Autonomous Drone Racing with Vision-Aided Minimum Snap Trajectory Generation and MPC-Based Velocity Control

Team Name - mkg7,srm17, abhia2, ayusht3

Problem : Vision + State Based Control for Racing Drones

•Highly dynamic, 6-DOF nonlinear system for complex path following

•Prominence in applications like video-making and racing over the past decade

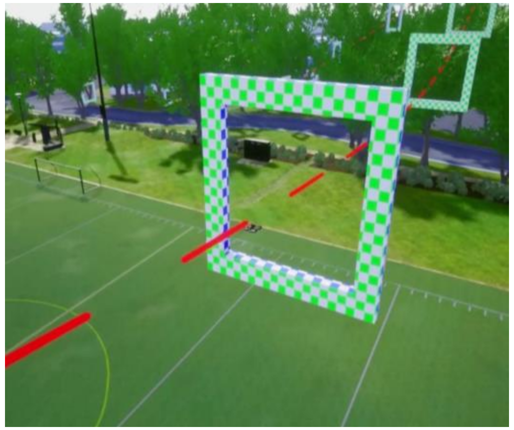
•Goal: Build an autonomous drone to navigate pre-defined gates

•Requirements: Avoid crashes, ensure safety, and prioritize speed for racing

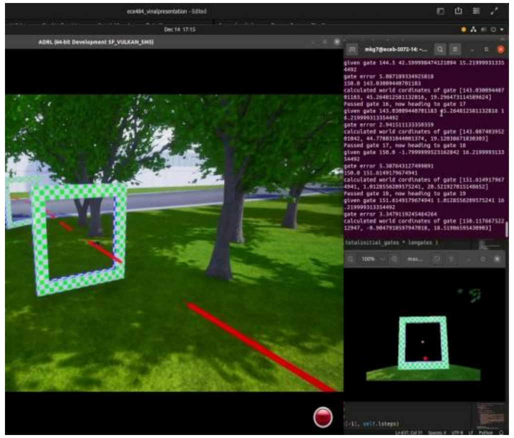
Solution Overview

- Reference Trajectory
- Vision-Guided Trajectory
- High Level Controller
- Low Level Controller

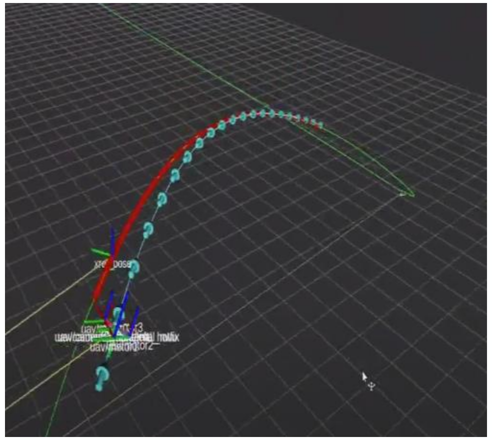
Spline-Interpolated Trajectory with Slerp Orientation Alignment



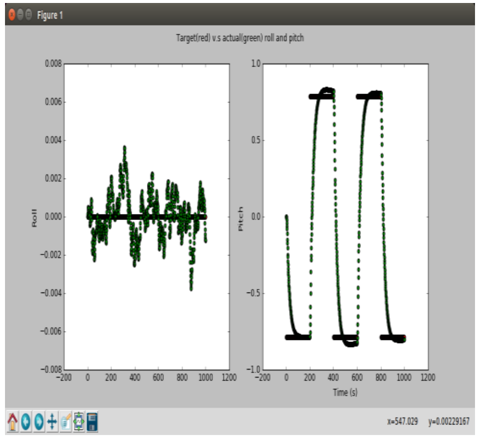
Real-Time Trajectory Adjustment using Vision

