AAE4002 Capstone Project
Final Presentation

AAE035
Vision-base of a Quadrotor UAV Navigation

LO Li-yu (17086854D)
YIU Chi Hao (17003167D)
Tang Yu (17086039D)

Supervisor: Prof. Chih-Yung WEN

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

01. **Introduction**
- Background

02. **Project Overview**
- Objectives/Contributions
- System Architecture

03. **Methodology**
- Learning-based Object Detection
- Pose Estimation
- Autonomous Object Searching UAV system
- Autonomous Object Tracking UAV system

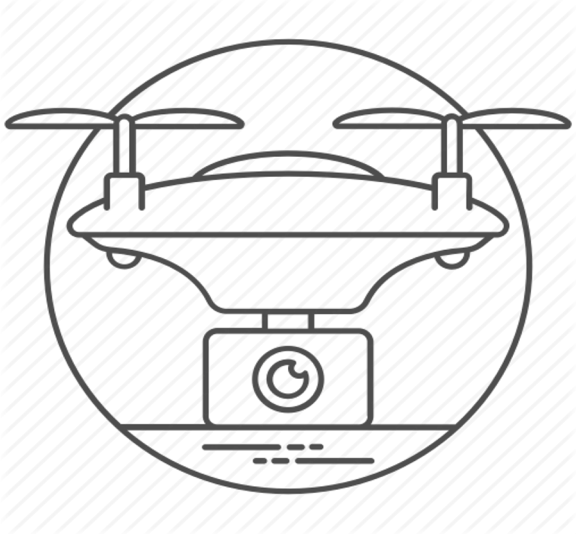
04. **Results and Discussions**
- Results of experiments

05. **Conclusion**
- Achievements and Future recommendations

06. **Question & Answer**

01
Introduction

Vision-based UAV



Characteristics

- Visual sensors (camera)
- Low weight and power consumption
- Distinct visualization and perception

The problems of conventional applications

search & rescue (SAR)

- time-consuming
- labour- intensive
- hazardous

reconnaissance

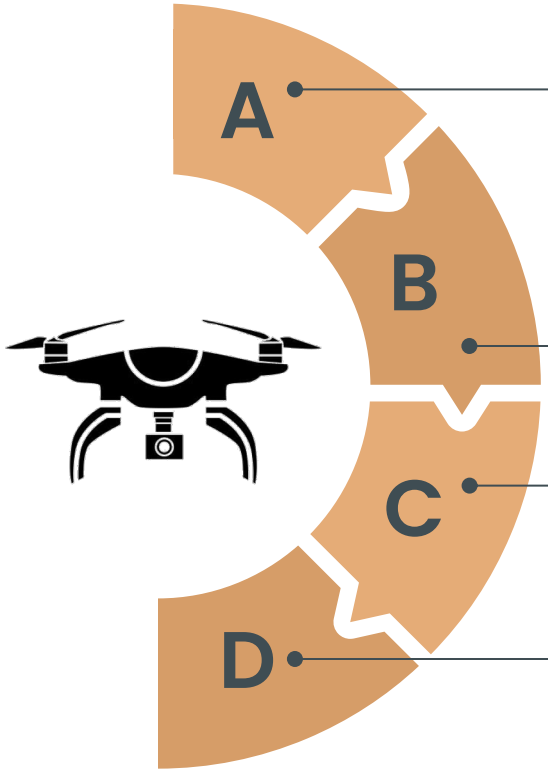
- tedious
- inefficient
- energy-consuming

What if we employ vision-based UAV to assist them?

- agile maneuverability
- visual remote-data in real time
- Cost-effective


We can use UAV to help searching target objects

Major Considerations



Autonomous operation

- Costly and arduous if conducting in manual teleoperation
- Inefficient with human intervention

Robust perception solution

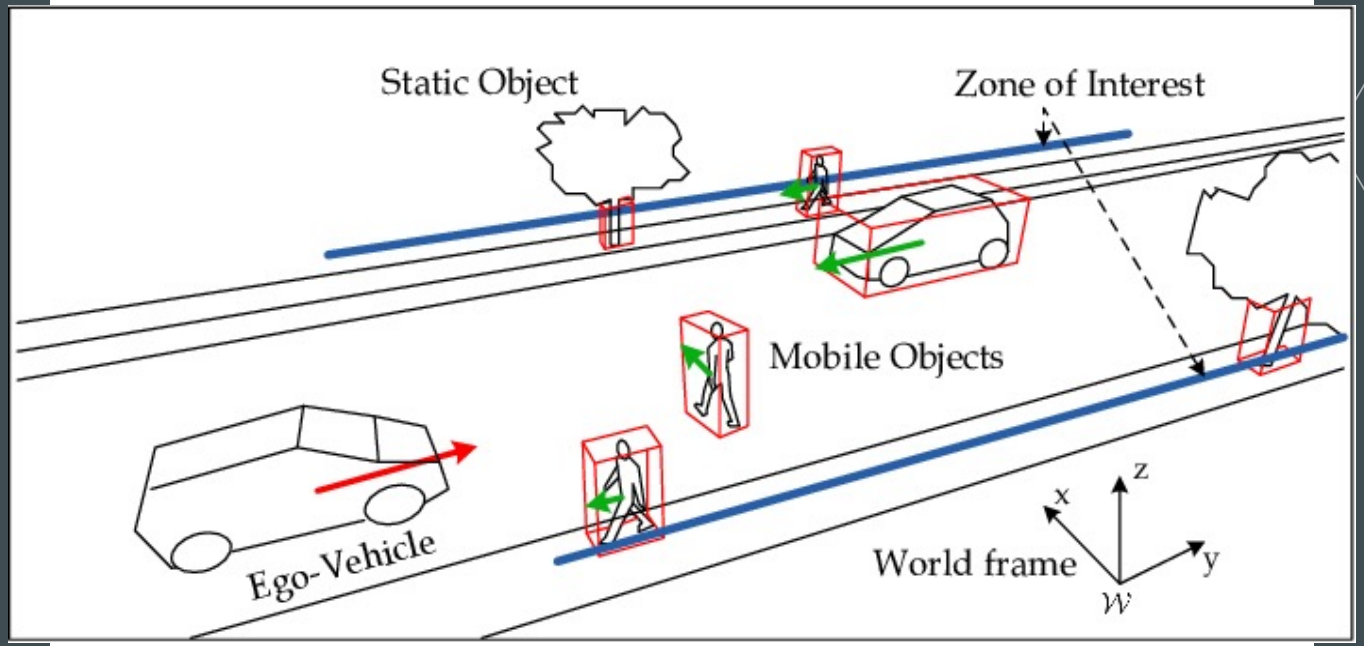
- To accurately detect target objects

Path-planning capability

- Collision-free and safe flight during searching process

Low payload and battery power consumption

WHAT IF the target object (victim) is moving? dynamic and mobile objects?



UAV should be able to track and follow target objects

Both searching and tracking actions are critical as preliminary procedures for a typical SAR or reconnaissance mission.

02

Project Overview

Learning-based navigation system for quadrotor UAV

What We were Working On

Autonomous Object searching

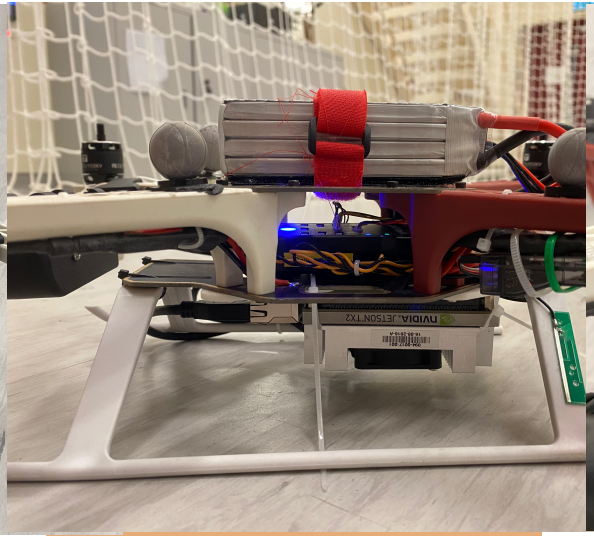
To search and locate the position of static target objects

Autonomous Object tracking

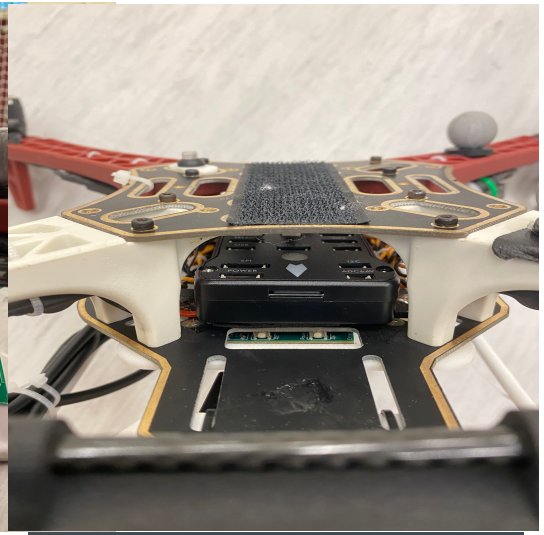
To track and follow the movement of mobile target objects

