単眼RGB画像と3Dシミュレーションを活用した認識・制御システムの開発

技術報告書

2026年10月16日

はじめに

本開発では、単眼RGB画像による位置・姿勢推定と制御を組み合わせ、実務への応用を想定したベースラインシステムの構築を目的とする。

開発にあたっては、以下の方針を採用した。

1. **単眼RGB画像の活用**
   画像認識によって対象物の位置・姿勢を推定し、制御に利用する。

2. **対象に応じた推定手法の選択**
   幾何計算、PnP、画像マッチングなどを対象の特性に応じて使い分ける。

3. **3Dシミュレーションの活用**
   学習データの生成、制御アルゴリズムの学習、システム全体の検証に活用する。

4. **制御方式の選択**
   制御対象に応じて、強化学習またはPID制御を採用する。

5. **Raspberry Pi 5での動作検証**
   限られた計算資源での実装可能性を検証する。

これらの技術を組み合わせたシステムを構築・評価し、実務上の要件に応じて応用・発展できるベースラインを確立する。

目次

1. Yaw 制御

1.1 全体概要｜2D画像から吊り荷のYaw角を制御
1.2 強化学習によるYaw制御｜旋回から姿勢維持まで
1.3 画像認識手法｜検出と幾何計算を分離
1.4 Raspberry Pi 5｜画像認識から閉ループYaw制御まで

2. Fine-tuning

2.1 全体概要｜単眼RGB画像から6D Poseを推定
2.2 Fine-tuning｜合成画像から実写環境へ
2.3 教師データ生成｜3Dモデルを実写画像にフィット
2.4 GPU vs Raspberry Pi 5

3. ドローン制御

3.1 概要｜2機で9箱を自動配送
3.2 PID 制御
3.3 画像処理｜PnP＋画像マッチング
3.4 配送計画｜CP-SATで担当・順序・時刻を決定

実装比較

2D画像認識を用いた3つの開発事例について、実装方法を比較する。

比較項目	Yaw制御	Fine-tuning	ドローン制御
目的	吊り荷の Yaw角制御	物体の 6D姿勢推定	箱の把持・配送
入力画像	単眼RGB	単眼RGB	単眼RGB
学習データ	3D Sim	3D Sim + 実写	3D Sim
画像認識	特徴点検出	物体検出 ＋特徴点検出	物体検出 ＋特徴点検出
位置・姿勢推定	幾何計算	PnP	PnP＋画像マッチング
制御方式	強化学習(SAC)	—	PID
最適化	—	—	CP-SAT
Raspberry Pi5	検証済み	検証済み	未検証 (注)

