Inter IIT Tech Meet 11.0

DRONA AVIATION™

PLUTO DRONE SWARM CHALLENGE

- Team 55

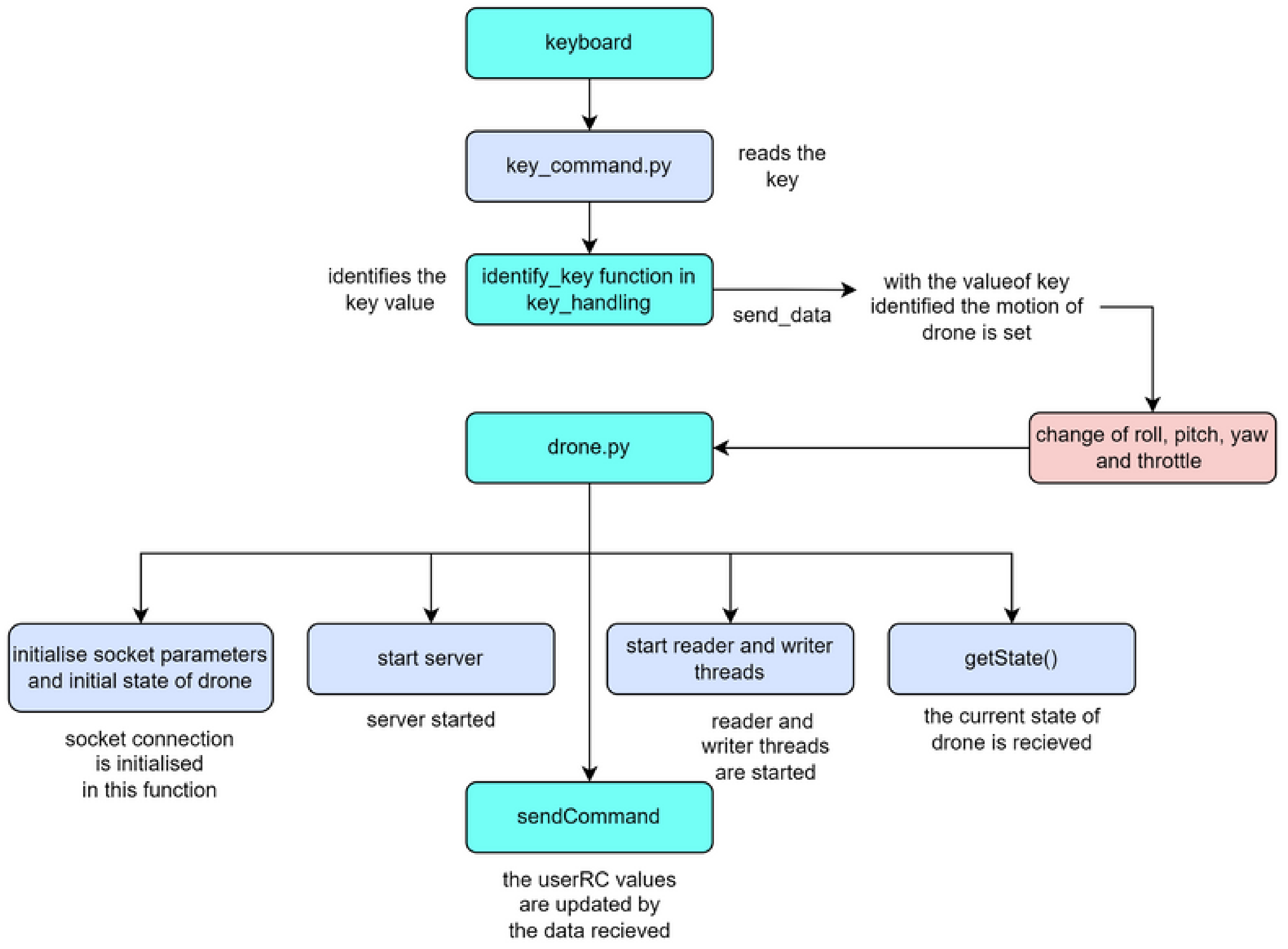
PYTHON WRAPPER