MAJOR Hardware platform

Intel RealSense D435i stereo camera

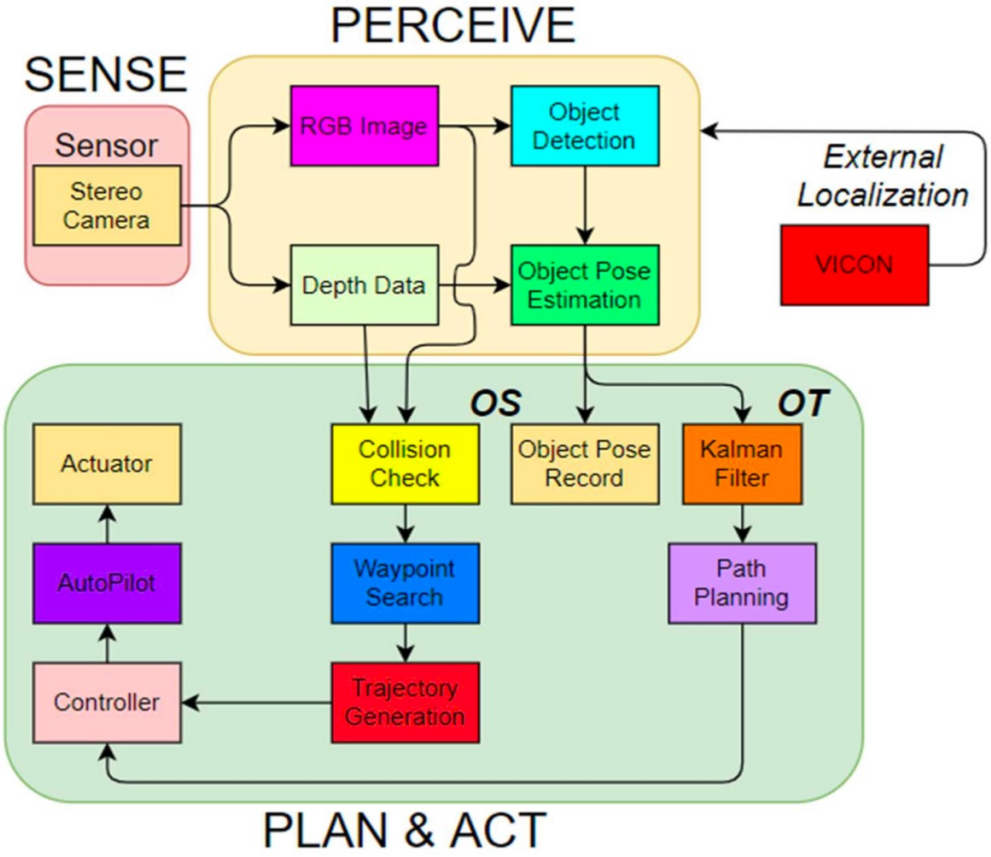


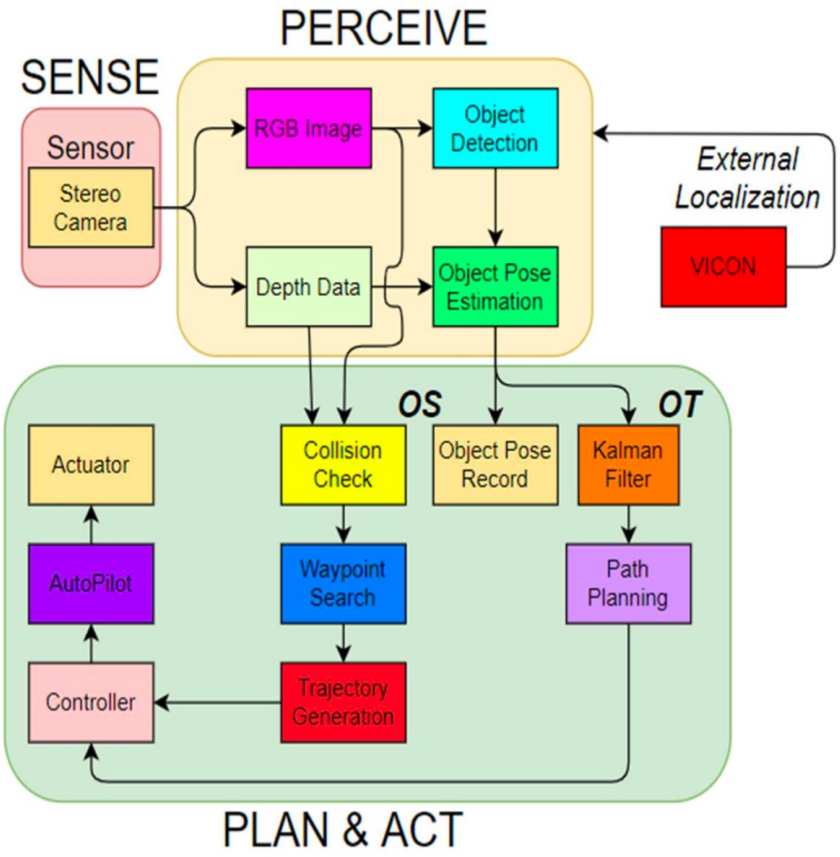
Jetson TX2 onboard computer



PX4 controller

Software Architecture of our designed UAV system





Object searching

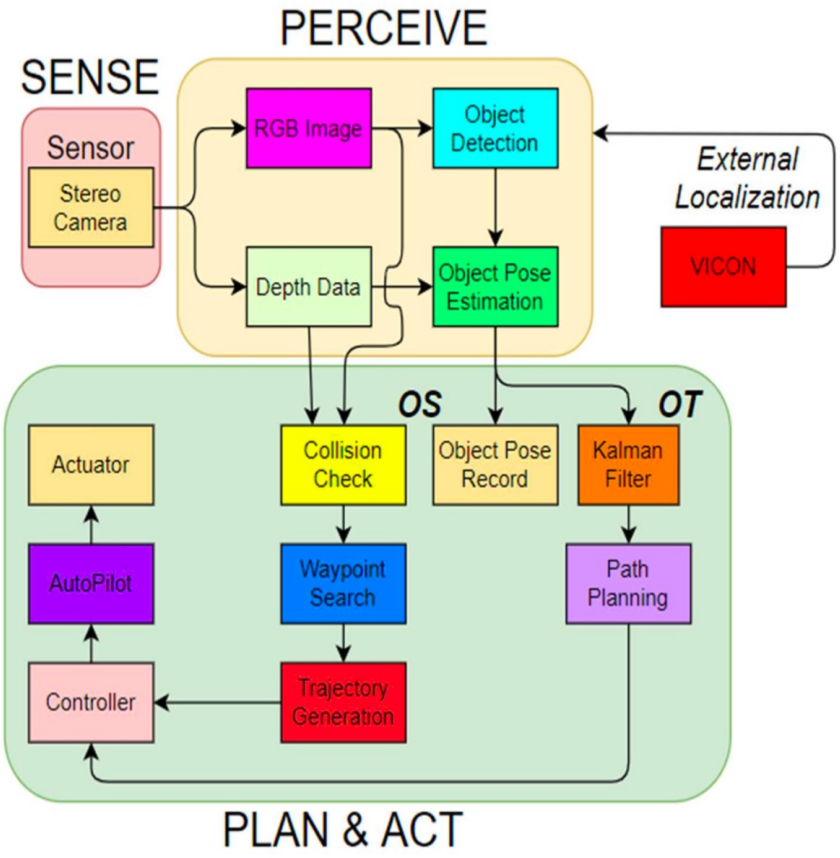
Consists of 2 Modules:

Perception:

- Learning-based object detection (YOLOv4-Tiny)
- 3D object pose estimation

UAV Path Planning & Act

- Front-end:
  - check collision
  - waypoint search via image-processing techniques (Edge Detection Planner)
- Back-end:
  - trajectory optimization (minimum jerk)



Object tracking

UAV front-end (perceive)

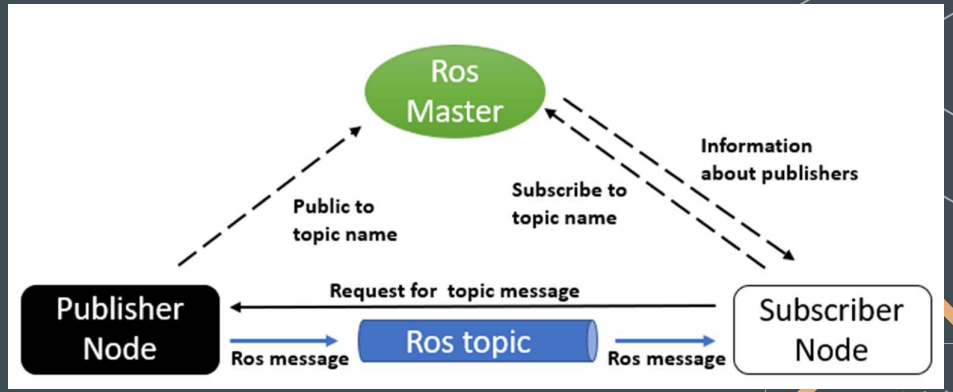
- Learning-based object detection (YOLOv4-Tiny)
- 3D object pose estimation

UAV back-end (plan & act)

- Kalman filtering (motion estimation and prediction)
- path planning method with learning-based 3D object state estimation

ROS

Robot Operating System



System framework was supported and done through ROS platform

03

Methodology

- Learning-based Object Detection
- Pose Estimation
- Autonomous Object Searching UAV system
- Autonomous Object Tracking UAV system

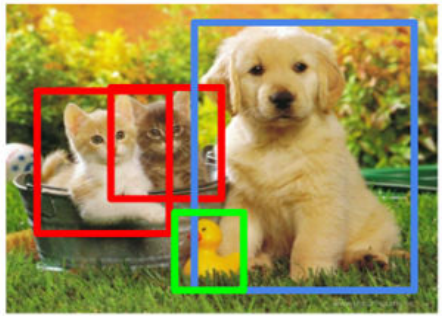
Learning-based Object Detection

Object Detection
(object localization and classification)

Classification

CAT

Object Detection

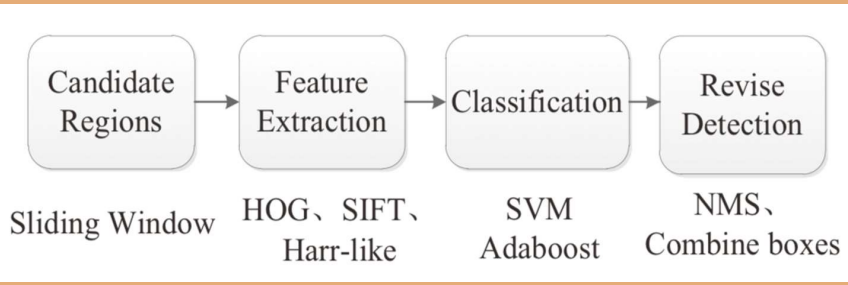


CAT, DOG, DUCK

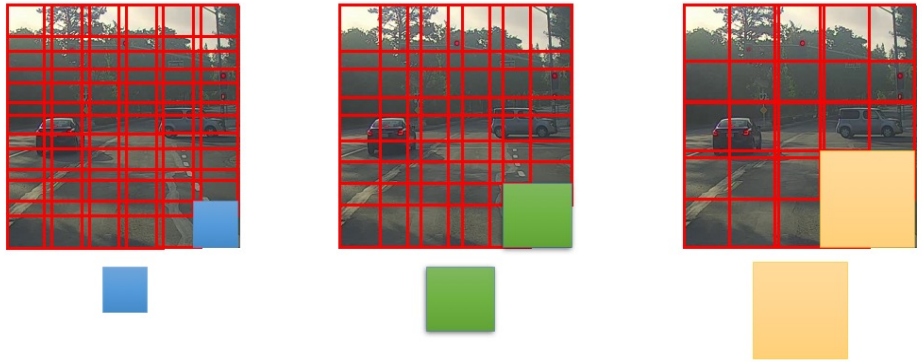
- finding the position of objects (object localization)
- classifying the corresponding categories of detected objects (object classification)

