Development of Vision-Based Perception and Control Systems Using Monocular RGB Images and 3D Simulation

# Technical Report

October 16, 2026

# Introduction

The objective of this work is to develop baseline systems for practical applications by integrating position and orientation estimation from monocular RGB images with control techniques.

The following principles were adopted:

**1. Monocular RGB Images**
Estimate object position and orientation for control.

**2. Target-Specific Estimation Methods**
Select geometric calculations, PnP, or image matching based on target characteristics.

**3. 3D Simulation**
Generate training data, train control algorithms, and validate systems.

**4. Control Methods**
Apply reinforcement learning or PID control depending on the target.

**5. Raspberry Pi 5 Validation**
Evaluate feasibility under limited computational resources.

These approaches establish reusable baseline systems that can be adapted and extended to meet practical requirements.

# Contents


## 1. Yaw Control

1.1 Overview | Controlling Suspended Payload Yaw from 2D Images
1.2 Reinforcement Learning-Based Yaw Control | From Rotation to Stabilization
1.3 Image Recognition | Separating Detection from Geometric Estimation
1.4 Raspberry Pi 5 | From Image Recognition to Closed-Loop Yaw Control

## 2. Fine-Tuning

2.1 Overview | 6D Pose Estimation from Monocular RGB Images
2.2 Fine-Tuning | From Synthetic Images to Real-World Environments
2.3 Ground Truth Generation | Fitting 3D Models to Real Images
2.4 GPU vs. Raspberry Pi 5

## 3. Drone Control

3.1 Overview | Autonomous Delivery of Nine Boxes Using Two Drones
3.2 PID Control
3.3 Image Processing | PnP and Image Matching
3.4 Delivery Planning | Assignment, Sequence, and Timing Using CP-SAT

# Implementation Comparison

This page compares the implementation approaches used in three development projects based on 2D image recognition.

| Item | Yaw Control | Fine-tuning | Drone Control |
|---|---|---|---|
| Objective | Suspended payload yaw control | 6D object pose estimation | Box pickup and delivery |
| Input | Monocular RGB | Monocular RGB | Monocular RGB |
| Training Data | 3D Simulation | 3D Sim + Real Images | 3D Simulation |
| Image Recognition | Keypoint Detection | Object Detection + Real Images | Object Detection + Keypoint Detection |
| **Position / Orientation Estimation** | Geometric Calculation | PnP | PnP + Image Matching |
| **Control** | Reinforcement Learning (SAC) | — | PID |
| **Optimization** | — | — | CP-SAT |
| Raspberry Pi 5 | Validated | Validated | Not Validated* |

Key Design Principles

**1. Image Recognition as a System Component**

Combine image recognition with geometric calculations and PnP.

**2. Simulation as a Development Platform**

Use simulation for data generation, control training, and system validation.

**3. Technology Selection Based on Objectives**

Select appropriate methods for control and optimization.

* Similar detection models have been tested on Raspberry Pi 5, but drone system performance remains unverified.

# 1. Yaw Control

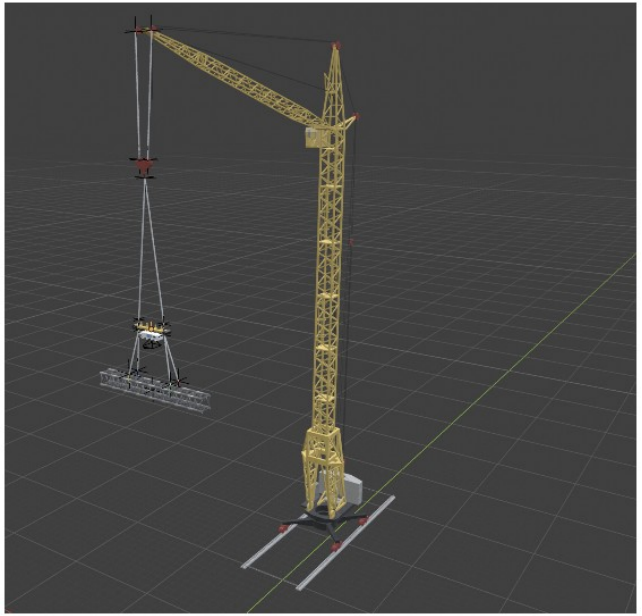
Estimate suspended payload yaw from monocular RGB images and control its orientation using reinforcement learning.