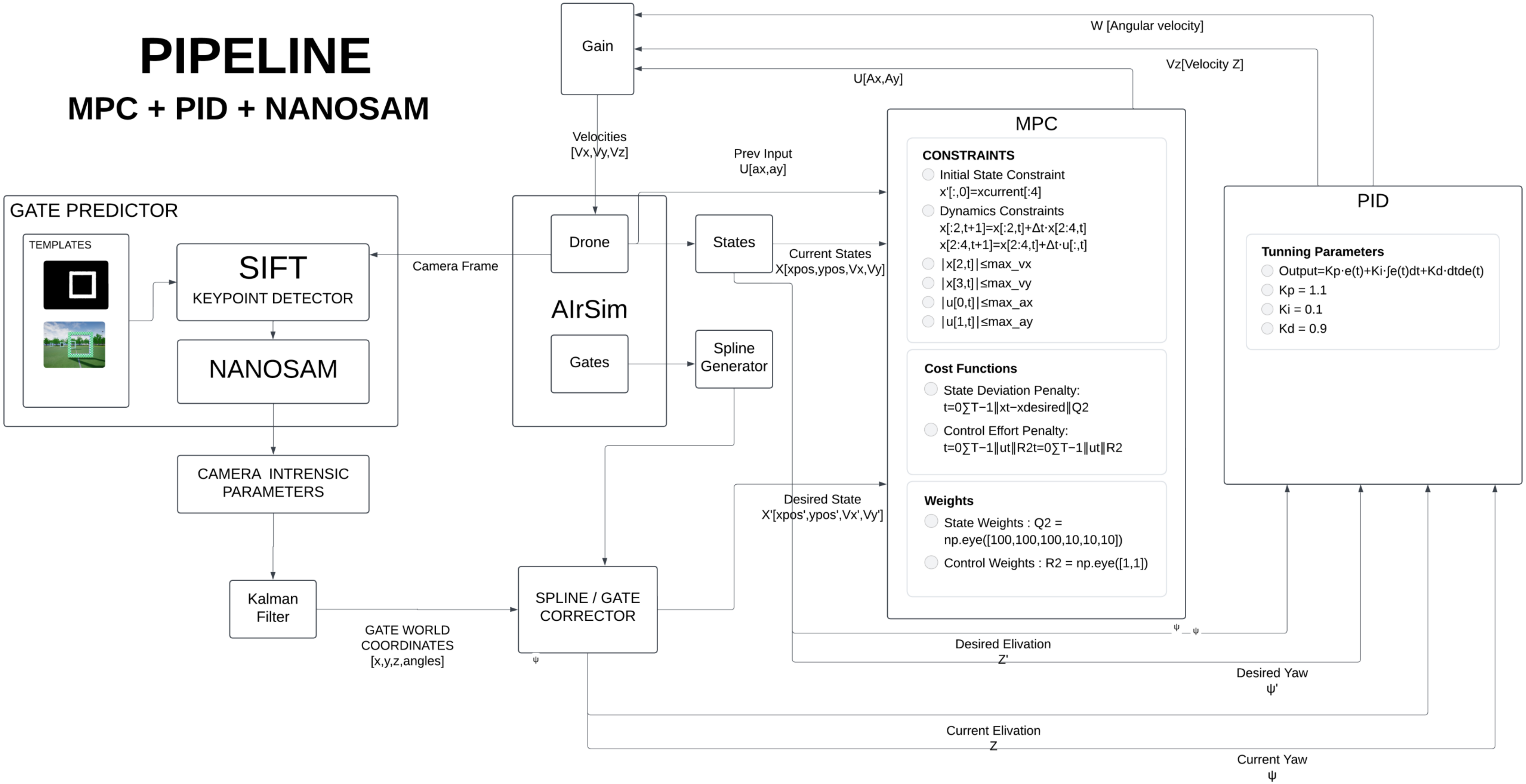


MPC For the Flat System



Cascaded PID Controllers for body elevation and orientation





PIPELINE
MPC + PID + NANOSAM
Gain
W [Angular velocity]
Vz[Velocity Z]
U[Ax,Ay]
Velocities
[Vx,Vy,Vz]
Prev Input
U[ax,ay]
MPC
CONSTRAINTS
Initial State Constraint
x'[:,0]=xcurrent[:4]
Dynamics Constraints
x[:2,t+1]=x[:2,t]+Δt·x[2:4,t]
x[2:4,t+1]=x[2:4,t]+Δt·u[:,t]
|x[2,t]|≤max_vx
|x[3,t]|≤max_vy
|u[0,t]|≤max_ax
|u[1,t]|≤max_ay
Cost Functions
State Deviation Penalty:
t=0∑T−1‖xt−xdesired‖Q2
Control Effort Penalty:
t=0∑T−1‖ut‖R2t=0∑T−1‖ut‖R2
Weights
State Weights : Q2 =
np.eye([100,100,100,10,10,10])
Control Weights : R2 = np.eye([1,1])
PID
Tunning Parameters
Output=Kp·e(t)+Ki·∫e(t)dt+Kd·dtde(t)