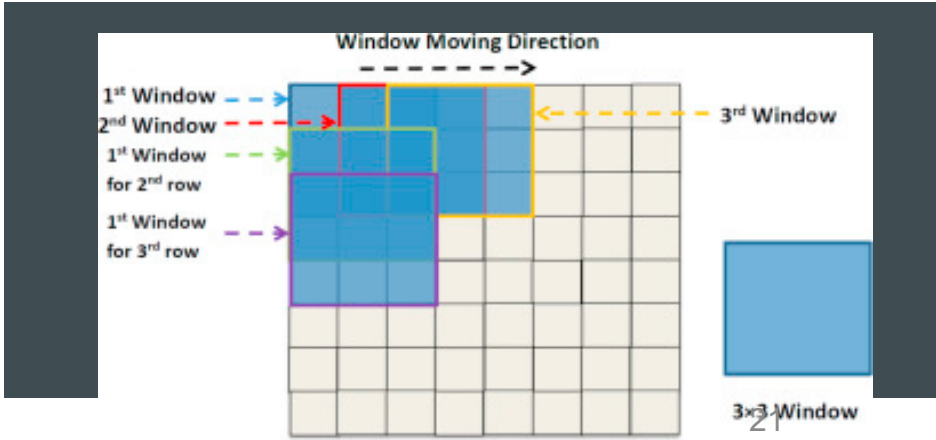
2-D Object Detection on real-time aerial images

Traditional methods



- low capacity of data model
- low robustness for various geometric or photometric changes
- excessive computation cost
- high inaccuracy
- huge semantic gap





2-D Object Detection on real-time aerial images

Deep-learning based detector

Two-stage detector
(region proposal-based detectors)

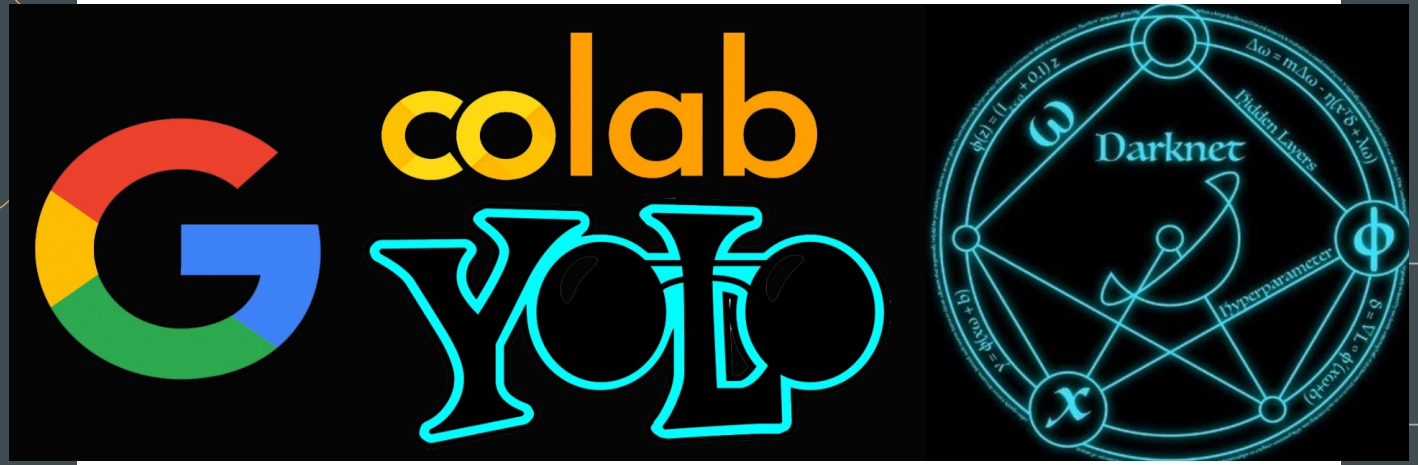
- Convolutional Neural Networks (CNNs) series
  - Region with CNN (R-CNN)
    - Fast R-CNN
    - Faster RCNN
      - R-FCN
    - Mask R-CNN

- Slow detection speed
- Relatively high accuracy

One-stage detector
(regression-based detectors)

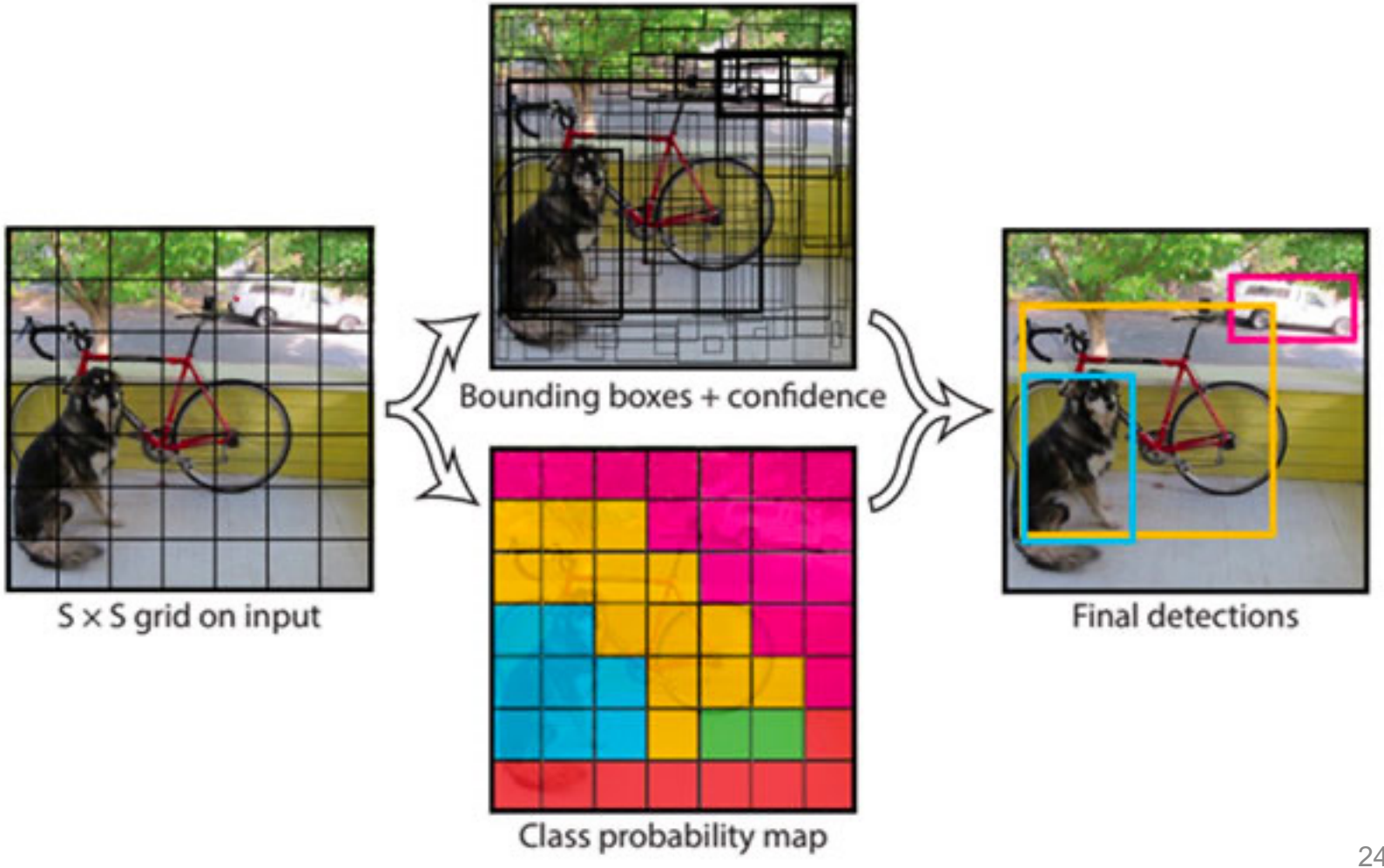
- 'You Only Look Once' (YOLO) series
  - YOLOv2,v3,v4
  - YOLO-Tiny
  - RetinaNet
- Single Shot MultiBox Detector (SSD)

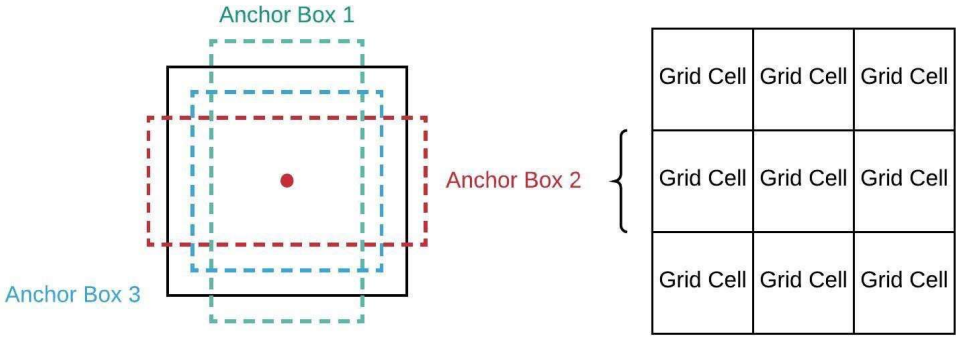
- Fast detection speed
- Relatively low accuracy