1. 2D画像認識をシステムの一部として利用

画像認識モデルにすべてを任せず、検出結果を幾何計算やPnPなどを組合わせた。

2. シミュレーションを開発基盤として活用

学習データの生成だけでなく、制御アルゴリズムの学習やシステム全体の検証にも利用。

3. 目的に応じて異なる技術を選択

制御・最適化には、それぞれの課題に適した手法を採用。

注：同系統の検知モデルを使用しており、Raspberry Pi 5での実装可能性がある。
ただし、処理速度を含む検証は未実施。

1. Yaw 制御

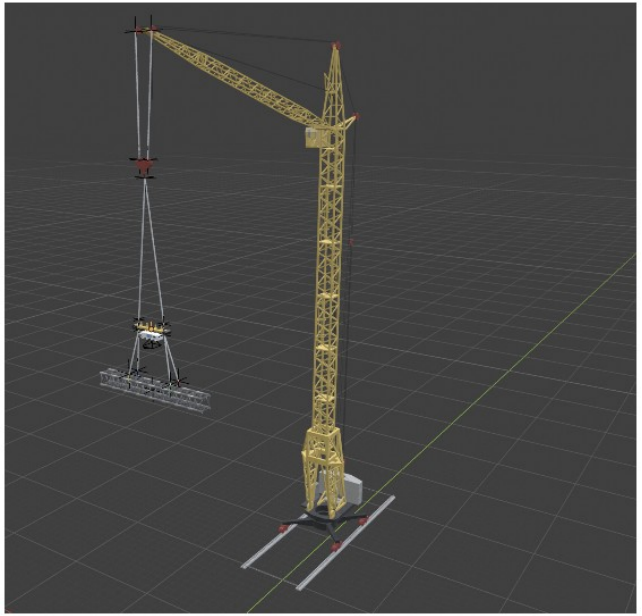
単眼RGB画像から吊り荷のYaw角を推定し、強化学習による姿勢制御を実現する。

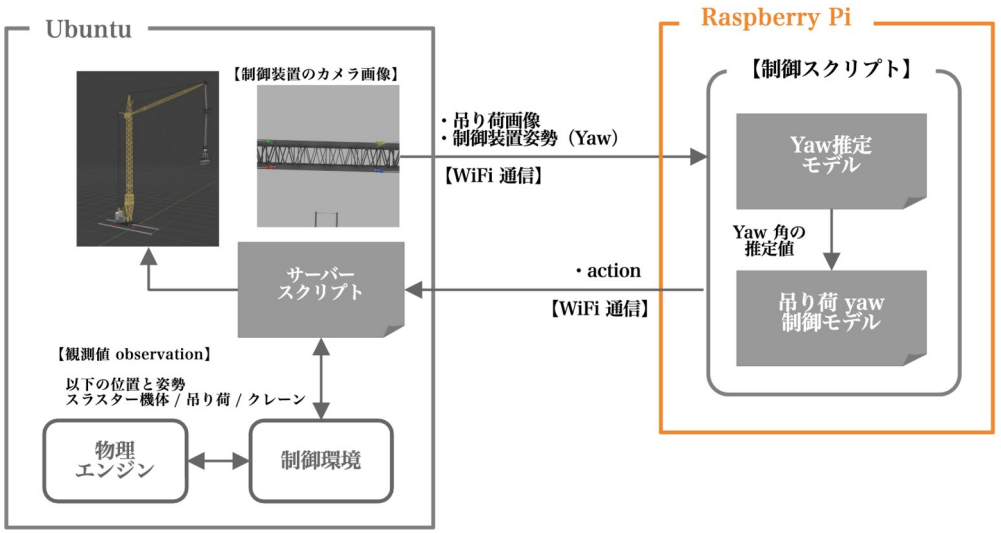
公開資料

GitHub Pages：https://rindajones.github.io/payload-yaw-control/ja/
Zenn：https://zenn.dev/rindajones/articles/a8b6189171848f

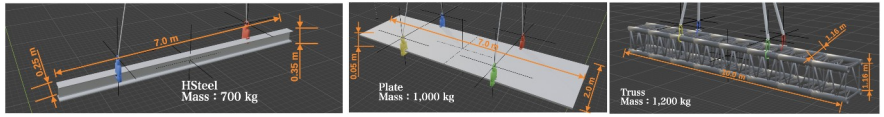
1.1 全体概要｜2D画像から吊り荷のYaw角を制御

物理シミュレーション × 画像認識 × 強化学習 × Raspberry Pi





対象とする吊り荷

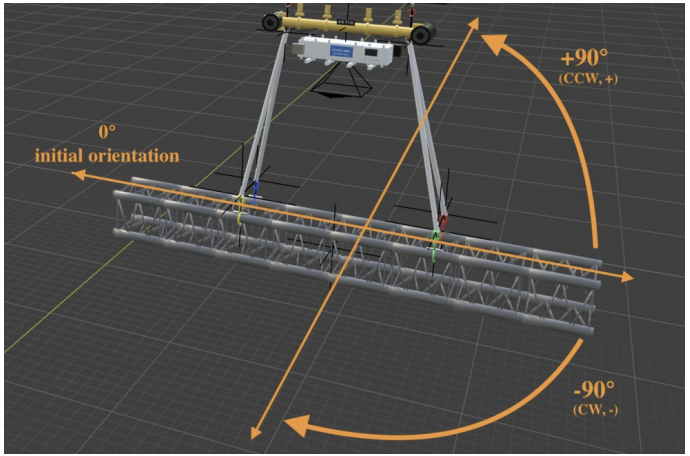


カメラ画像 → Yaw推定 → SAC制御 → 4基のスラスター

1.2 強化学習によるYaw制御｜旋回から姿勢維持まで

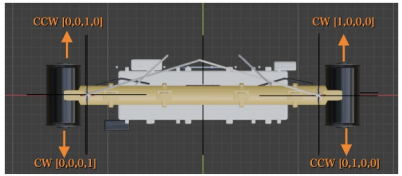
強化学習によって4基のスラスターの連続推力制御を学習し、可変目標Yaw角への旋回と姿勢維持を実現。

