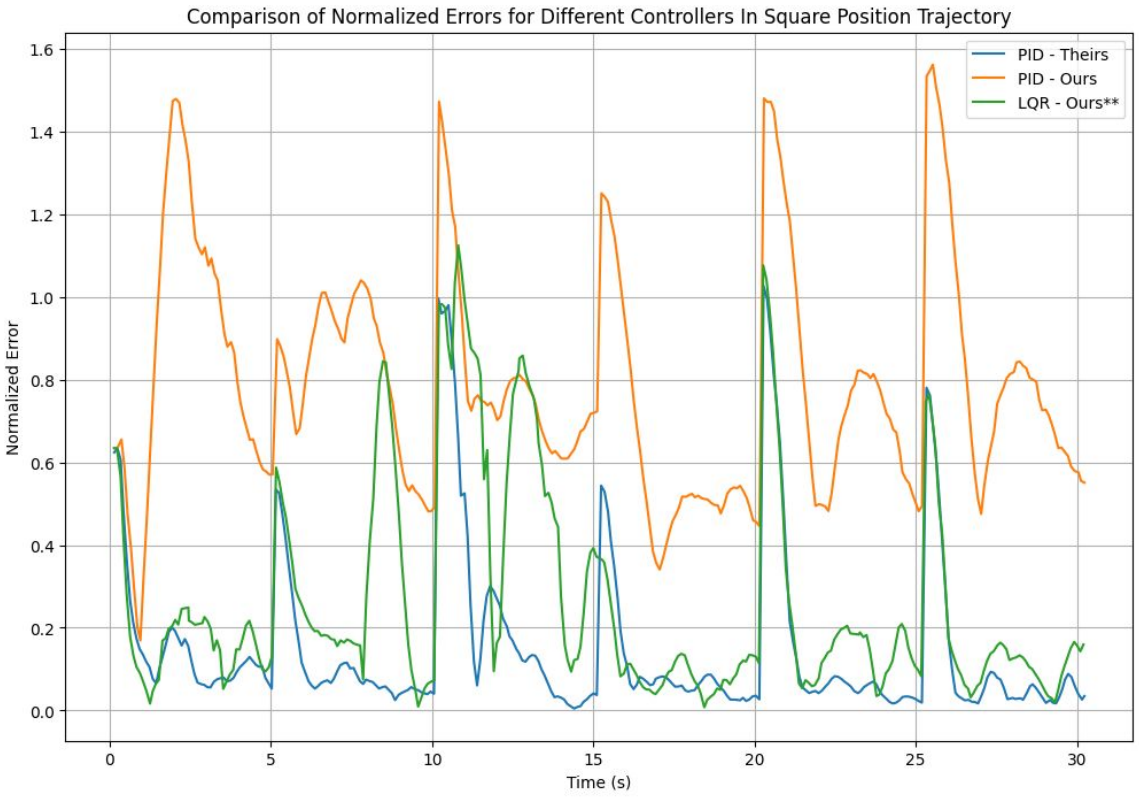
Progress

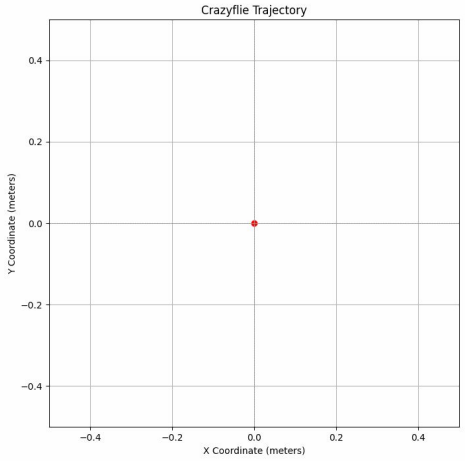
ITERATION 1 VIDEO

ITERATION 2 VIDEO

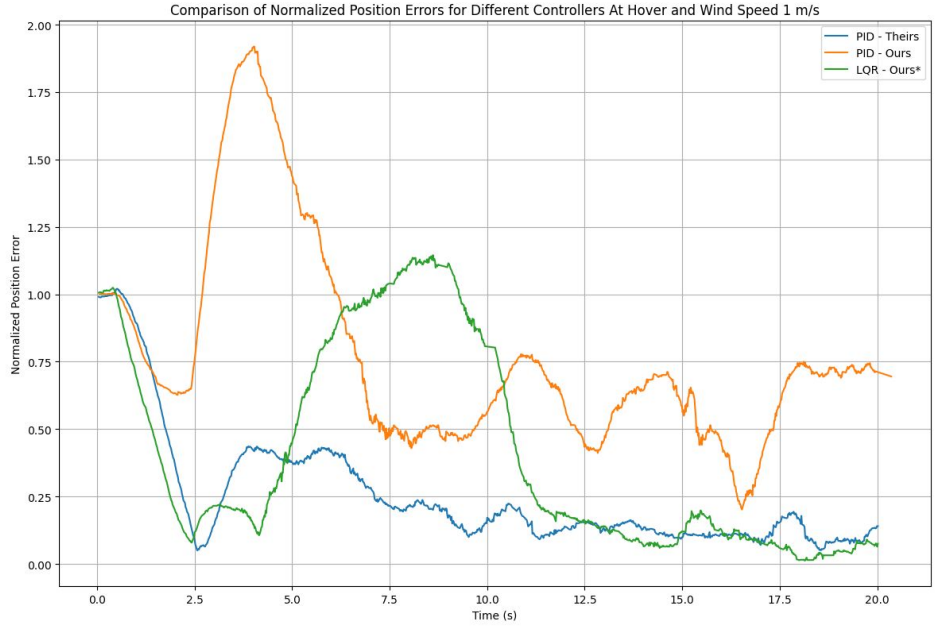
ITERATION 3 VIDEO

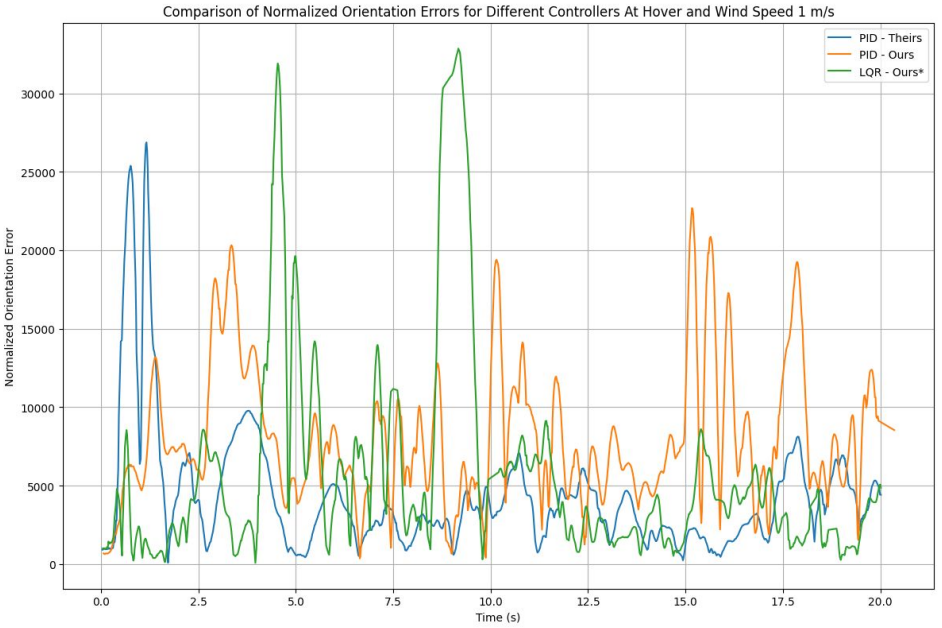
Hardware Implementation - Comparing Results





Hardware Implementation - Comparing Results





Concluding Thoughts

- Three controllers were implemented in Crazyflie
- Proved a Sim-to-Real pipeline that could be further expanded
  - With great tuning strategies for most controller types
- Hardware implementation is hard, but in the end, position control under low wind speed perturbations was achieved.