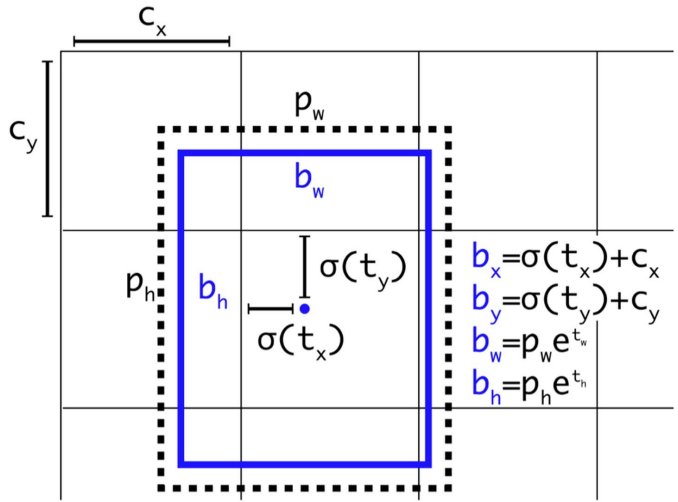


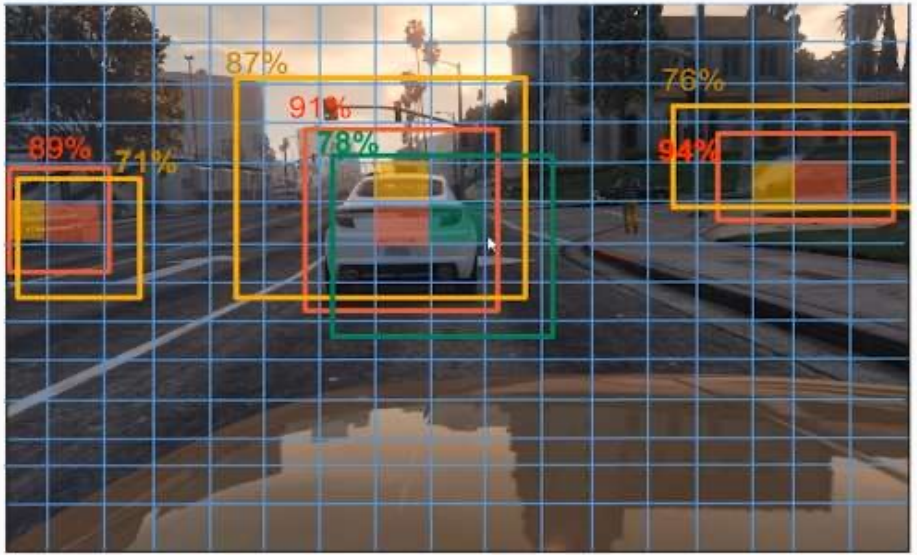
YOLOv4-Tiny

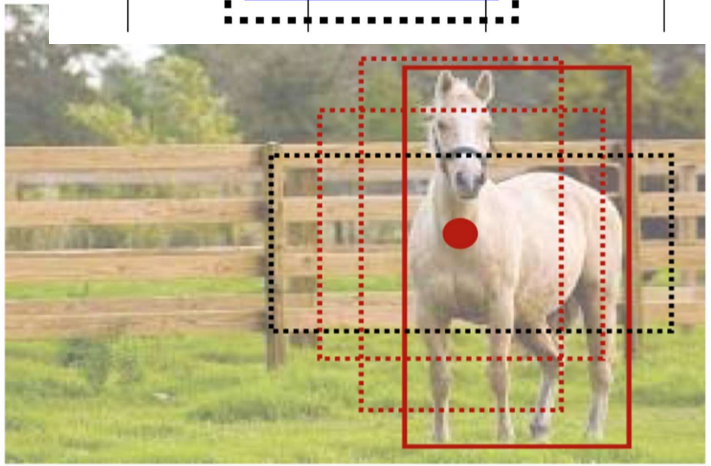
- Real-time detection speed with high frame per second (FPS)
- optimal precision
- low computational cost on embedded UAV platform with limited resources


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections





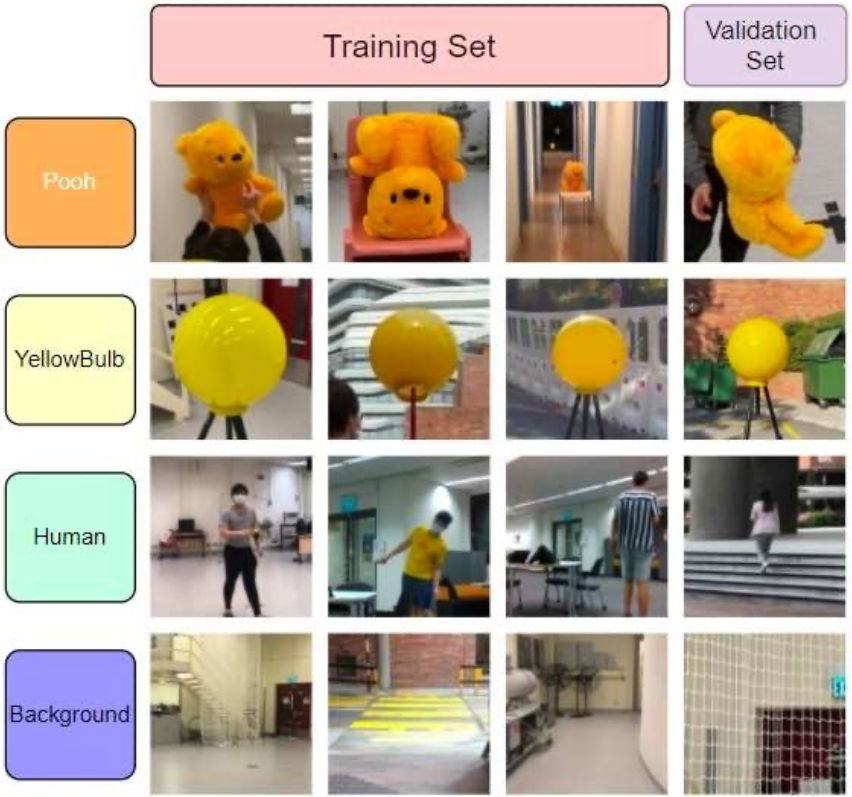




GT box Positive Anchor Negative Anchor

How to implement YOLOv4-Tiny (Supervised deep-learning algorithm) in our UAV system?

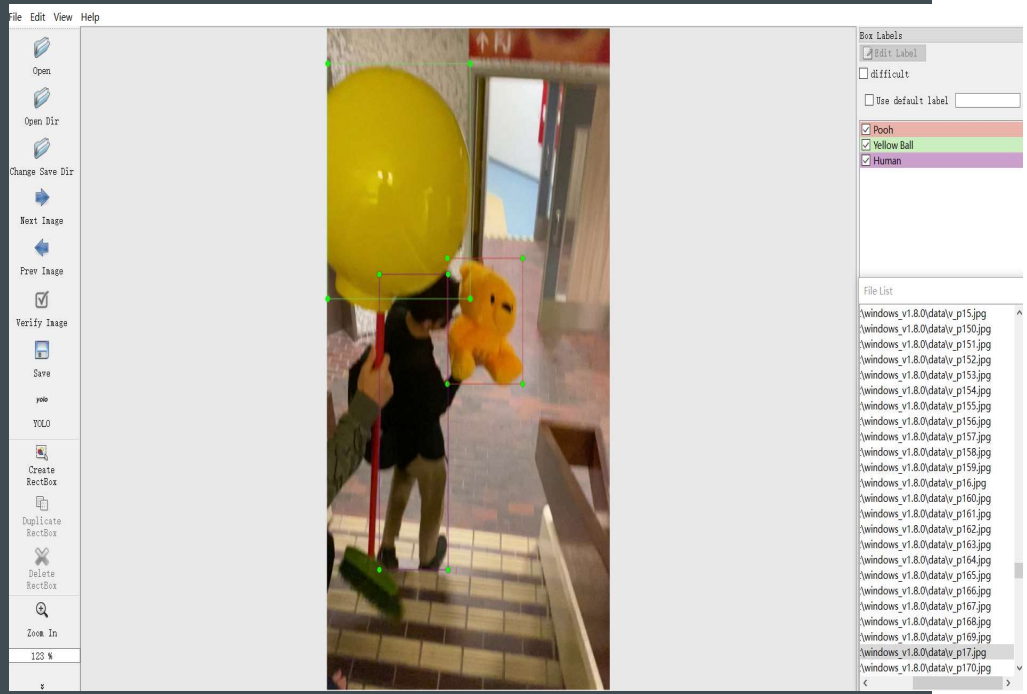
Customized Dataset Establishment



in-house dataset (13,500 images)

- 2000 training images plus 500 validation images (4:1 ratio) for each class
- 6000 background images with no target object
- random and dissimilar illumination conditions, scales, view of angles, aspect ratios, resolutions, backgrounds
- multiple objects (i.e., pooh, yellow bulb ball, human) in a single frame
- aerial scale images accorded with the captured view of flying UAV

Dataset labelling

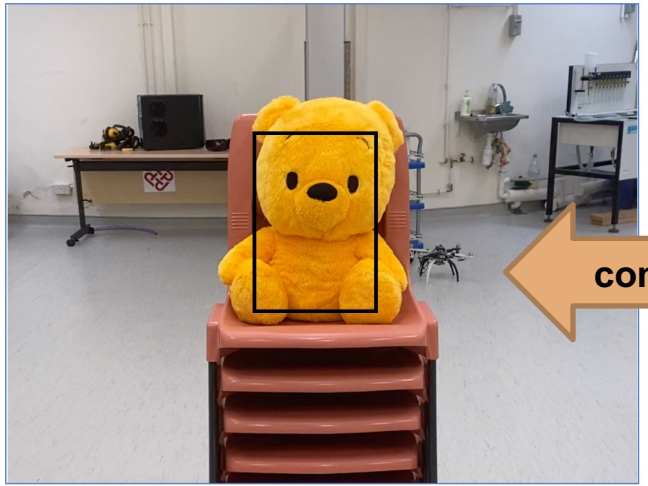


- Manual labelling
- provides ground truth bounding box and the corresponding class names (i.e., Pooh, Yellow Bulb, Human)

Model training

Input image in dataset

Predicted output during training



Actual output (ground truth box)