目標Yaw角 −90°〜＋90°



【制御対象】

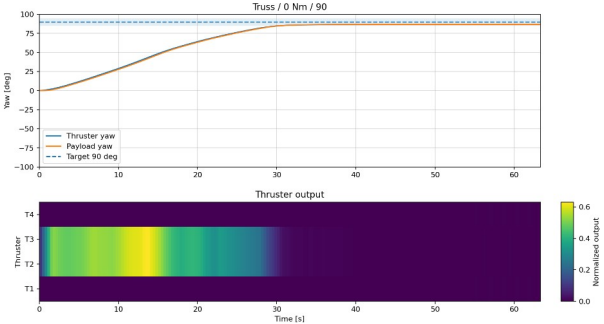
4基のスラスター



【強化学習による制御】

- アルゴリズム：SAC（Soft Actor-Critic）
- 目標Yaw角：−90°〜＋90°（可変）
- 4基のスラスターに対する連続推力指令を学習
- 吊り荷の形状・質量・外乱条件を変えて学習・評価

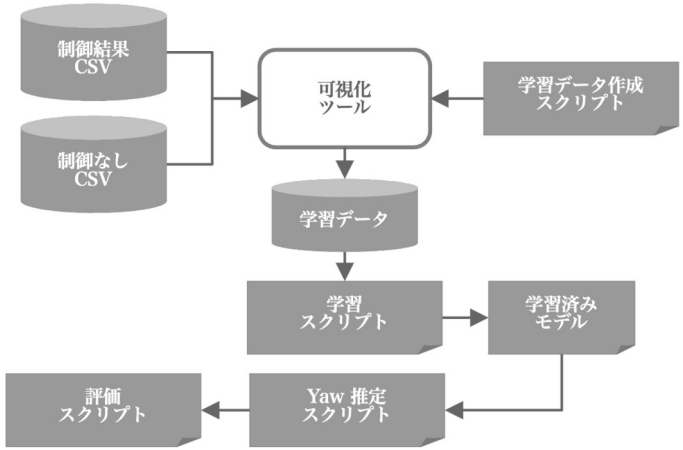
制御結果の例（目標角と実測Yaw）



- 旋回開始から約30秒で目標角付近に到達
- 目標角への接近に伴い推力を低下
- 到達後は低推力でYaw角を維持

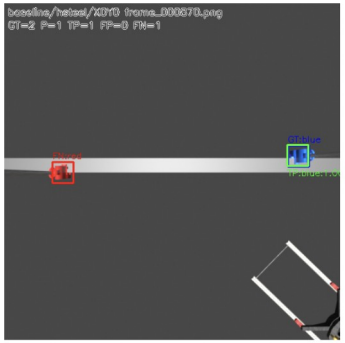
1.3 画像認識手法｜検出と幾何計算を分離

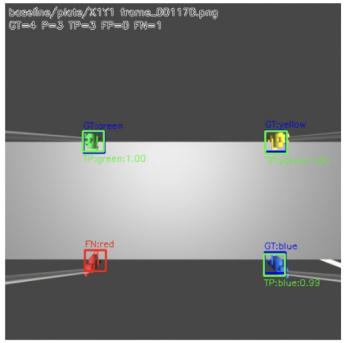
2D画像 → フック検出 → 位置関係からYaw角を算出 → 制御へ入力。