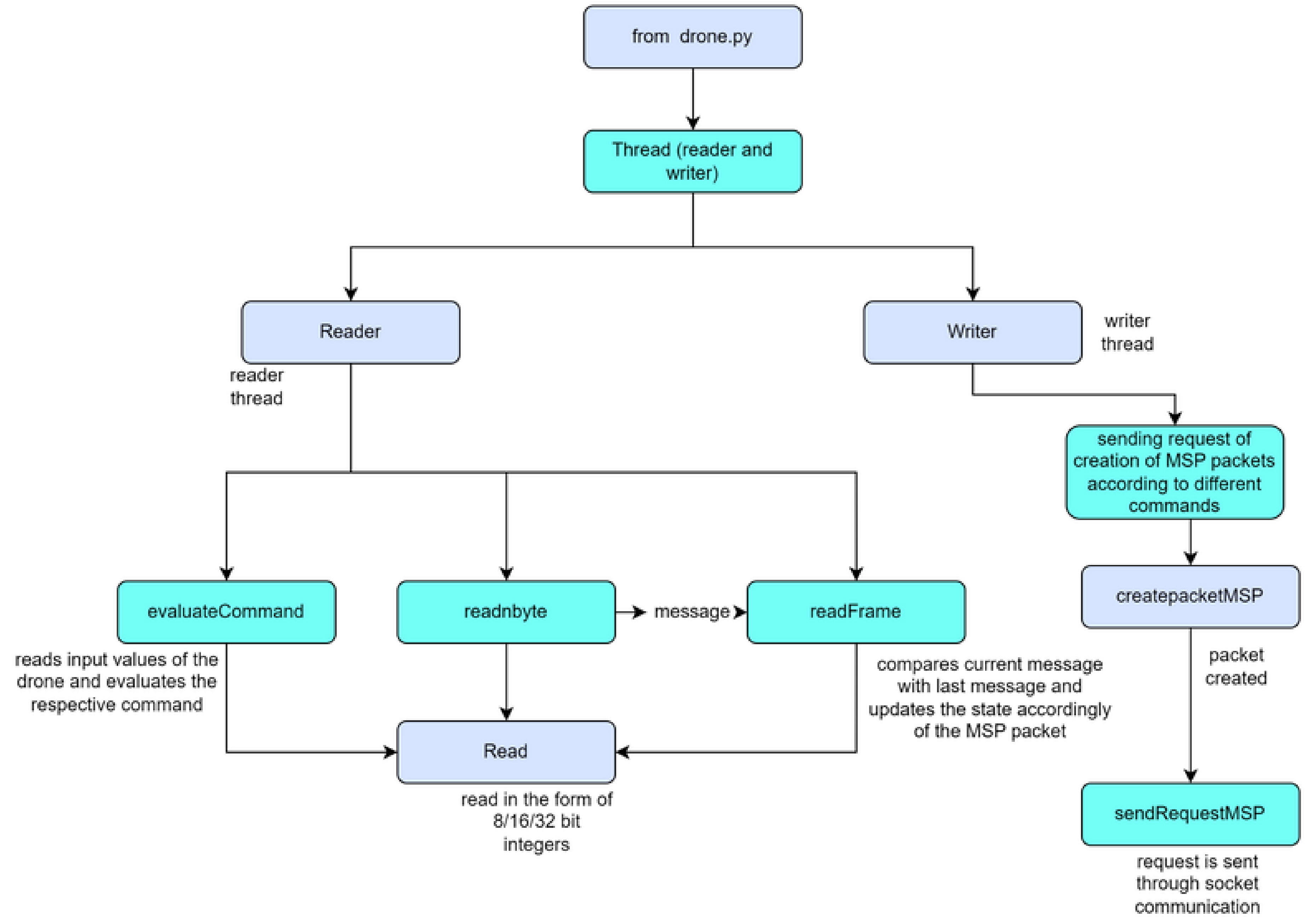
Python wrapper is a code base solely created by our team to control and communicate with the drone without any loss or lag of information.