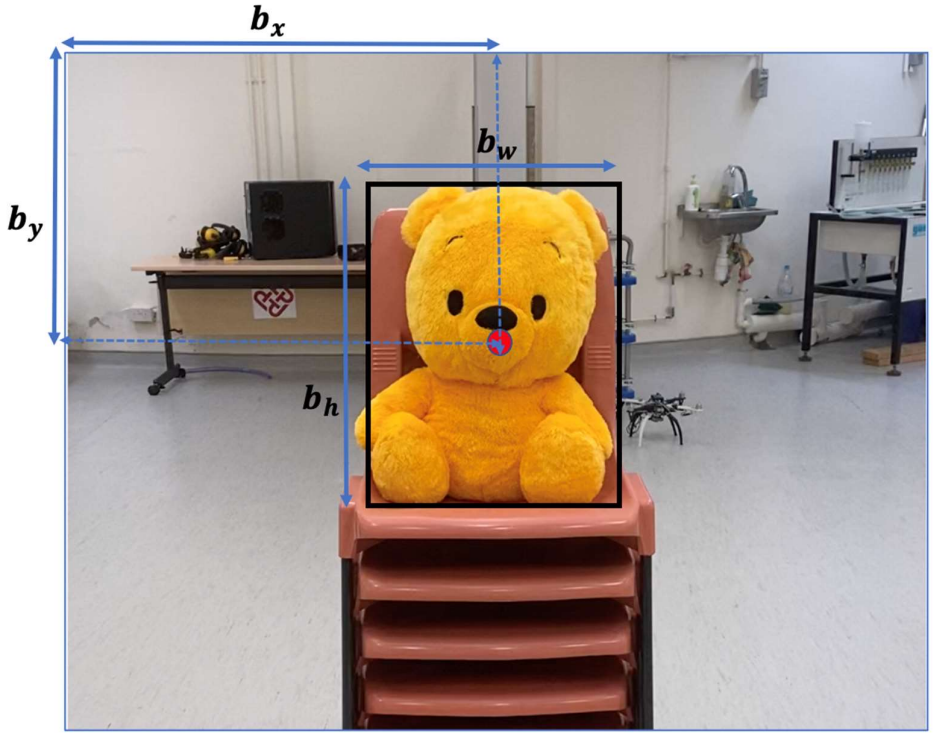
compare

- *Aims*

- Increases the mean Average Precision (mAP), the detection accuracy of target object in images
- Decreases the Loss, the discrepancies between model predictions and the ground truths
- Different input resolutions of input image affects the performance (FPS)!

A trained model with acceptable mAP (>75%) and sufficiently low Loss

Real-time : 2D bounding boxes on detected target objects

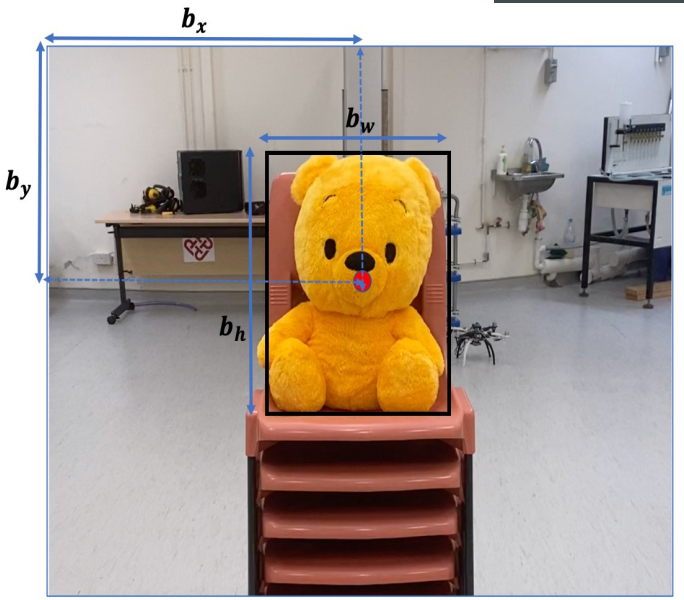


- b_x : center coordinates x
- b_y : center coordinates y
- b_w : width
- b_h : height
- coordinates of the bounding box which contained the target object as box_{yolo}

Pose Estimation

Pose Estimation

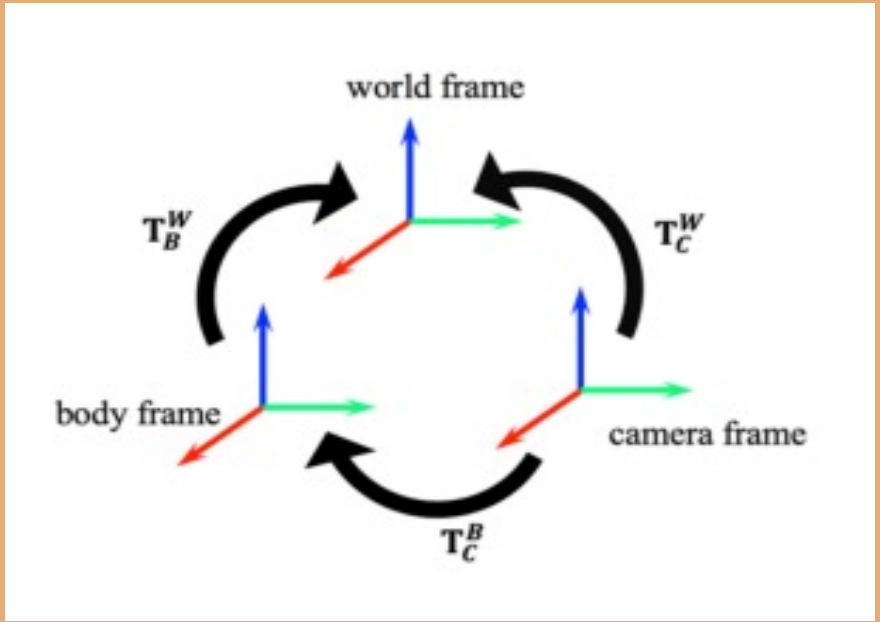
- Data from RGB-D camera
- Each pixel contains depth info
- Object bounding boxes
- Inner box box_{depth}
- Depth info of the object



- Camera -> Body -> World

$$\begin{bmatrix} X_i^W \\ 1 \end{bmatrix} = T_B^W T_C^B \begin{bmatrix} X_i^C \\ 1 \end{bmatrix}$$

Frame Transformation



IIR Filter

- For **object Searching system**
- Infinite Impulse Response Filter

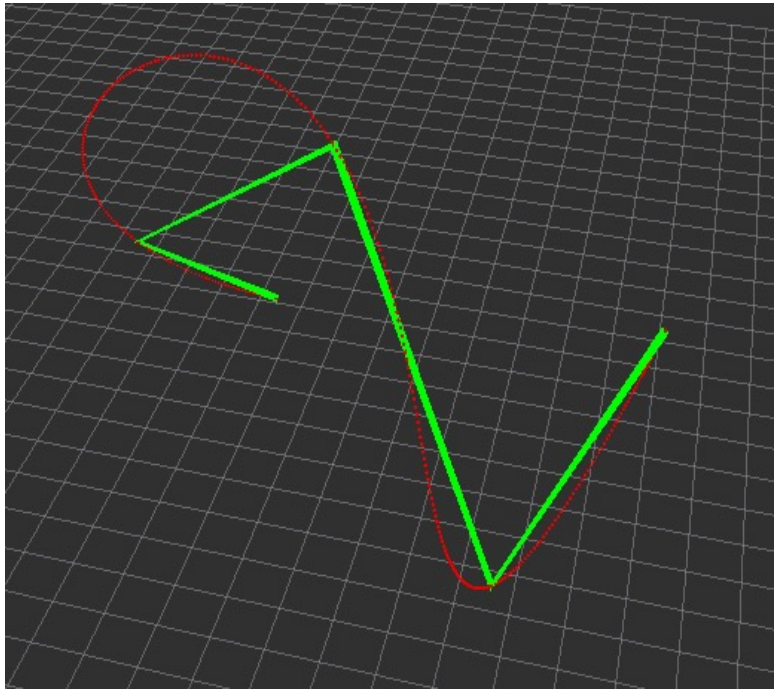
$$X_{i_{avg}}^{W} = \alpha X_{i}^{W} + (1 - \alpha) X_{i-1_{avg}}^{W}$$

- Recursive computations
- $\alpha = 0.9$ (in our case)

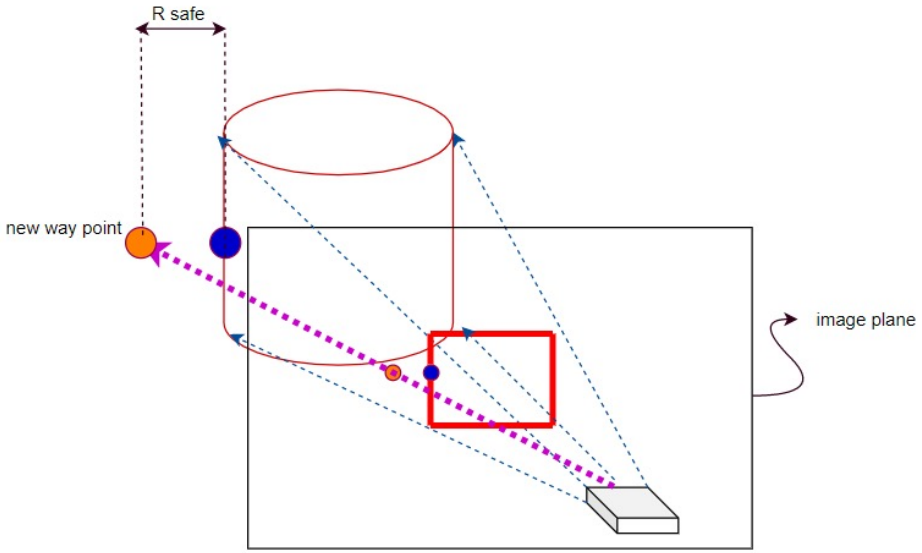
Object Searching System

Local Planner

- We Need To Move without Collision
- A Planner
  - Front-End: Path Finding
  - Back-End: Trajectory Optimization
- We proposed:
  - Edge Detection Planner
  - For Column-wise obstacles