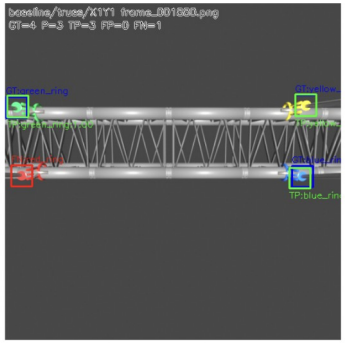


画像認識は特徴点を検出し、
Yaw角は幾何学的に推定する。

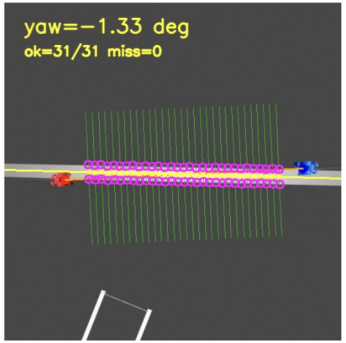
1. カメラ画像・フック検出

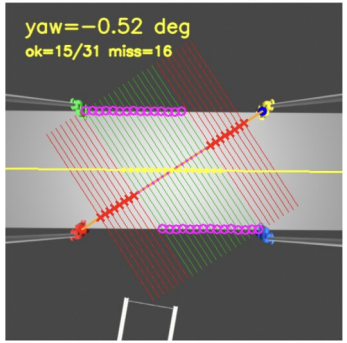


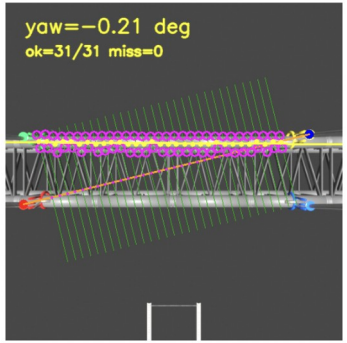




2. Yaw角推定の結果



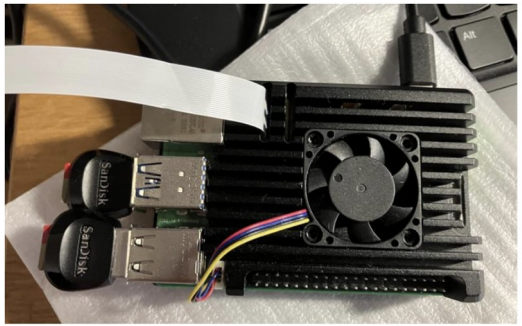


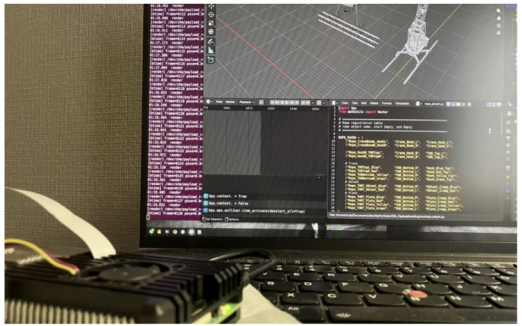


1.4 Raspberry Pi 5｜画像認識から閉ループYaw制御まで

Ubuntu側で物理シミュレーション、Raspberry Pi側で画像認識・Yaw推定・SAC制御を実行。

Raspberry Pi 5 による処理・接続





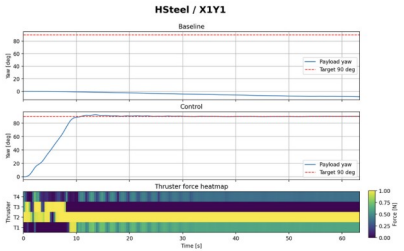
制御周期

3 Hz / 333 ms