Highlights of the code

- Object Oriented Program
- Thread Pool Execution
- Data Plotting
- Low Latency
- Safety Measures

- The following two flowcharts explain the **codeflow** and **working of code** in a comprehensive manner describing the working of different threads and data flow between different function calls.


keyboard
key_command.py
reads the key
identifies the key value
identify_key function in key_handling
send_data
with the valueof key identified the motion of drone is set
change of roll, pitch, yaw and throttle
drone.py
initialise socket parameters and initial state of drone
socket connection is initialised in this function
start server
server started
start reader and writer threads
reader and writer threads are started
getState()
the current state of drone is recieved
sendCommand
the userRC values are updated by the data recieved


from drone.py
Thread (reader and writer)
Reader
reader thread
Writer
writer thread
sending request of creation of MSP packets according to different commands
createpacketMSP
packet created
sendRequestMSP
request is sent through socket communication
evaluateCommand
reads input values of the drone and evaluates the respective command
readnbyte
message
readFrame
compares current message with last message and updates the state accordingly of the MSP packet
Read
read in the form of 8/16/32 bit integers

Localization: ArUco Detection and Pose Estimation

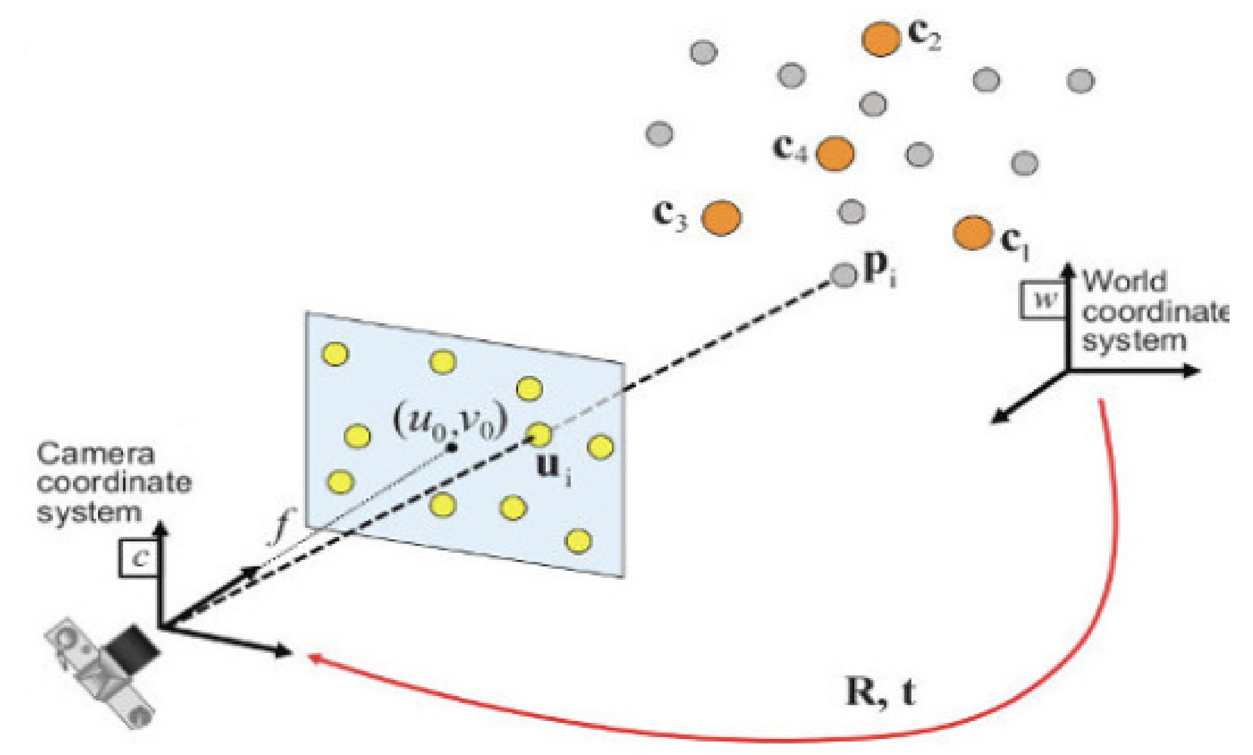
1 **CAMERA CALIBRATION**
Obtained intrinsic parameters of camera. i.e camera matrix and distortion coefficients.

2 **ARUCO DETECTION**
OpenCV library used, which returns the corners of detected marker.

3 **POSE ESTIMATION**
SolvePnP() calculates rotation and translation matrices between camera and marker coordinate system.

4 **ERRORS**
Decent x and y but noisy height estimation.