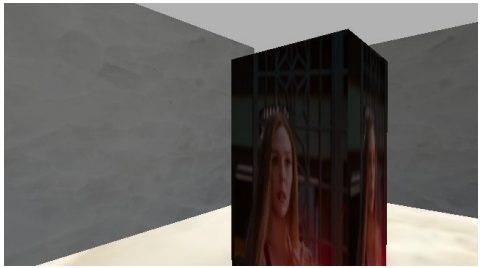


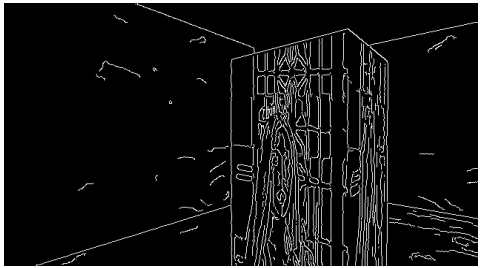
Edge Detection Planner



Edge Detection Planner –Front End

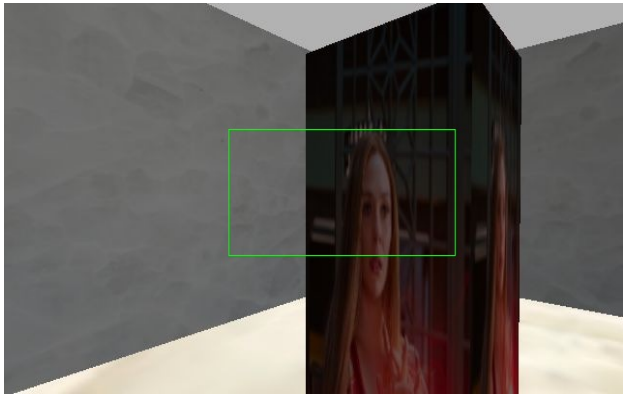
- Waypoint Search
- Reference point on an obstacle
- We used
  - Canny Edge Detection
  - For waypoint search

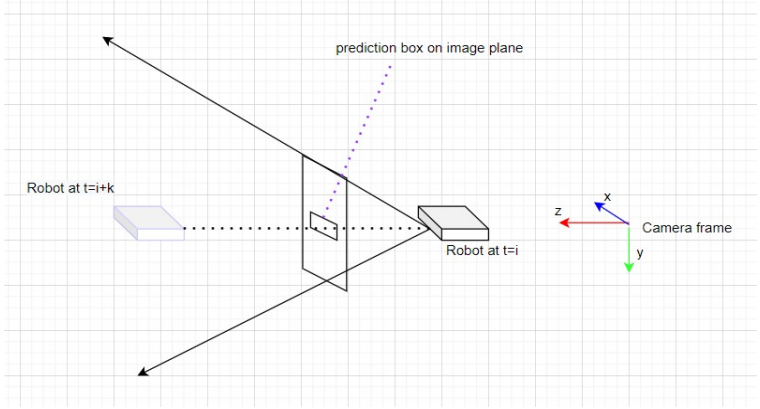


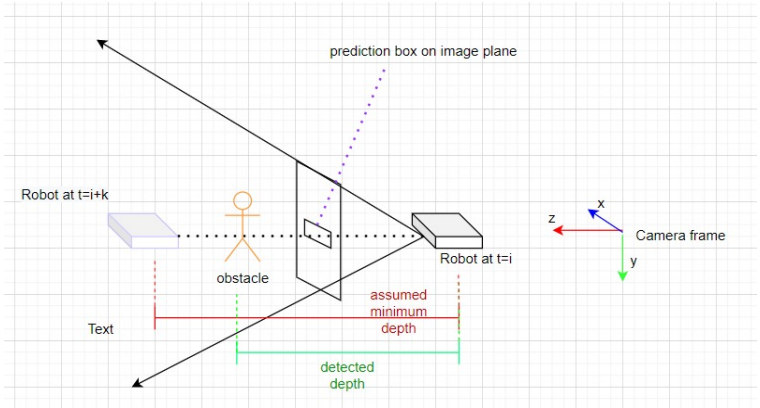


Edge Detection Planner –Front End

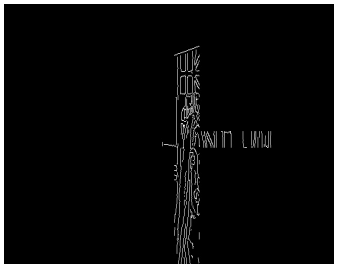
- Conduct Collision check



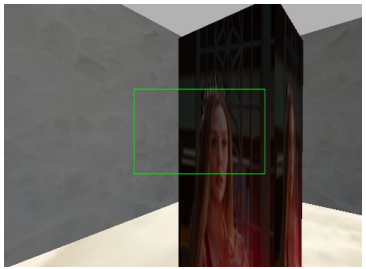




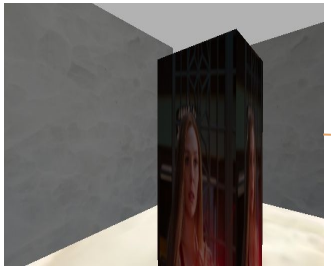
Get reference point

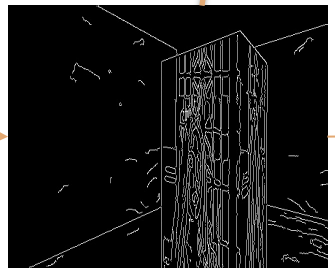


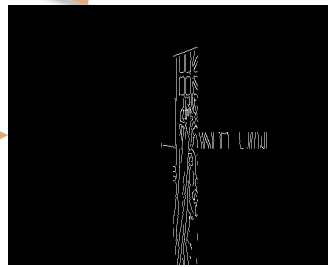
Filter and get the pixel with the shortest distance w/ the “UAV future box”

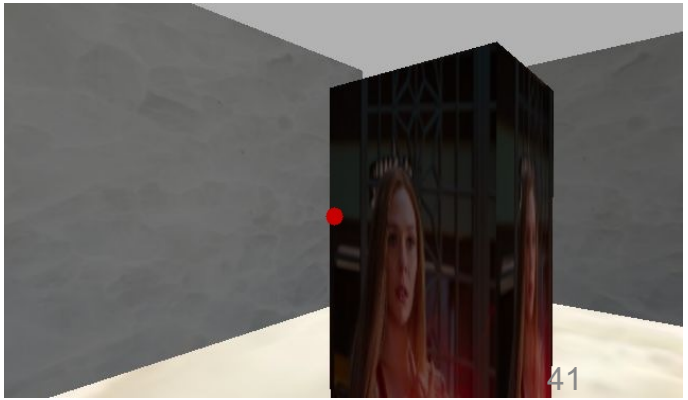


pixels





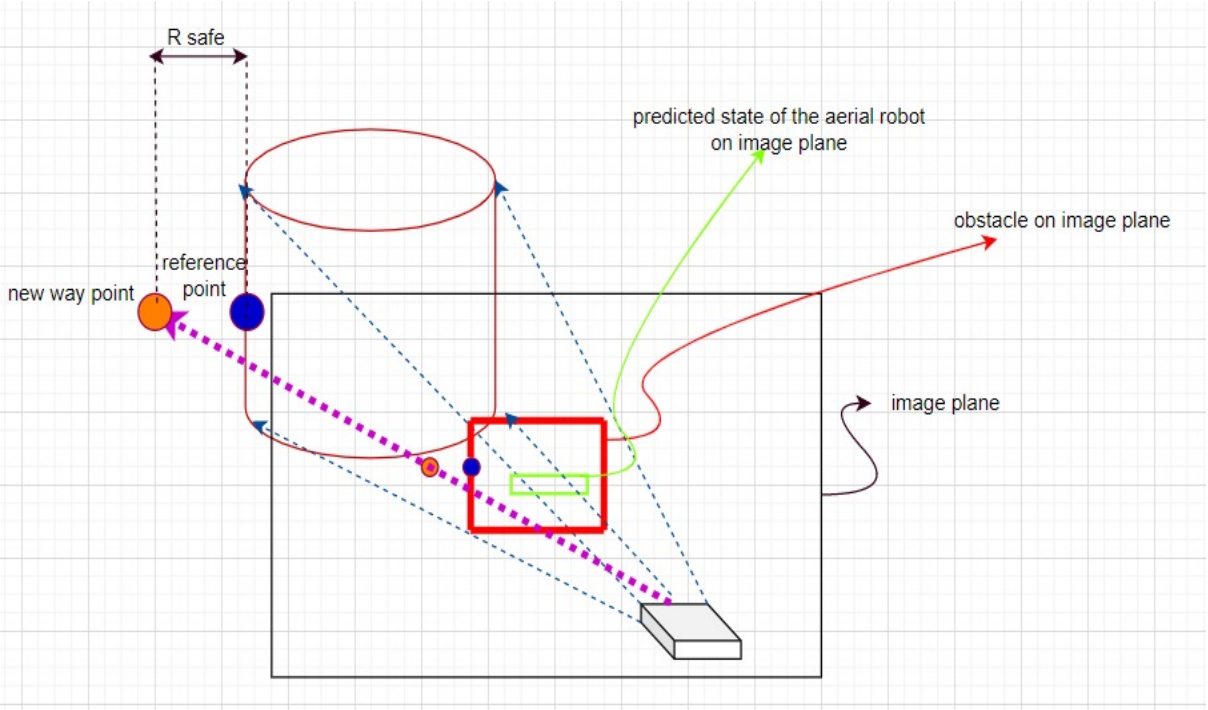




Edge Detection Planner –Front End

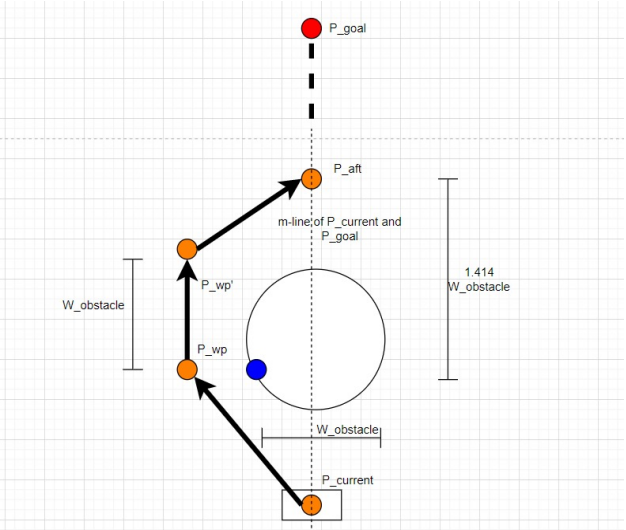
- Determine which way should go
- Calculate the relative position with the obstacle