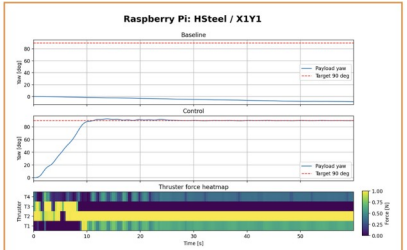
Pi側の処理時間

平均 68.3 ms
最大 124.0 ms

Ubuntu単体（左）と、Pi を用いた閉ループの制御（右）の結果



開発環境



Raspberry Pi 環境

2. Fine-tuning

3Dシミュレーションで生成した学習データを実写画像で補完し、6D姿勢推定の精度向上を図る。

公開資料

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

2.1 全体概要｜単眼RGB画像から6D Poseを推定

物体検出 → キーポイント推定 → PnPによる位置・姿勢計算





画像クロップ



2.2 Fine-tuning｜合成画像から実写環境へ

Blenderによる学習データ生成と、少量の実写画像による追加学習。

合成画像（Sim）



実写画像（Real）

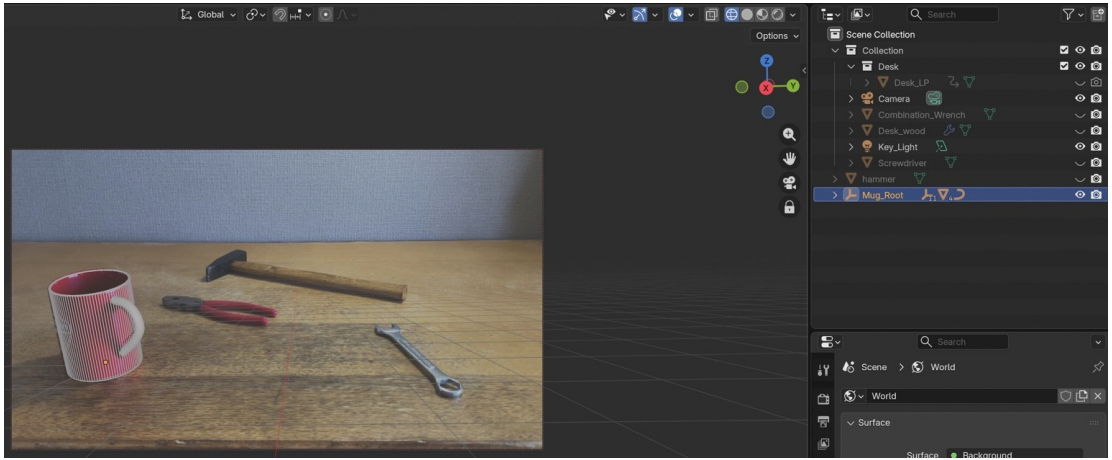