5 **POSE CORRECTION**
Collected our own data and used machine learning for pose correction



Pose Correction using Machine Learning

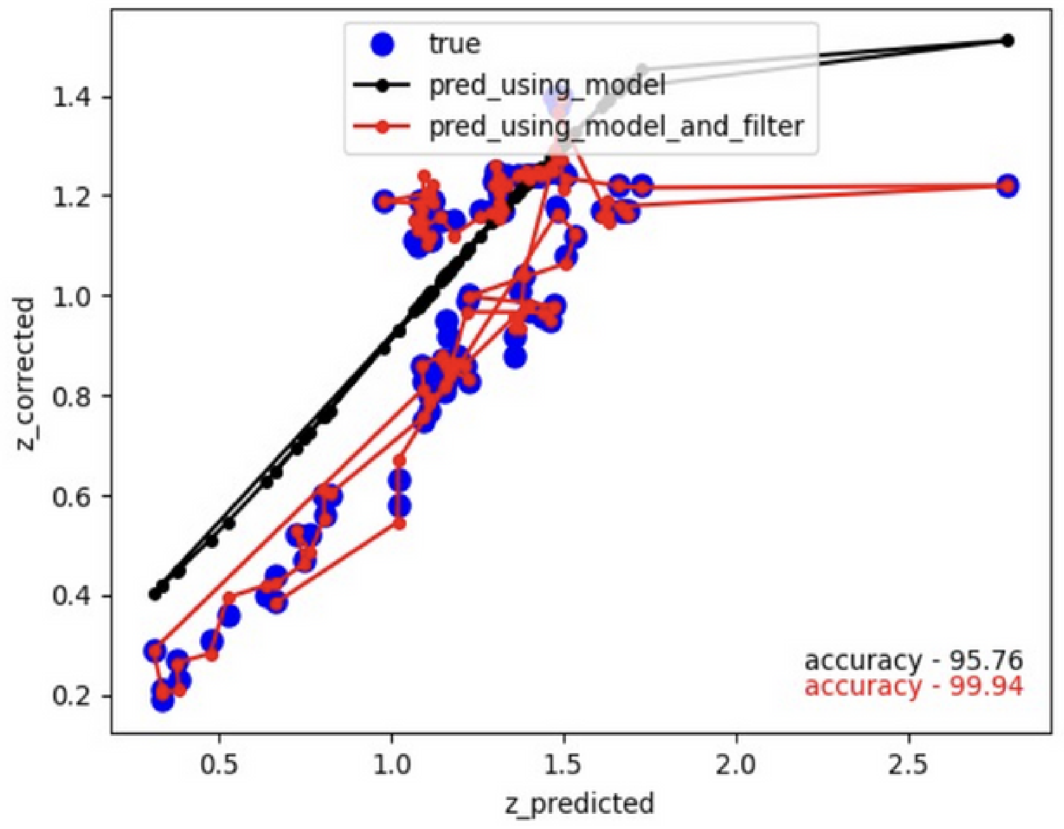
1
Collecte true height using ToF

2
Passed it along with corners and pose

3
Applied Linear Regression

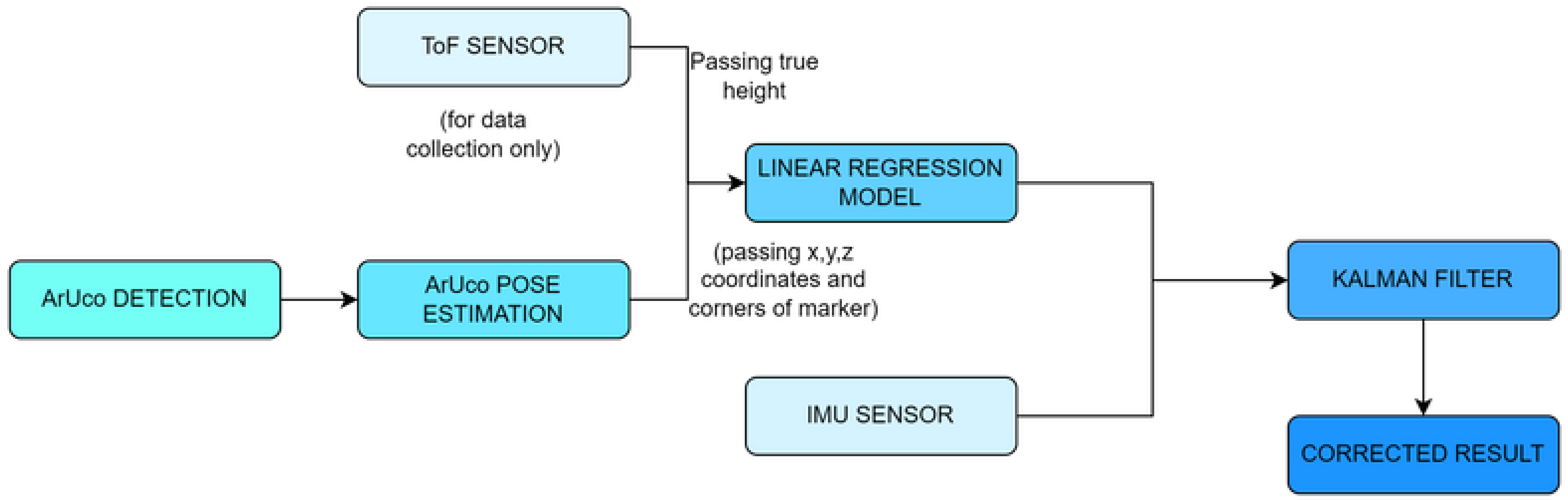