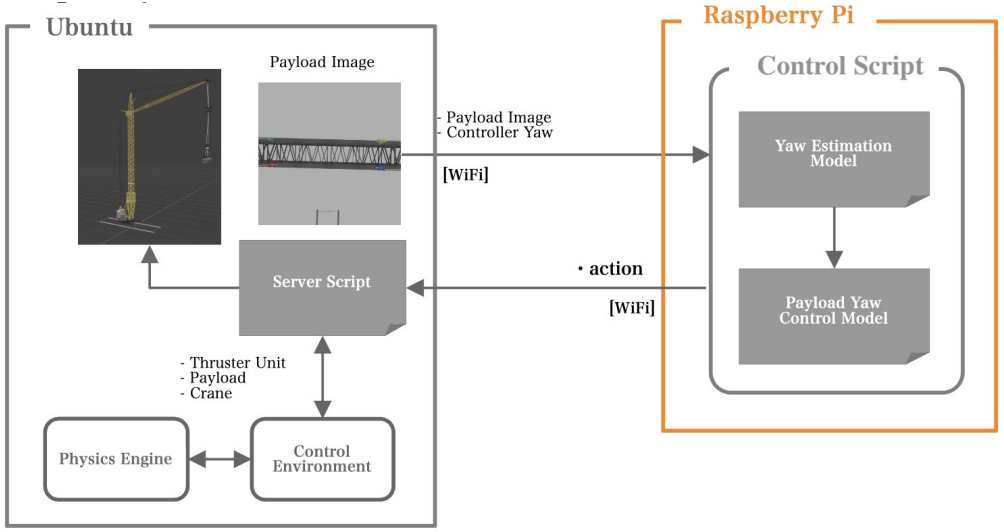
### Publications

GitHub Pages : https://rindajones.github.io/payload-yaw-control/en/
Zenn(Japanese) : https://zenn.dev/rindajones/articles/a8b6189171848f

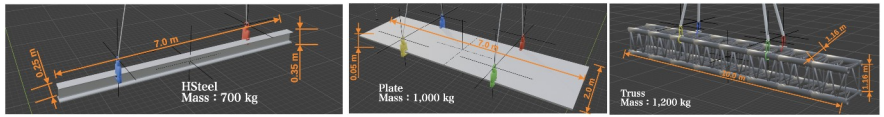
# 1.1 Overview | Suspended Payload Yaw Control from 2D Images

Physics Simulation × Image Recognition × Reinforcement Learning × Raspberry Pi 5






## Target Payloads



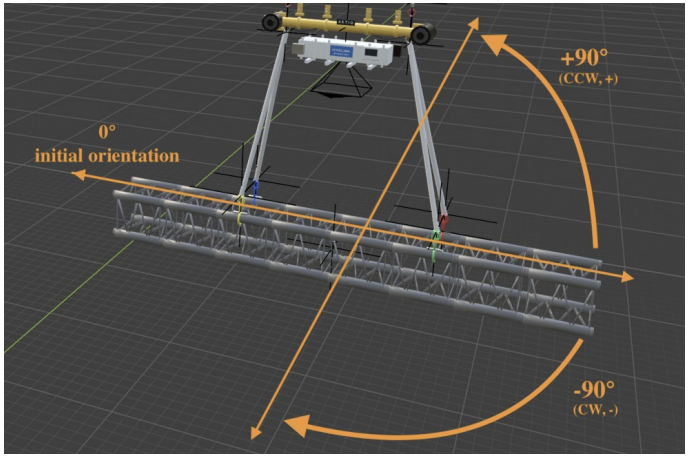

## Control Pipeline

Camera Image → Yaw Estimation → SAC Control → Four Thrusters

# 1.2 Reinforcement Learning-Based Yaw Control

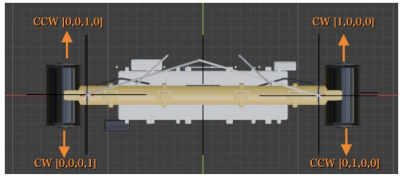
Train continuous thrust control for four thrusters to rotate and stabilize the payload at variable target yaw angles.