Simで事前学習（3,000枚）　→　3Dモデルを実写へフィット・教師点生成　→　Real 173枚で追加学習

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

2.3 教師データ生成｜3Dモデルを実写画像にフィット

手動での正確な指定が難しいキーポイントを、3Dモデルの投影によって一貫した基準で定義した。



1. 実写画像に3Dモデルを合わせる
2. 定義済み3Dキーポイントを投影
3. 実写画像の2D教師点を生成
4. Fine-tuning

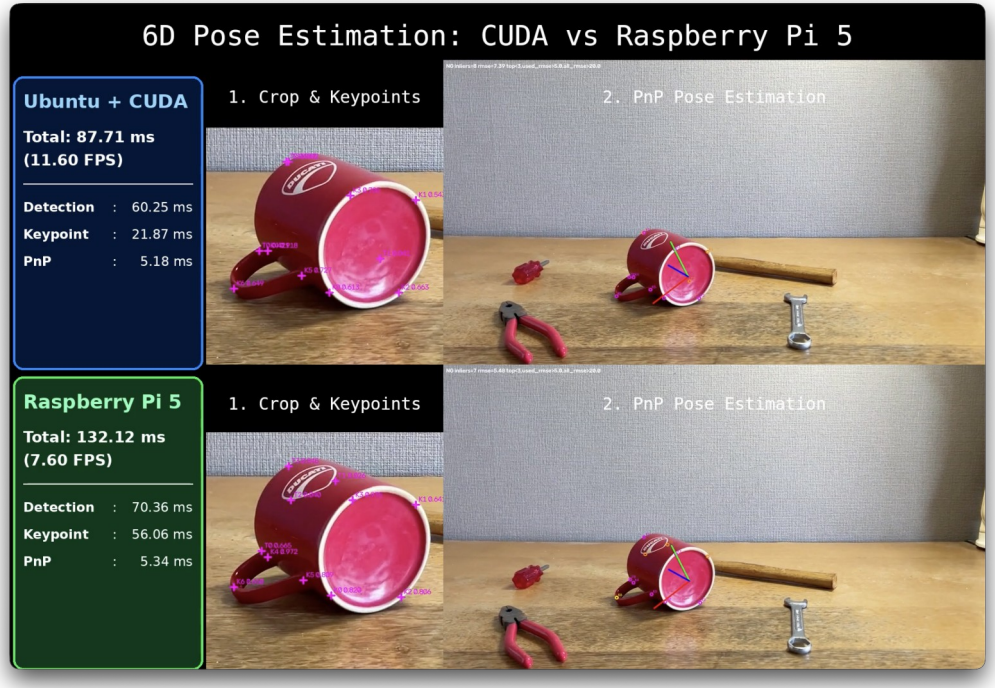
実写画像 + 3Dモデルの位置合わせ

※ 精度はフィッティング、カメラ条件、3D形状の整合性に依存。

手動でキーポイントを設定（前頁参照）と比較して、姿勢推定の精度が向上したことを確認。

YouTube動画
https://youtu.be/rC1AJGKOdmE

2.4 GPU vs Raspberry Pi 5



同一の画像認識モデルと姿勢推定アルゴリズムを使用し