4
Corrected the estimated pose

Minimized error from **6.31%** to **0.036%**



Kalman Filter

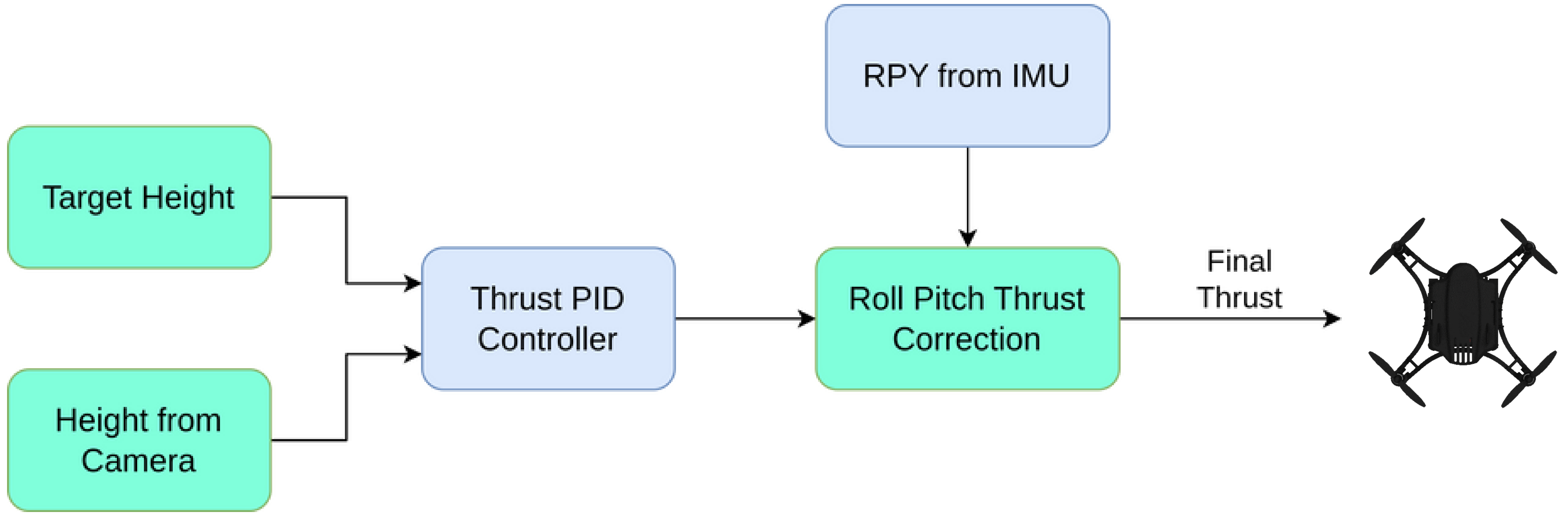
The filter combines current acceleration from the IMU sensor and pose estimate from the regression function to provide an estimation for the next time step.