Kp = 1.1
Ki = 0.1
Kd = 0.9
GATE PREDICTOR
TEMPLATES
SIFT
KEYPOINT DETECTOR
NANOSAM
Camera Frame
Drone
AIrSim
Gates
States
Current States
X[xpos,ypos,Vx,Vy]
Spline
Generator
CAMERA INTRENSIC
PARAMETERS
Kalman
Filter
GATE WORLD
COORDINATES
[x,y,z,angles]
SPLINE / GATE
CORRECTOR
Desired State
X'[xpos',ypos',Vx',Vy']
Desired Elivation
Z'
Desired Yaw
ψ'
Current Elivation
Z
Current Yaw
ψ

Vision-Guided Trajectory Adjustment

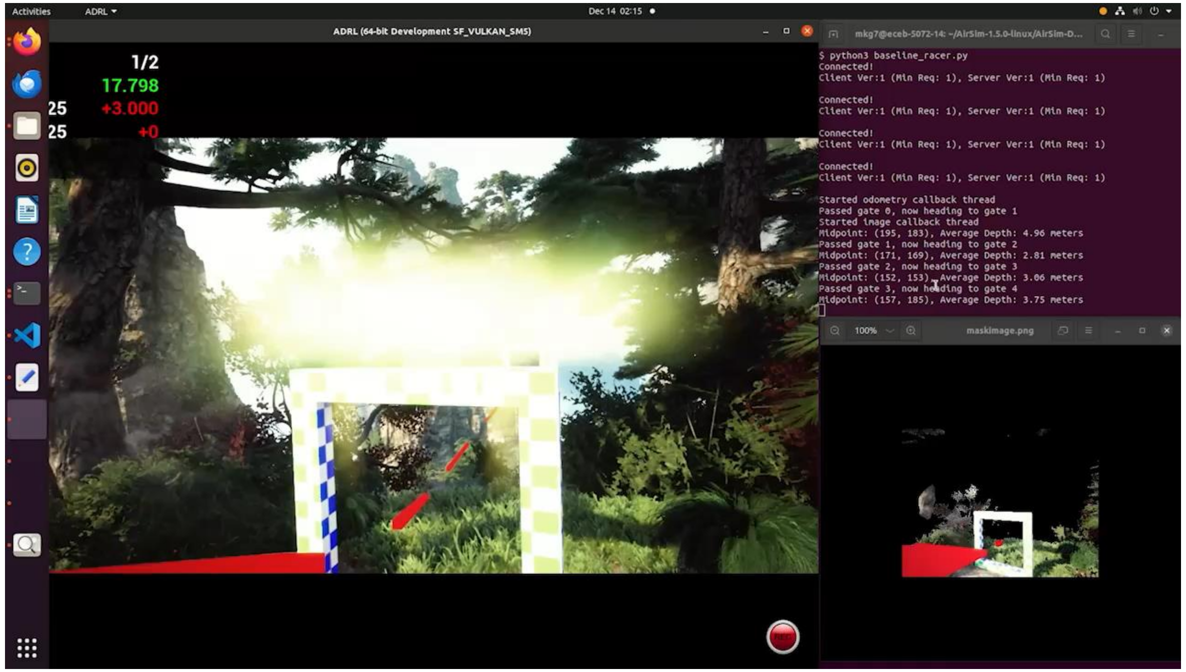
•Use proximity sensor to locate next gate (<3m).

•Activate Nanosam if gate is within range.
•Compute gate's real-world coordinates via projection matrix.

•Check error between measured and reference positions.

•If error > 2m, recalculate spline trajectory.