GPU（CUDA）環境とRaspberry Pi 5で処理速度を比較した。

処理	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

Raspberry Pi 5でも7.60 FPSを達成。GPUとの性能差は主にキーポイント推定にありPnP処理ではほぼ同等の性能を確認した。

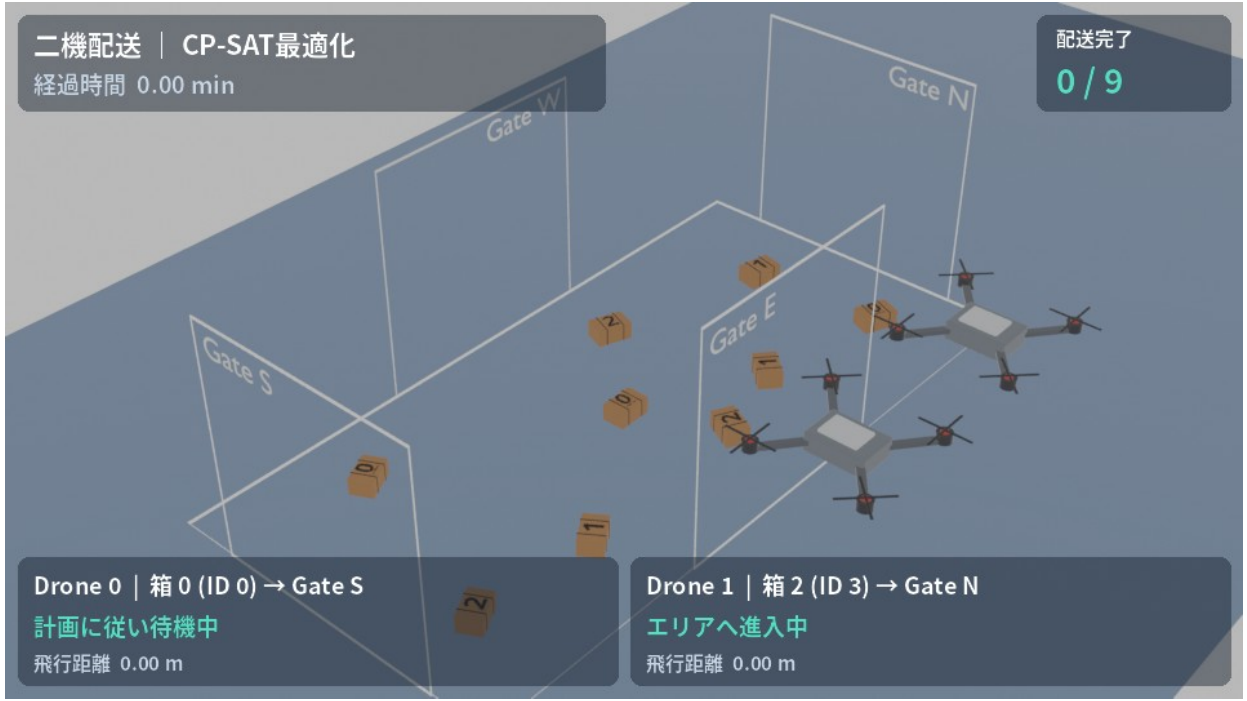
YouTube動画
https://youtu.be/zFwZ-Rls9bs

3. ドローン制御

画像認識による位置推定、PID制御、配送計画を統合し、複数機による自動配送を実現する。

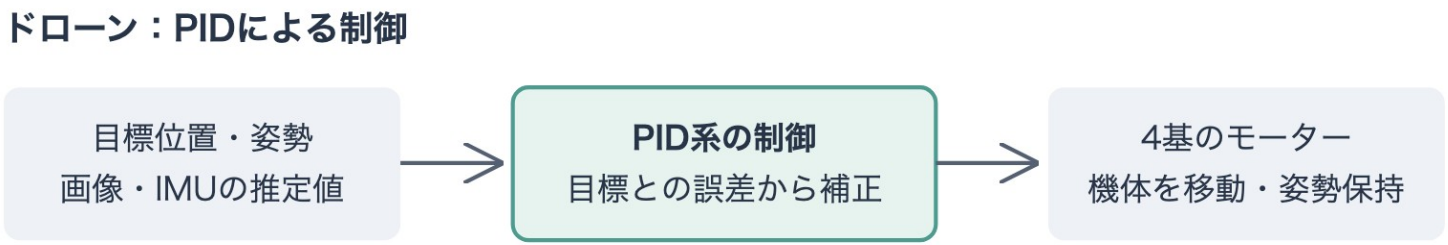
3.1 概要｜2機で9箱を自動配送

画像で箱に合わせ、PIDで飛行、CP-SATで配送を計画。



3.2 PID 制御

ドローン：目標との誤差からPIDでモーター推力を制御。



・状態推定：画像から箱との位置・向きを求め、IMUで機体の姿勢・角速度を推定。