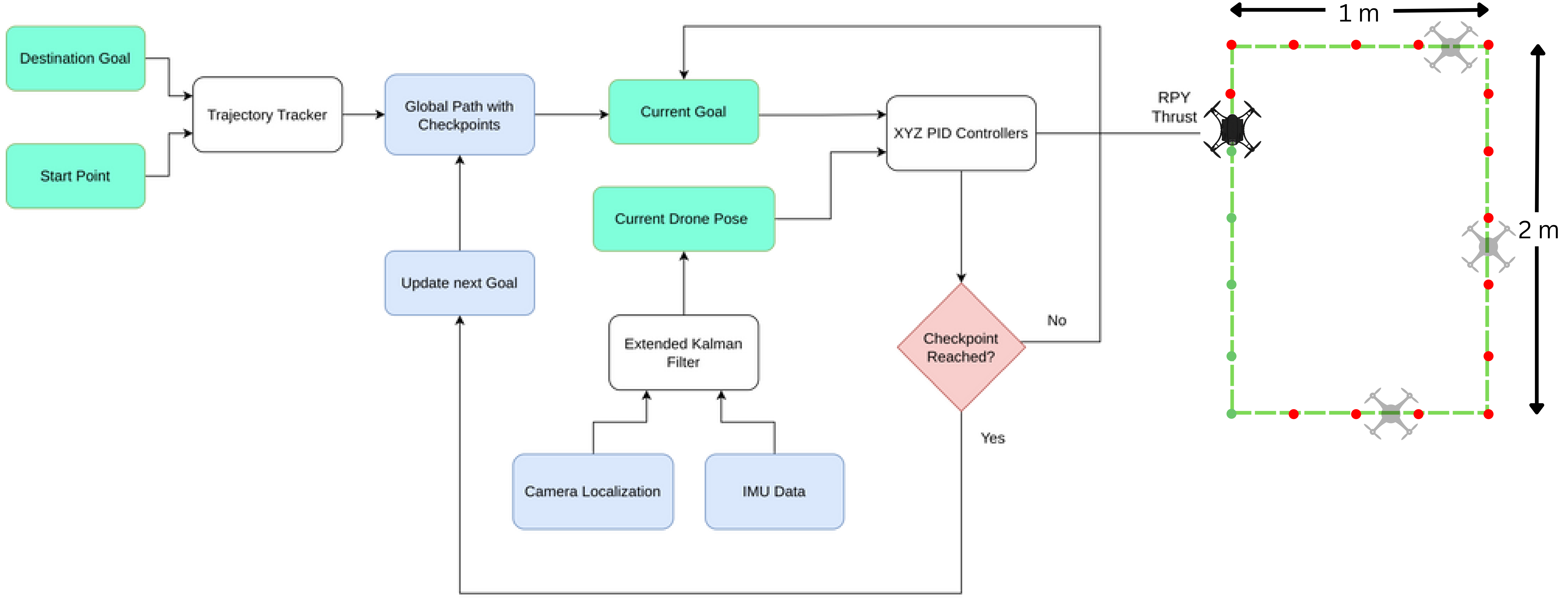
PID Controllers

PID has been implemented to control the movement in the x,y,z directions. PID estimates the value of roll, pitch, and thrust to achieve the required x,y,z coordinates.

We used **trackbars** to dynamically tune PID values and **matplotlib** library to visualize the system's continuous state and target state continuously.



Trajectory Tracking



- Took final destination 3D pose as input
- Broke global path down into local target checkpoints with fixed path resolution.
- A function checks if the current position is within error tolerance limit and updates next local goal point.