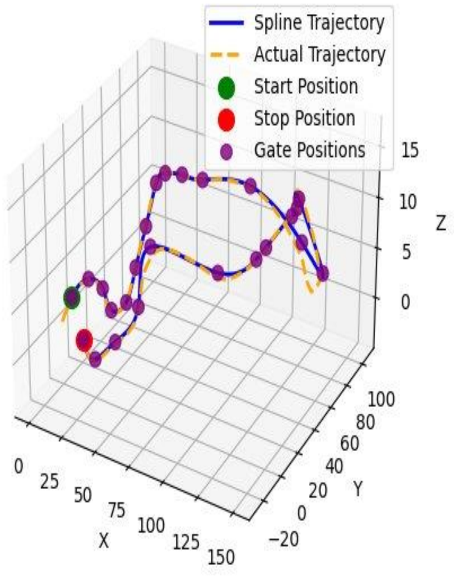
•Initiate control loop.



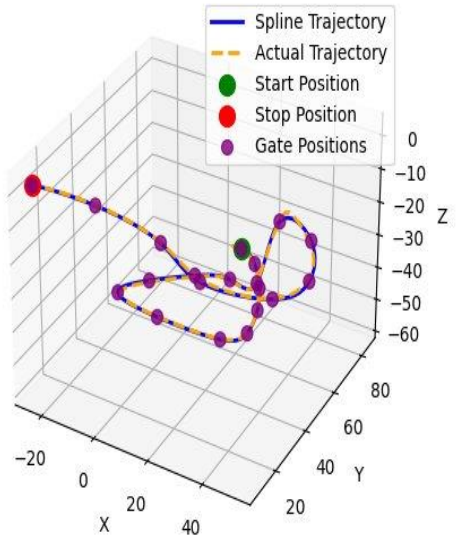
Video LINK

Results

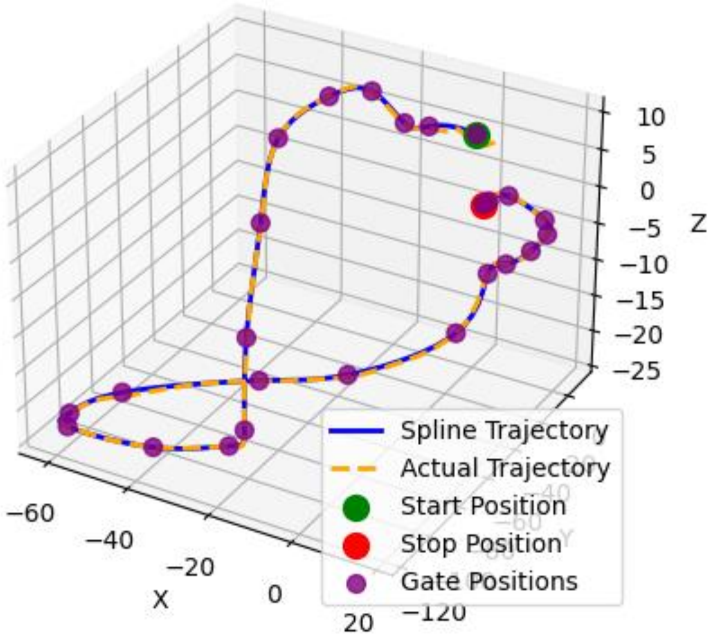
	Soccer easy	Soccer Medium	Qualifier	Zhangjia
Time avg of 10 samples	39.9 seconds	132.77 seconds	104.69	106.94
Gates avg of 10 samples	12 /12	25/25	20/20	24/25
Links	link	link	link	link