### Target Yaw: −90° to +90°




### Control Target

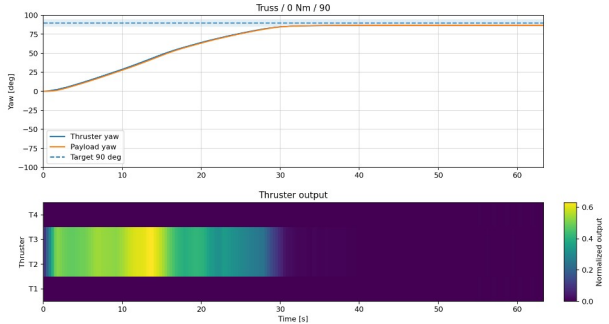
Four Thrusters




### Reinforcement Learning Control

- Algorithm: SAC (Soft Actor-Critic)
- Target yaw: −90° to +90° (variable)
- Continuous thrust commands for four thrusters
- Training and evaluation under varying payload shapes, masses, and disturbances

### Control Results: Target vs. Actual Yaw

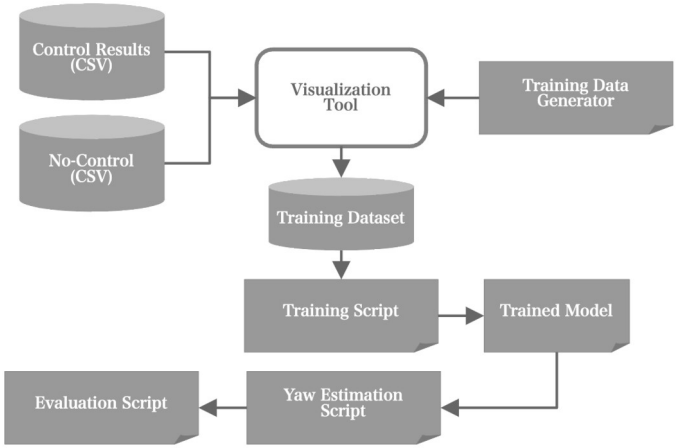



### Results

- Reached the target yaw in approximately 30 seconds
- Reduced thrust as the target was approached
- Maintained yaw with low thrust after reaching the target

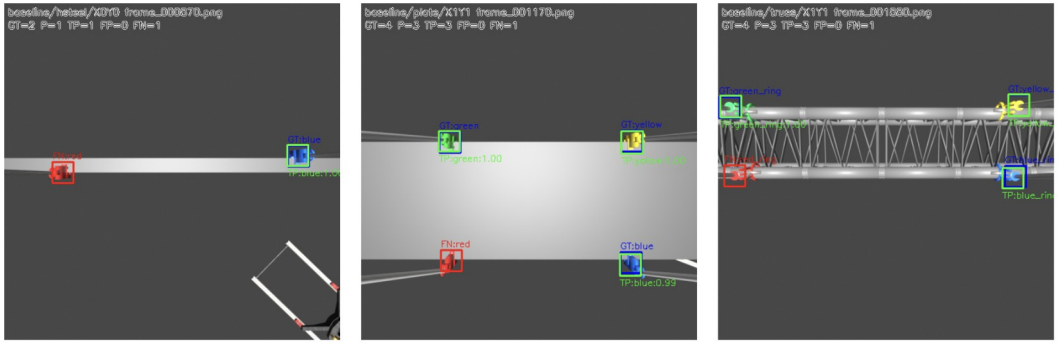
# 1.3 Image Recognition | Detection and Geometric Estimation