Edge Detection Planner –Front End –waypoint search

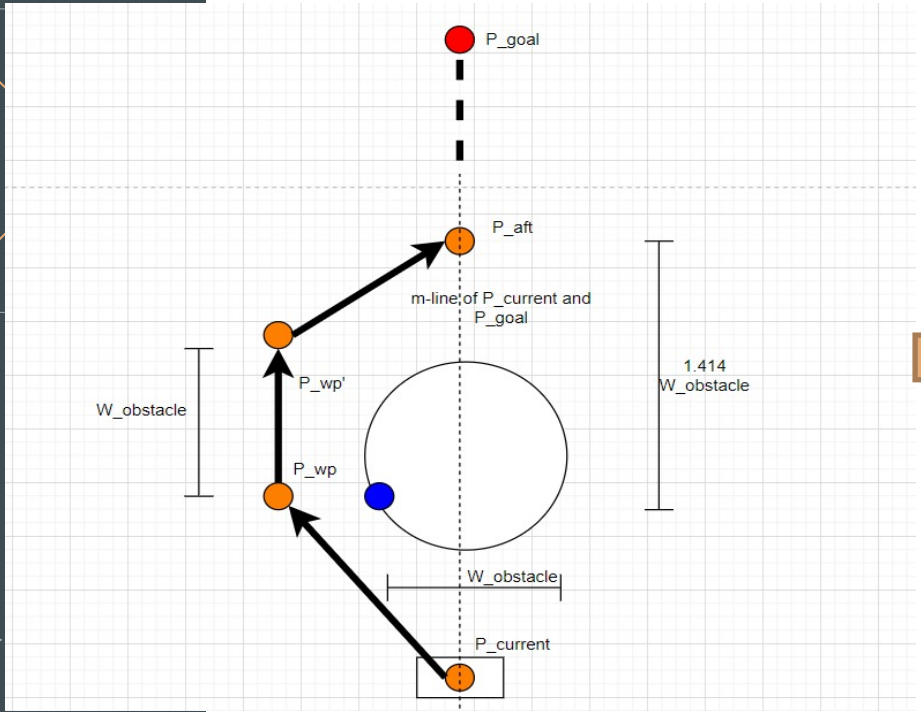


Edge Detection Planner –Front End

- Safety guarantee
- Assume all obstacle are column-wise
- Get 4 local waypoints



Edge Detection Planner –Back End



Trajectory optimization

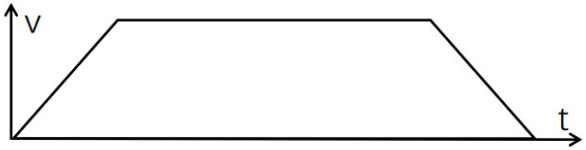
Optimization

- Smooth movement, reduce control effort
- Optimized as piece-wise polynomial
- Minimum Snap Trajectory Generation (4th derivative of position)
- We used Minimum Jerk (3rd derivative of position)

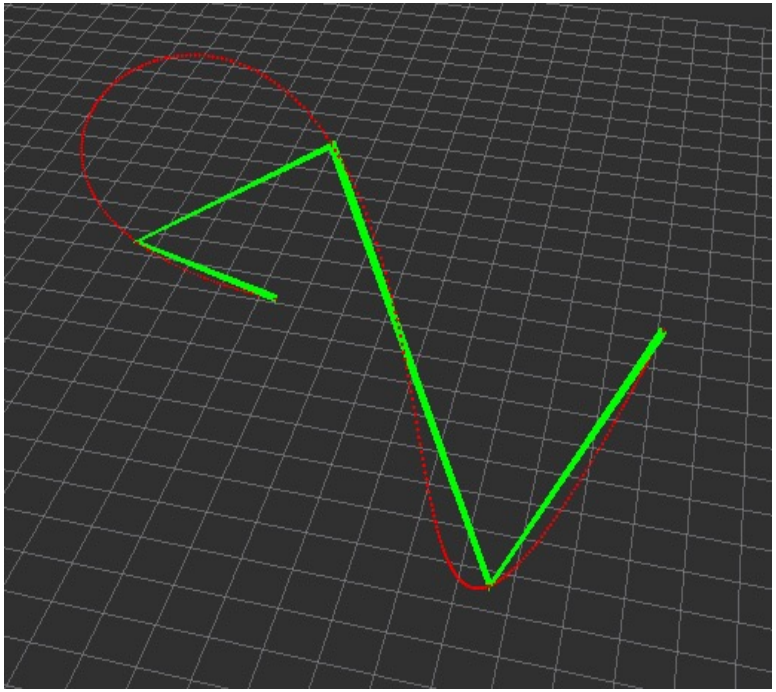
$$min \int_0^T (P^{(3)}(t))^2 dt$$

$$\min J = P^T Q P \, , while \; M_j P_j = d_j$$

- Time allocation - trapezoidal velocity time profile
- Solve Quadratic Programming, get P (polynomial coefficients vector)



Optimization Result

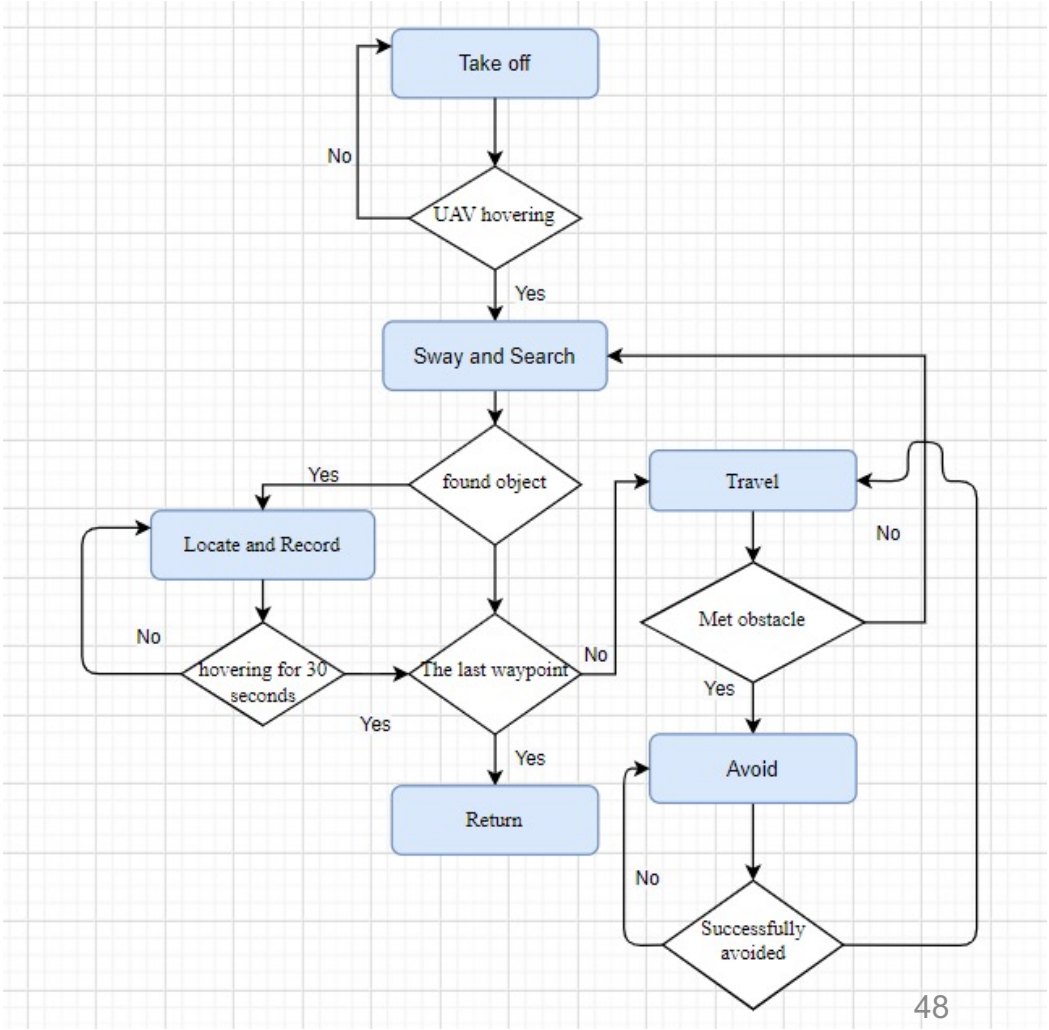


State Machine

We now have:
1. YOLO-Tiny
2. EDP

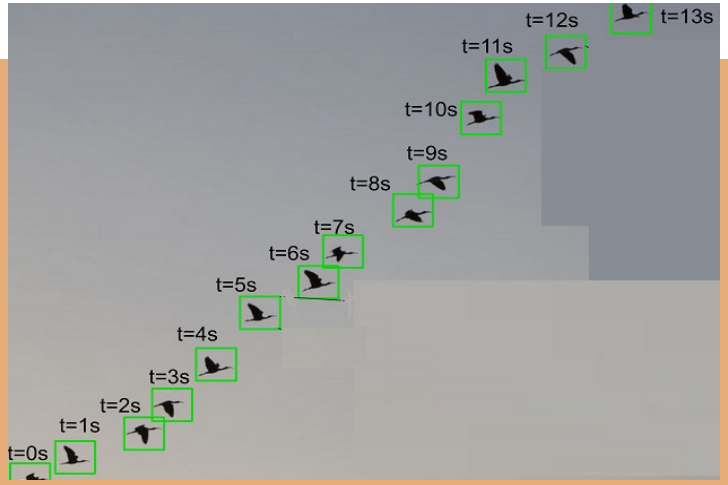
- Predefined waypoints:
- wp1, wp2, wp3, wp4, …, wpk

States:
1. Take-off
2. Sway-Search
3. Locate and Record
4. Travel
5. Avoid
6. Return



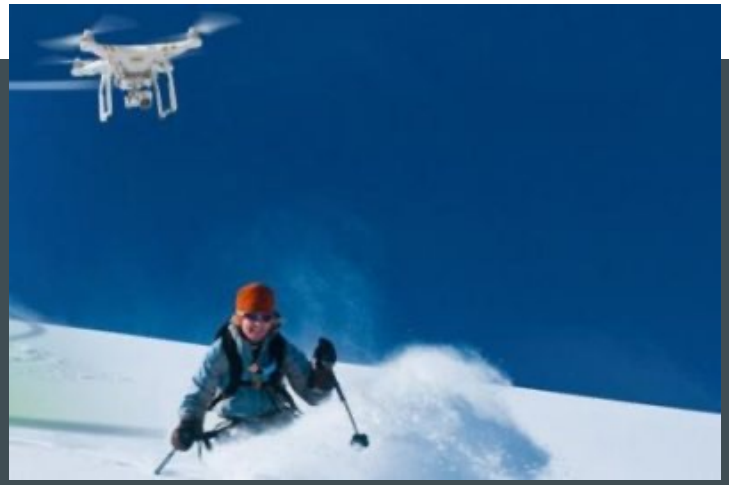
Object Tracking System

Single-Object Track & Follow



Object Tracking

- Object detection algo
- Motion prediction methods
- Data association problem solution

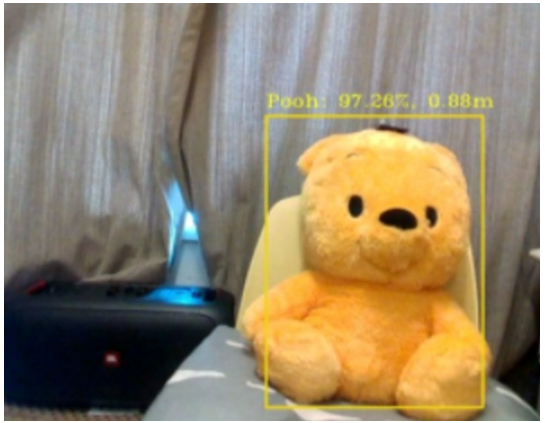


Quadrotor Action

- Object following action

Object Tracking

- Detection algo- YOLOv4-Tiny
- Motion Prediction –Kalman Filter
- Association problem –as we focus on single object, so it is not used.