Soccer Field Medium



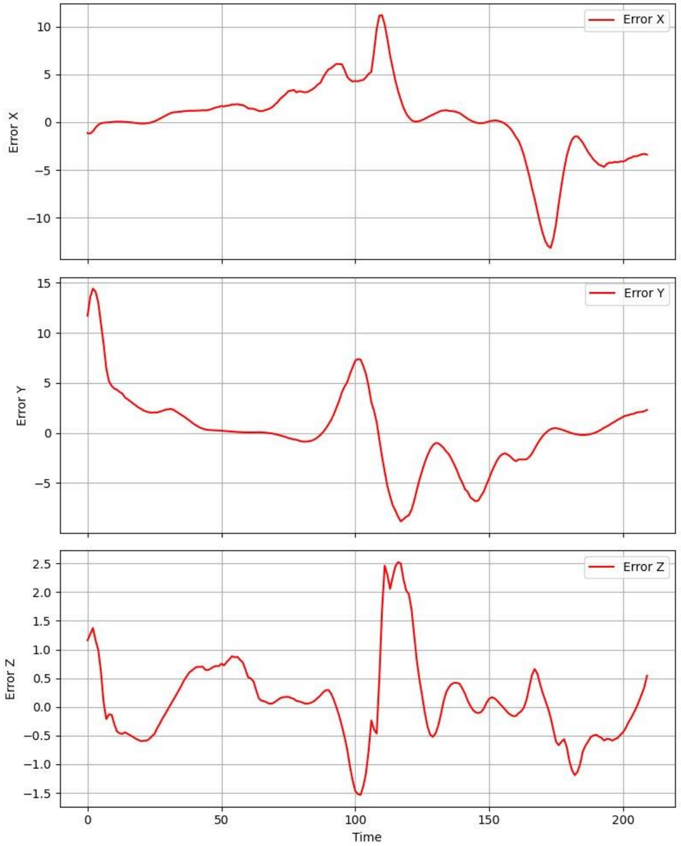
Qualifier



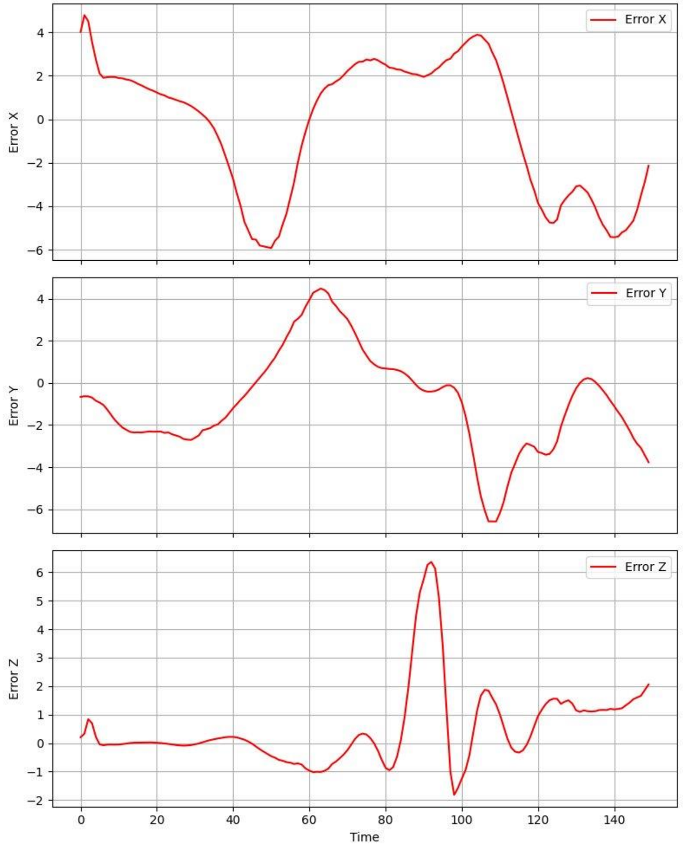
Zhangjia

Metrics

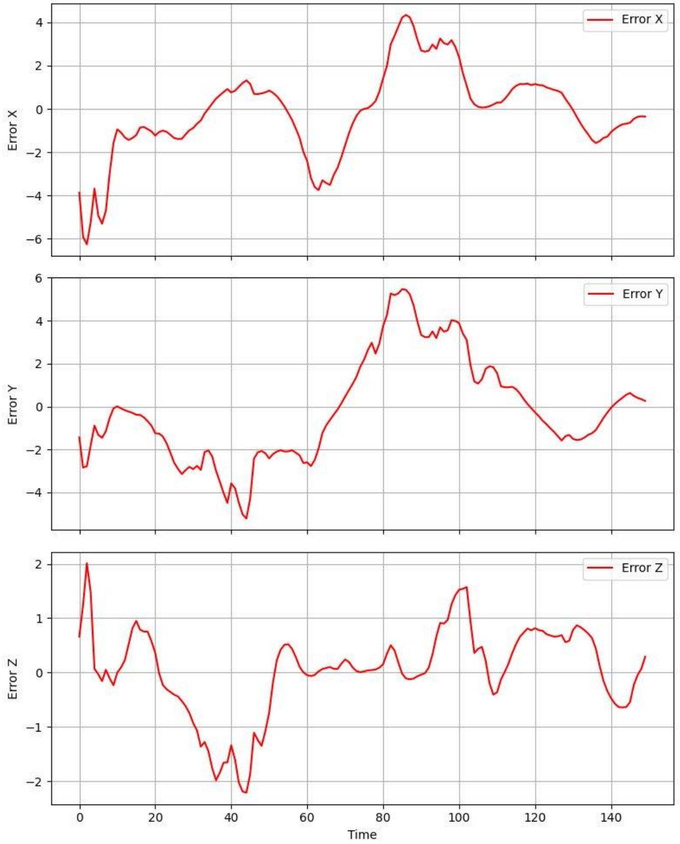
- Disturbances added : wind magnitude rand(3)
- Position Error X,Y,Z