2D Image → Hook Detection → Geometric Yaw Estimation → Control

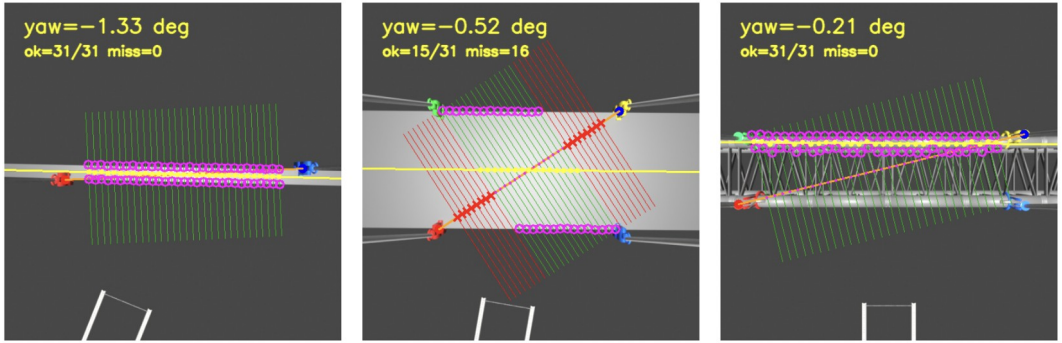



**Key Concept**
Image recognition detects keypoints, while yaw is estimated geometrically.

## 1. Camera Image and Hook Detection



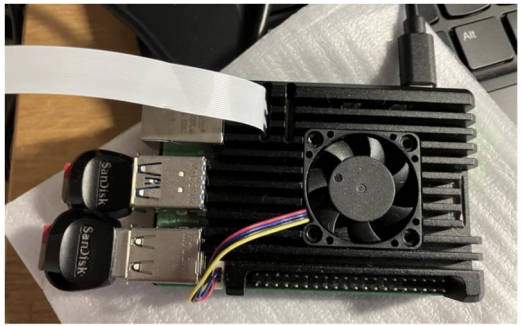
## 2. Yaw Estimation Results

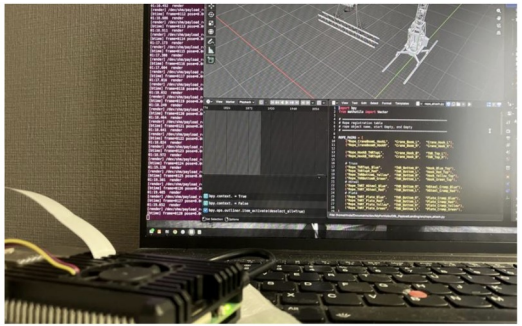


# 1.4 Raspberry Pi 5 | Closed-Loop Yaw Control

Physics simulation runs on Ubuntu, while Raspberry Pi 5 handles image recognition, yaw estimation, and SAC control.

## Raspberry Pi 5 Processing and Connection



### Control Frequency

3 Hz / 333 ms

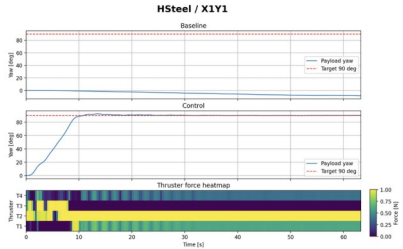
### Processing Time on Raspberry Pi 5

Average: 68.3 ms

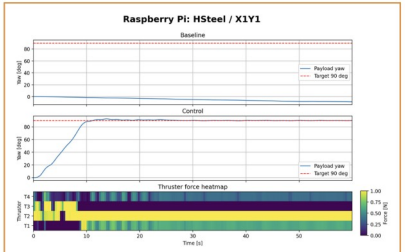
Maximum: 124.0 ms

## Control Results

Ubuntu-Only (Left) vs. Raspberry Pi 5 Closed-Loop Control (Right)




Development Environment




Raspberry Pi

# 2. Fine-tuning

Improve 6D pose estimation by fine-tuning models trained on synthetic images with real-world images.

### Publication

GitHub: https://rindajones.github.io/projects/6d-pose-estimation/

# 2.1 Overview | 6D Pose Estimation from Monocular RGB Images

Object Detection → Keypoint Detection → 6D Pose Estimation (PnP)



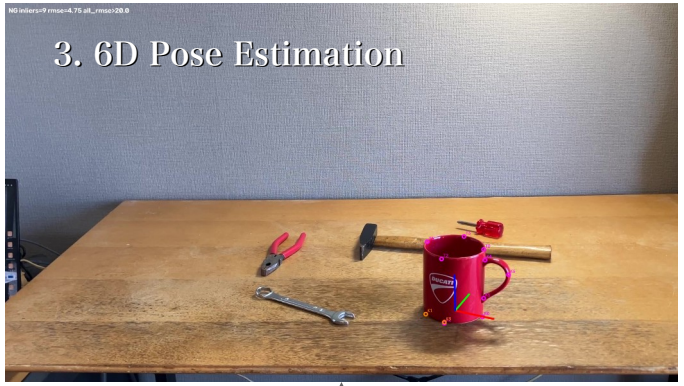




Image Cropping



# 2.2 Fine-Tuning | From Synthetic to Real Images

Generate training data using Blender and fine-tune with a small set of real images.