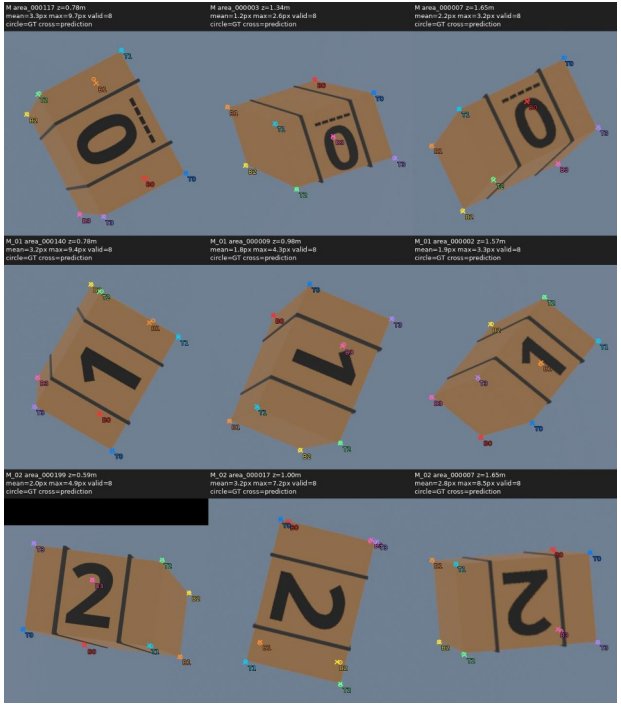
・PID制御：目標と位置・姿勢の誤差に応じて、各モーターの推力を調整。

・周期の分離：画像による補正の間も、IMUによる状態推定と姿勢制御を継続。

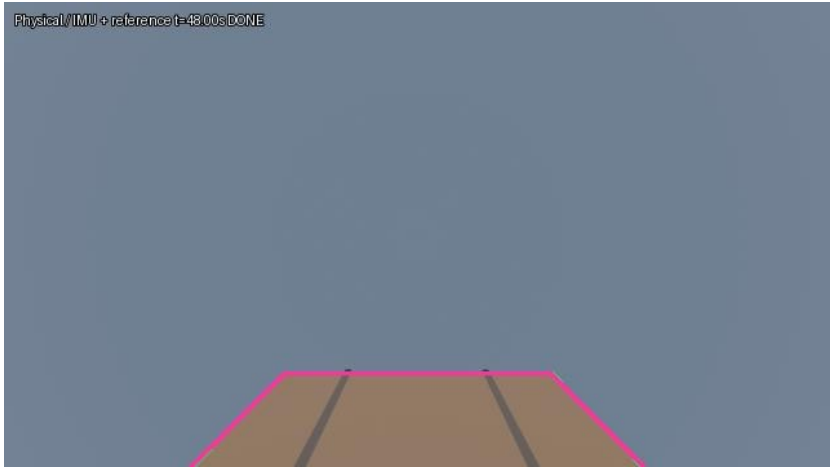
注：シミュレーションの基盤に PyBullet/gym-pybullet-drones を使用。

3.3 画像処理｜PnP＋画像マッチング

接近：特徴点＋PnPで箱との位置・向きを推定。



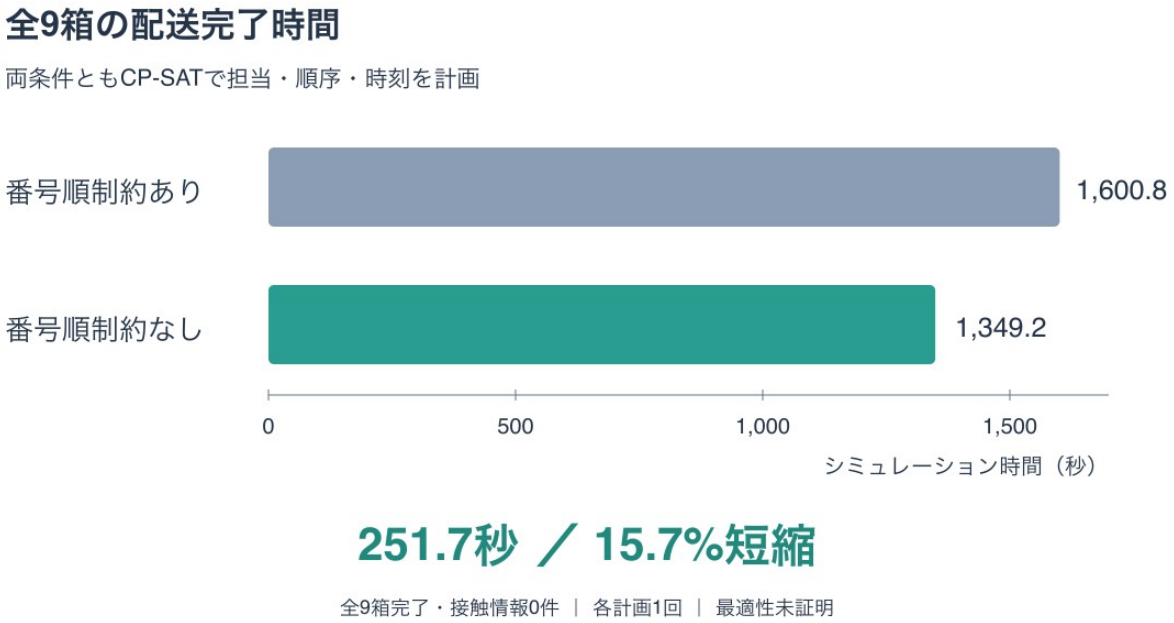
把持直前：参照画像とのマッチングで位置合わせ。



YouTube動画
https://youtu.be/Tr1Pd5xofdM

3.4 配送計画｜CP-SATで担当・順序・時刻を決定

2機の担当・配送順・待機を事前計画。
番号順制約を外すと、今回の配送時間は15.7%短縮。



YouTube動画
https://youtu.be/XLUYuL7K9-U