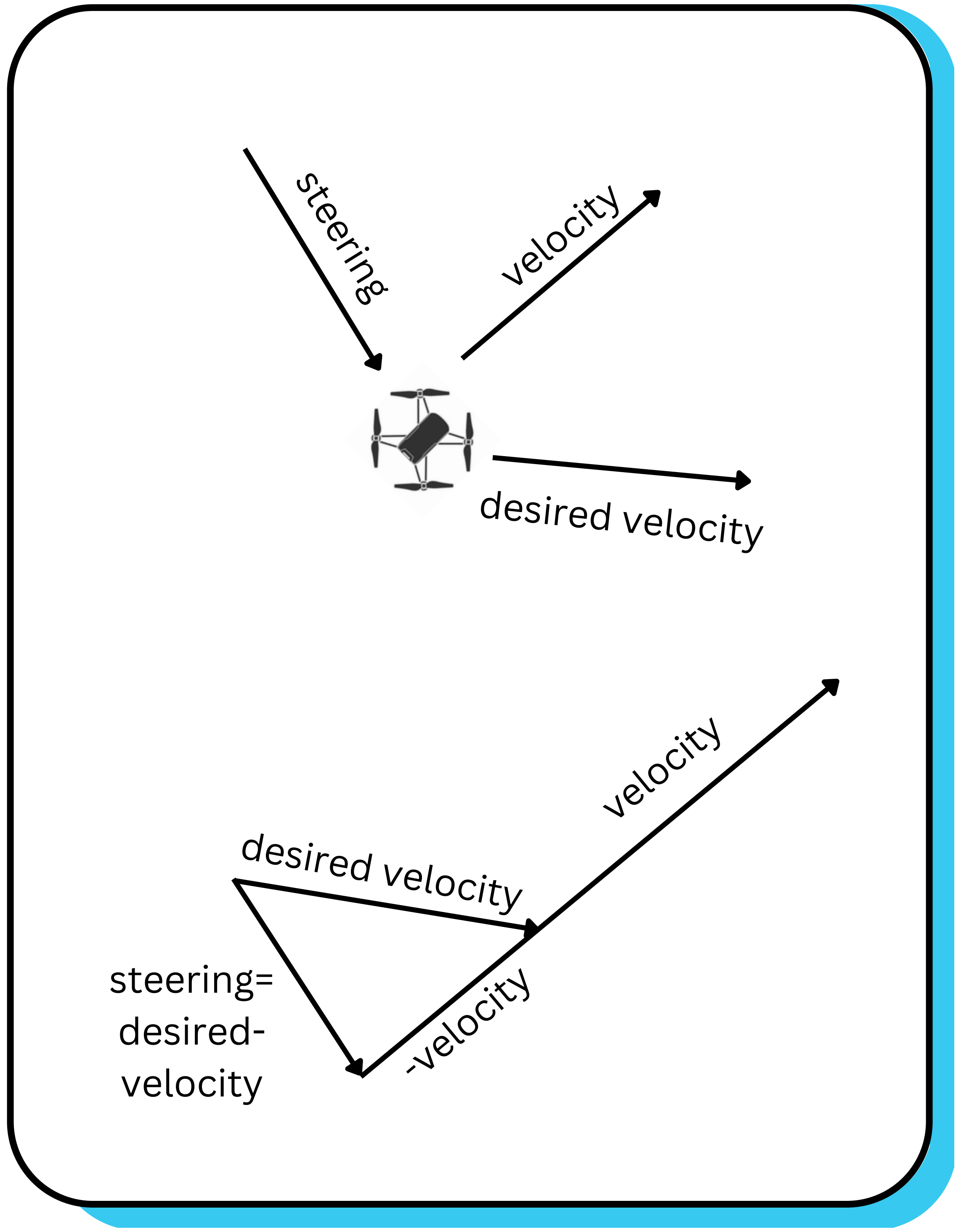
Drone Swarming

Implemented Flocking Algorithm for a swarm of drones to traverse the rectangular trajectory.

Flocking Algorithm

Flocking is inspired from behaviour of birds. The algorithm consists of following three rules:

Alignment	Follower drone moves in velocity direction of primary drone
Cohesion	Follower drone moves toward average position of primary drone
Separation	Follower drone moves away when too close to primary drone



Challenges Faced & Scopes of Improvement

Challenges:

- No external integration options in the flight controller.
- Discontinous ArUco marker detection at varying heights during motion.
- Multi-variable PID tuning on hardware drone.

Improvements:

- Polynomial regression for correcting the estimated x and y coordinates.
- Using more robust controllers like Iterative LQR.
- Using better trajectory tracking algorithms like Backstepping, Feedback Linearization and Lyapunov-based Algorithms.

Conclusion

We are **extremely grateful to Drona Aviation and Inter IIT Tech Meet 11.0** to provide us with an amazing opportunity to explore the domains of **Controls, Localisation, Trajectory Tracking and Swarms using Pluto drones**.

The tasks involved several levels of algorithmic development, hardware implementation and its constraints. We believe we **invested significant time and effort in developing accurate solutions for all tasks**, thereby solving the primary objective of this event.