### Synthetic Images (Sim)



### Real Image



**Training Pipeline**

Pre-training on 3,000 Synthetic Images

↓

3D Model Fitting and Ground Truth Generation

↓

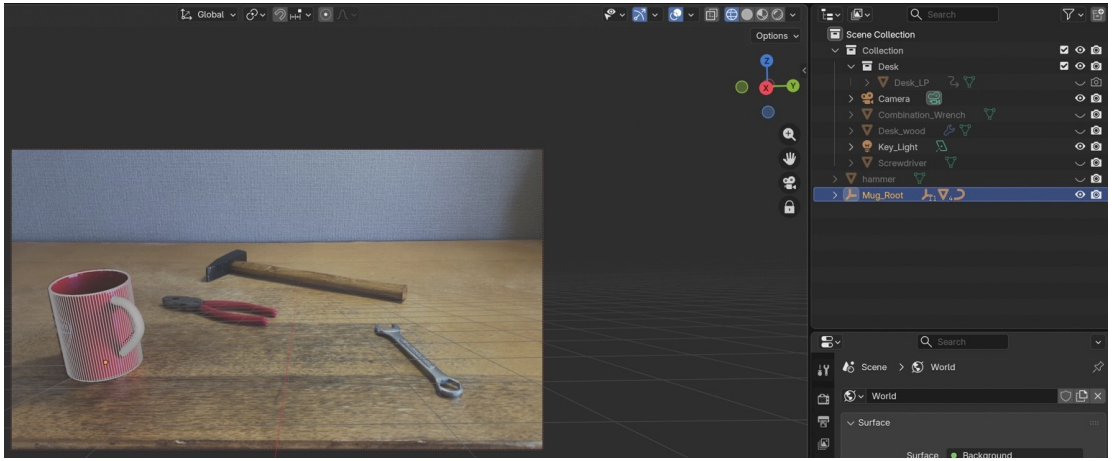
Fine-Tuning on 173 Real Images

**YouTube Videos**

https://youtube.com/shorts/THVDTcLsF0w?feature=share
https://youtu.be/3KiCMTUQuoI

# 2.3 Ground Truth Generation | Fitting 3D Models to Real Images

Project 3D model keypoints to generate consistent ground truth annotations that are difficult to label manually.




## Workflow

1. Align the 3D model with a real image.
2. Project predefined 3D keypoints.
3. Generate 2D ground truth keypoints.
4. Fine-tune the model.

## Real Image + 3D Model Alignment

Accuracy depends on model alignment, camera conditions, and 3D geometry.