Soccer Field Medium



Qualifier



Zhangjia

Approach & Metric

- **Performance Metrics:**
  - **Time to Complete Course:**
    - Measure the time taken to navigate through all gates.
  - **Accuracy of Navigation:**
    - Track the number of successful gate passes versus attempts.
  - **Response to Disturbances:**
    - Assess the controller's ability to handle unexpected changes (e.g., wind, obstacles).
- **Visual Feedback Metrics:**
  - **Pose Estimation Accuracy:**
    - Evaluate the precision of gate localization using visual data.
  - **Real-Time Processing Efficiency:**
    - Measure latency in image processing and path adjustments.

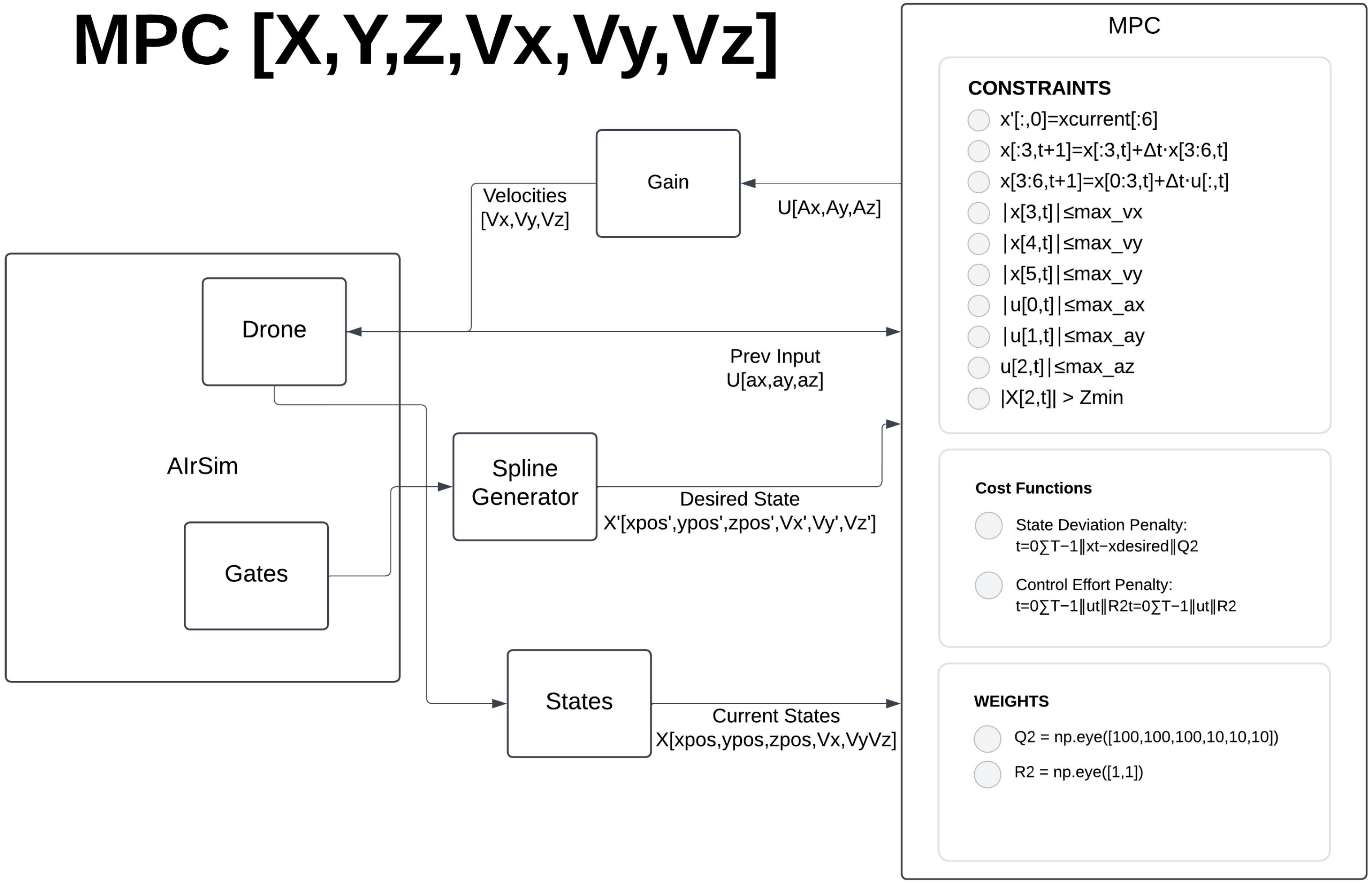
References

- Kaufmann, E., Bauersfeld, L., Loquercio, A. *et al.* Champion-level drone racing using deep reinforcement learning. *Nature* **620**, 982–987 (2023).
- Alvin Shek and Tom Scherlis
- https://alvinosaur.github.io/AboutMe/projects/drone_mpc/final_report.pdf
- https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing/_files/Chuchichaschtli.pdf

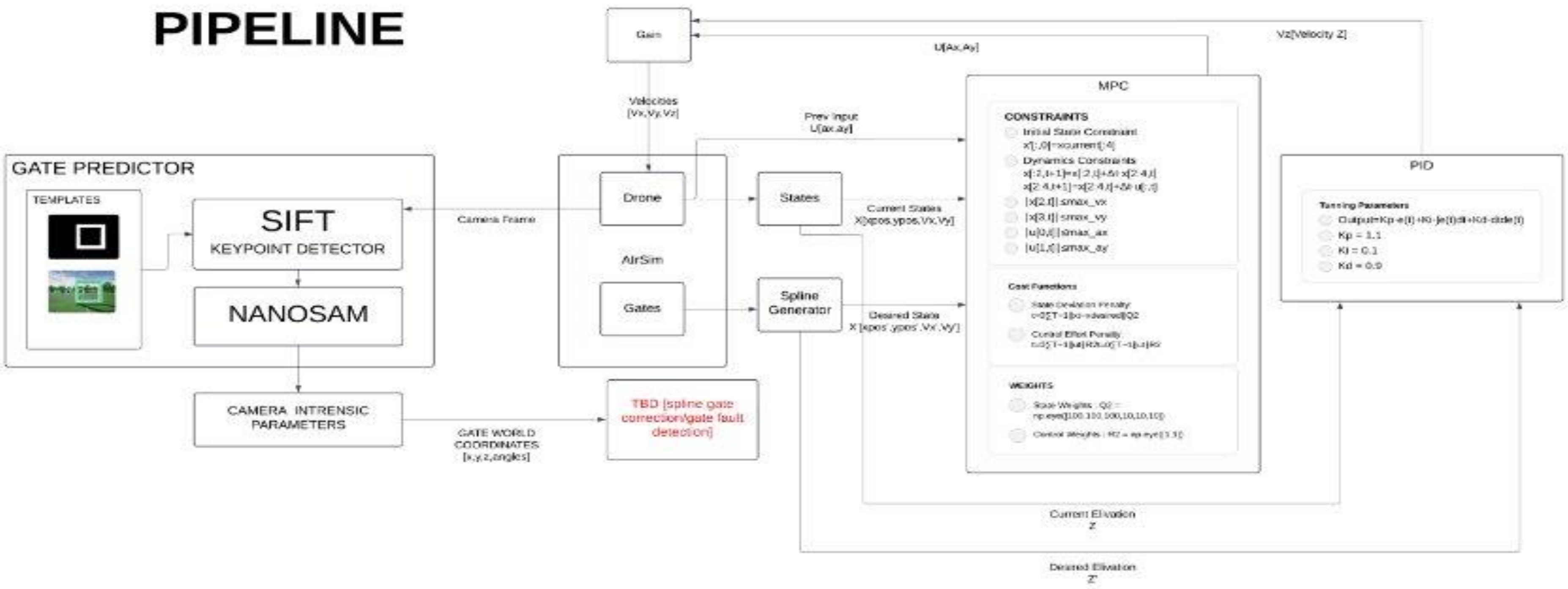
APPENDIX 1 [Failed Approach]