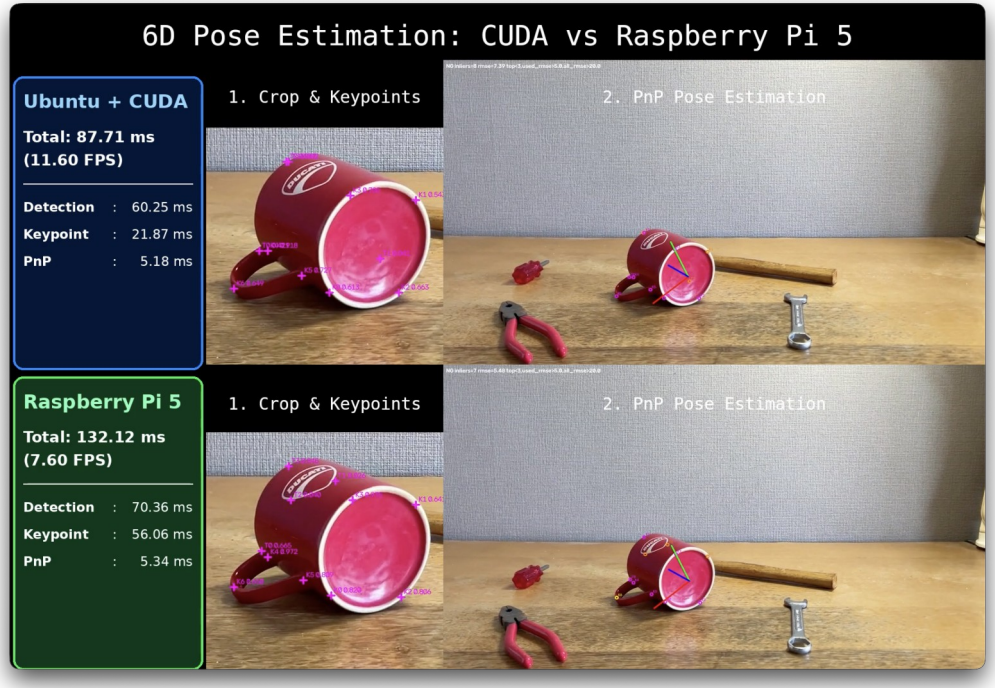
## Results

Improved the consistency of ground truth annotations compared with manual labeling.

## YouTube Video

https://youtu.be/rC1AJGKOdmE

# 2.4 GPU vs. Raspberry Pi 5




Compare processing speeds using the same image recognition models and pose estimation algorithms on GPU (CUDA) and Raspberry Pi 5.

| Process | GPU (CUDA) | Raspberry Pi 5 |
|---|---|---|
| Detection | 60.25 ms | 70.36 ms |
| Keypoint | 21.87 ms | 56.06 ms |
| PnP | 5.18 ms | 5.34 ms |
| Total | 87.71 ms | 132.12 ms |

## Results

Raspberry Pi 5 achieved 7.60 FPS. The main performance gap was in keypoint detection, while PnP performance was nearly identical.

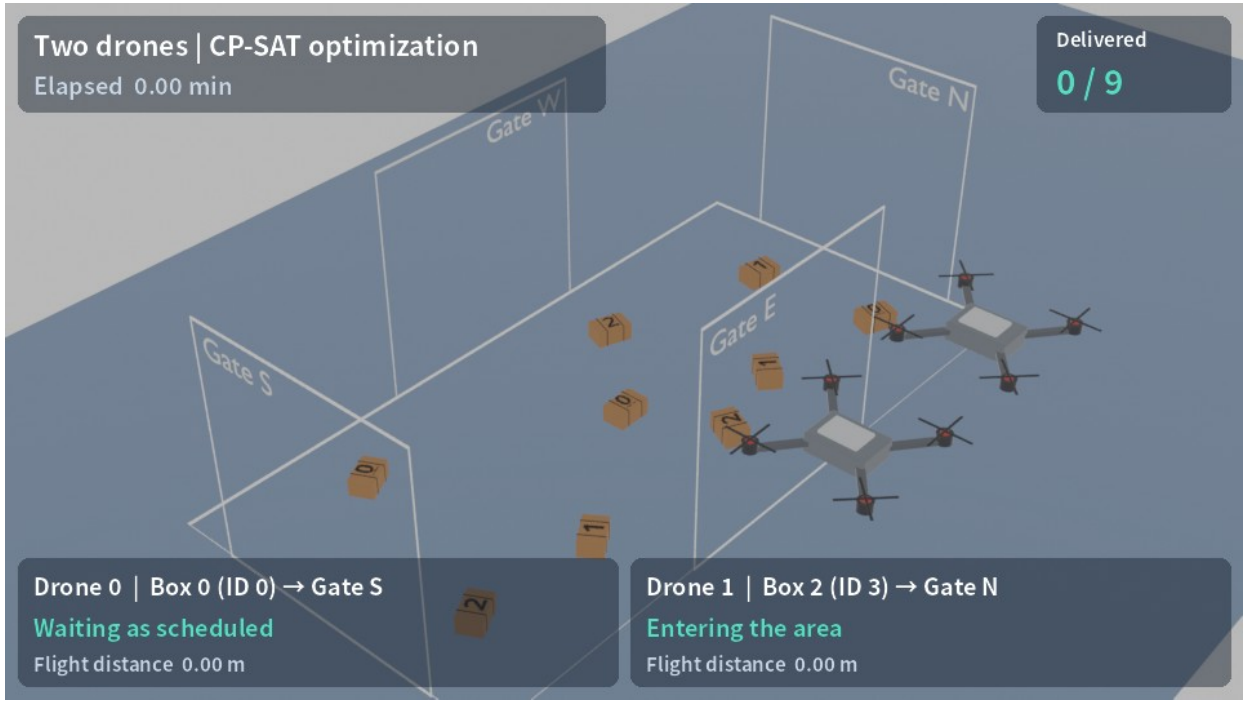
**YouTube Video**
https://youtu.be/zFwZ-Rls9bs

# 3. Drone Control

Integrate vision-based position estimation, PID control, and delivery planning for autonomous multi-drone delivery.

# 3.1 Overview | Autonomous Delivery of Nine Boxes Using Two Drones

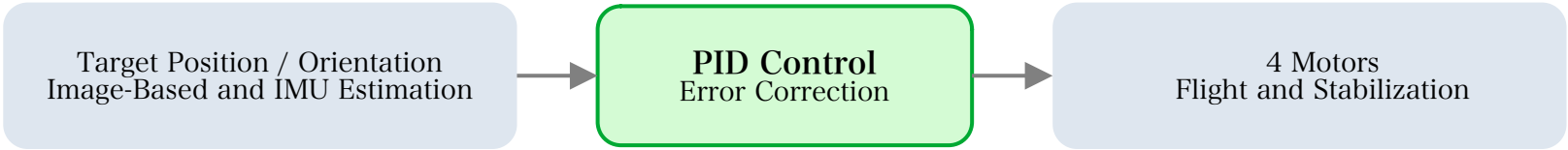
Use image recognition for box alignment, PID for flight control, and CP-SAT for delivery planning.



# 3.2 PID Control

Control drone motor thrust using PID based on errors between target and estimated states.

**Drone Control with PID**



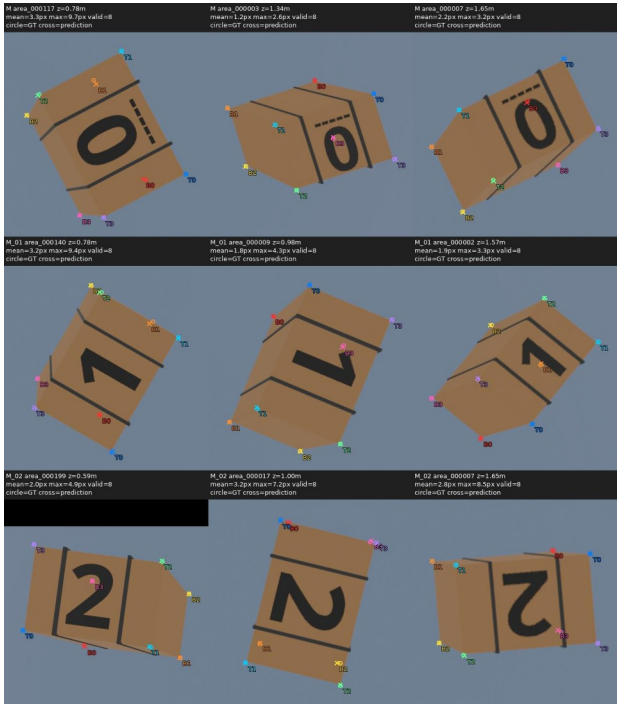

**Control Architecture**

- State Estimation: Estimate box-relative position and orientation from images, and drone attitude and angular velocity from IMU.
- PID Control: Adjust motor thrust based on position and orientation errors.
- Multi-Rate Control: Maintain IMU-based state estimation and attitude control between image updates.