- Failed Approach with MPC in X,Y,Z
  - Mpc failed to calculate optimal solution

MPC [X,Y,Z,Vx,Vy,Vz]

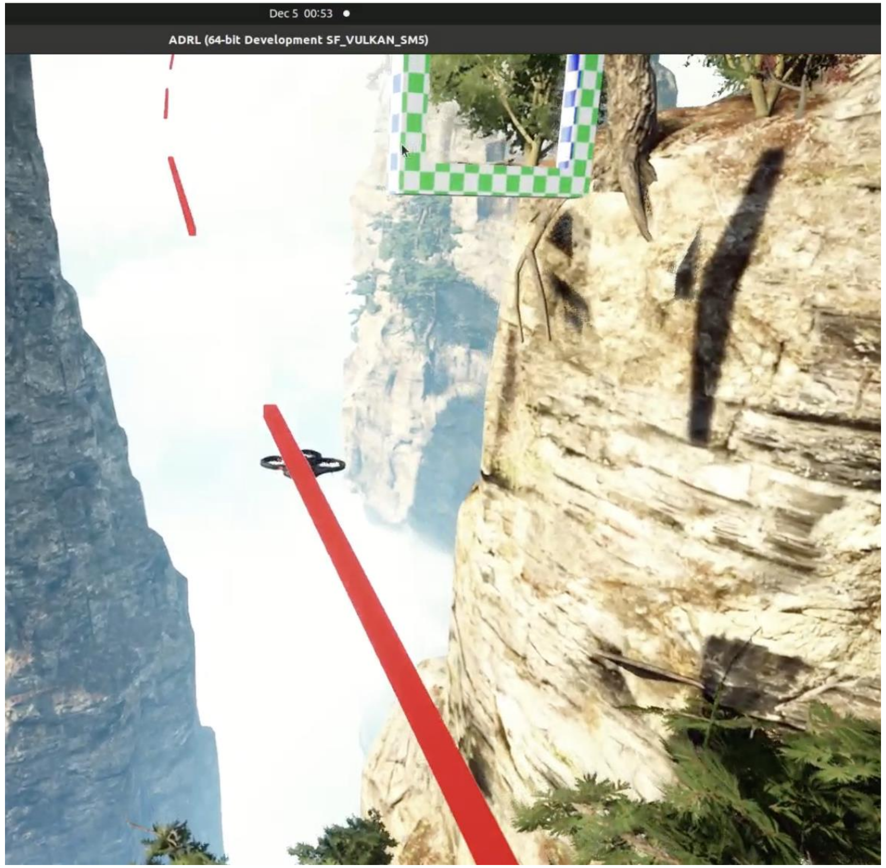


APPENDIX 2 [Failed Approach lack of vision]



APPENDIX 3 past failures [video1] [video2]

- Drawbacks
  - Faulty gate waypoints
    - Results in a incorrect spline Trajectory leading to collisions or missing gates
    - Solution – Get visual feedback and correct gate positions/ spline trajectory
  - Uniform spread of interpolated points across spline
    - Uniform spread of points results a slow moment of drone, sometimes takes the drone outside trajectory
    - Solution – take more points across the trajectory where the degree of curve is higher and less points where degree is lower



Incorrect gate positions lead to incorrect spline trajectory

APPENDIX: Detailed approach

- The problem is composed of two parts
  - **Trajectory Generation**:
    - We collected gate positions and orientations, then created a time-parameterized reference trajectory using interpolation methods for smooth transitions. Cubic Spline Interpolation was used to generate a smooth trajectory. We also experimented with B-splines with N=3 and N=4 polynomials, but the results were not satisfactory.
  - **Controller Design**:
    - We developed a tracking controller that minimizes the error between the current state and the reference trajectory while accounting for system constraints. Initially, Linear MPC was implemented for the x, y, and z directions. However, the MPC optimization failed to compute a low-cost solution (local minimum). To address this, we implemented MPC for the x and y directions and used PID control for the z direction.