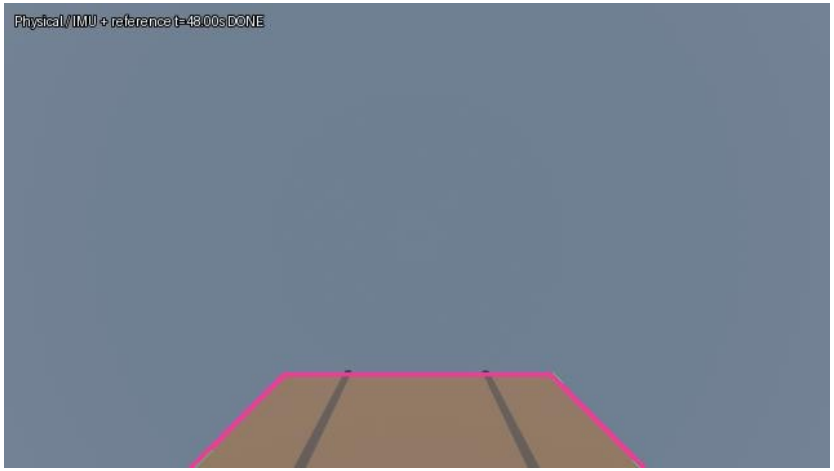
*Simulation framework: PyBullet / gym-pybullet-drones.*

# 3.3 Image Processing | PnP and Image Matching

Approach: Estimate box-relative position and orientation using keypoints and PnP.



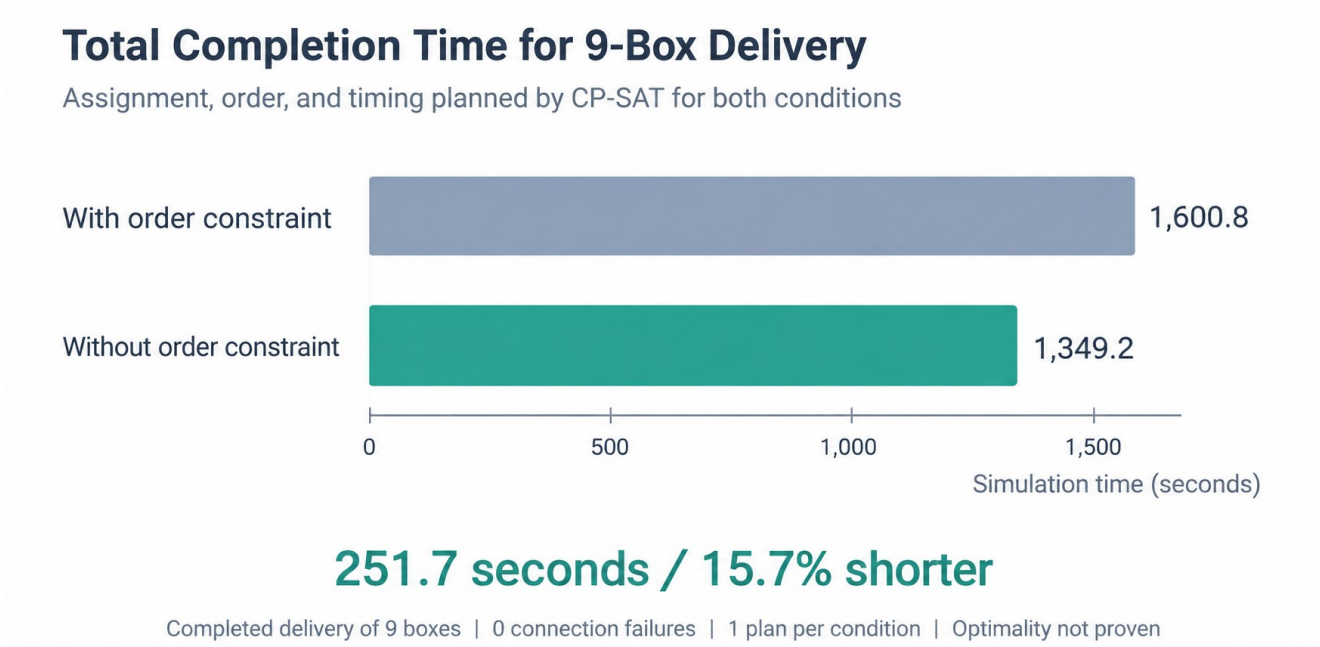
Before Pickup: Align with the box using reference image matching.




**YouTube Video**
https://youtu.be/Tr1Pd5xofdM

# 3.4 Delivery Planning | Assignment, Sequence, and Timing with CP-SAT

Plan box assignments, delivery sequences, and waiting times for two drones.
Removing the numerical delivery order constraint reduced total delivery time by 15.7% in this simulation.




**YouTube Video**
https://youtu.be/XLUYuL7K9-U