Kalman Filter

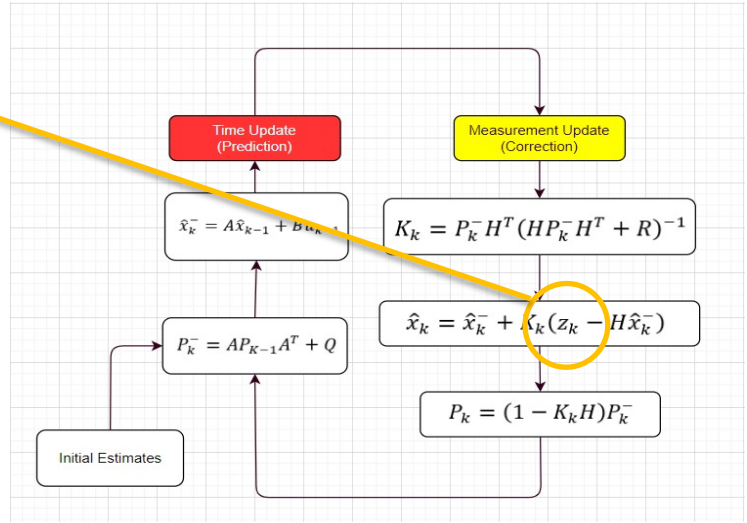
- Object could be occluded/not detected whilst performing tracking and following
- Discrete Kalman filter is applied in our case
- Consider a state (object) as a weighted average of 2 things:
  - a new measurement and a past estimation
- Make educated estimations

$$\hat{X}_k = K_k \cdot Z_k + (1 - K_k) \cdot \hat{X}_{k-1}$$

Kalman Filter

- Recursive process
- Time update and measurement update (Prediction and correction)

Update Zk by YOLO-Tiny



Kalman Filter

- State is sets as (relative state parameters):

$$\hat{x}_k = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{6\times 1}$$

- Zk, measurement:
  - YOLO-Tiny detection data is used when detected
  - Else -> Kalman Filter's past estimation

UAV action

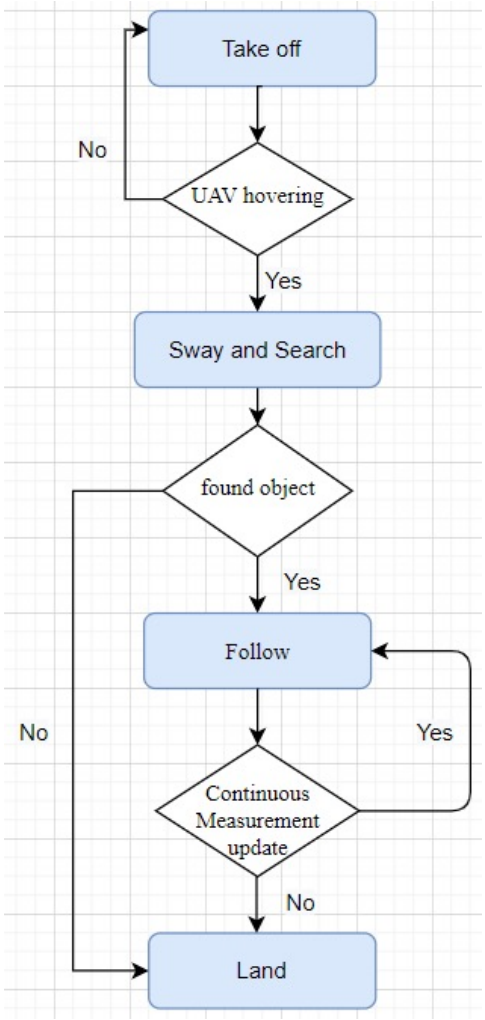
- Optical axis should be focusing on the object (within the region Sc): yaw angle change, UP,DN
- Safety distance should be maintained (stay within D safe to D max): move FO, BA.

- UAV will change yaw angle, move UP, DN, FO, BA

State Machine

We now have:
1. YOLO-Tiny + KF tracking algo
2. UAV action

States:
1. Take-off
2. Sway-Search
3. Follow
4. Land: will happen will object is missed out for a certain of duration



04

Results and Discussions

Results and Discussions

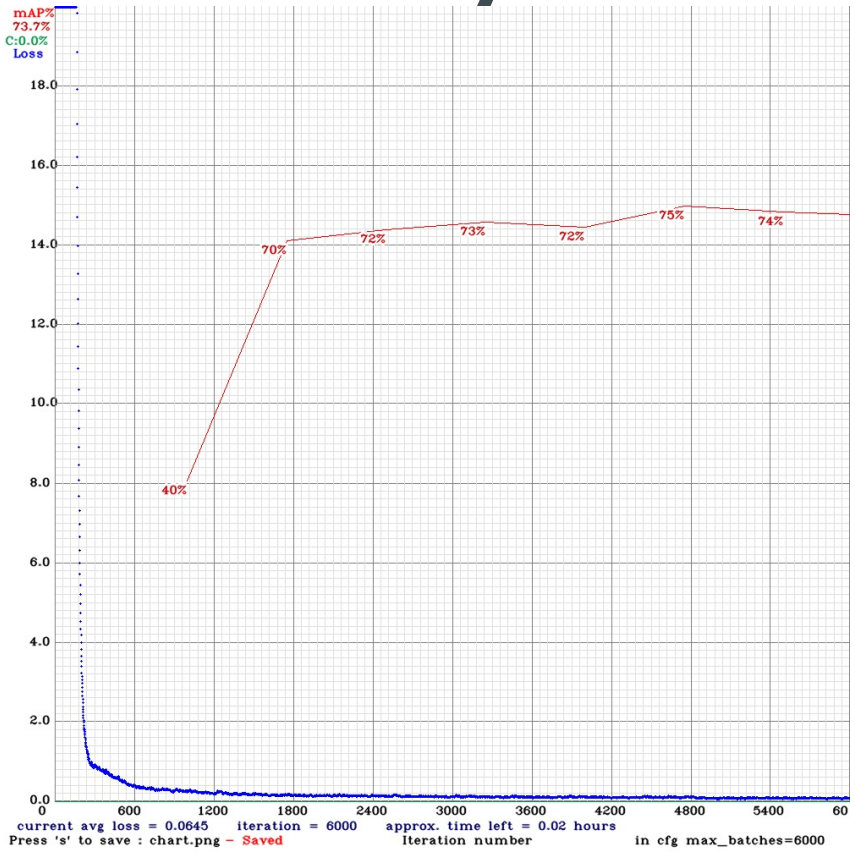
- Training Result of YOLOv4-Tiny Object Detector
- Object Searching Flight Tests
- Object Tracking Flight Tests

Training Result of YOLOv4-Tiny Object Detector

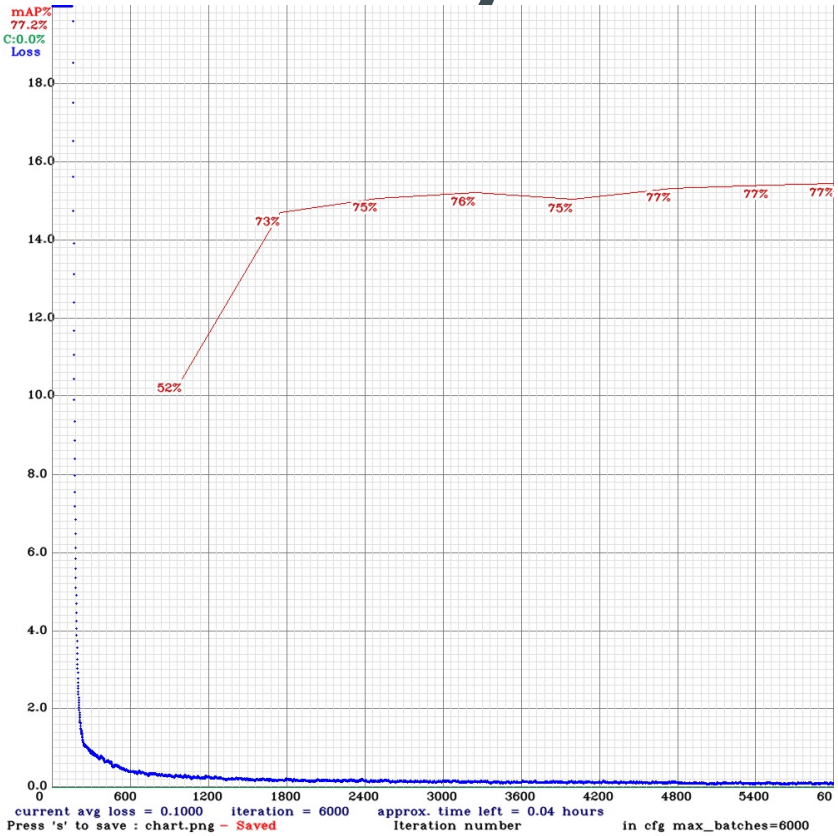
Trained Model

Method	Backbone	Size	mAP@0.50 (AP_{50})	FPS (@TX2)
YOLOv4-tiny	CSPDarknet-53-tiny	(a) 320 × 320	74.85%	5.2539
		(b) 416 × 416	77.21%	5.3657
		(c) 512 × 512	79.36%	5.5503
		(d) 608 × 608	80.20%	5.1557
YOLOv4	CSPDarknet-53	416 × 416	97.09%	1.0521

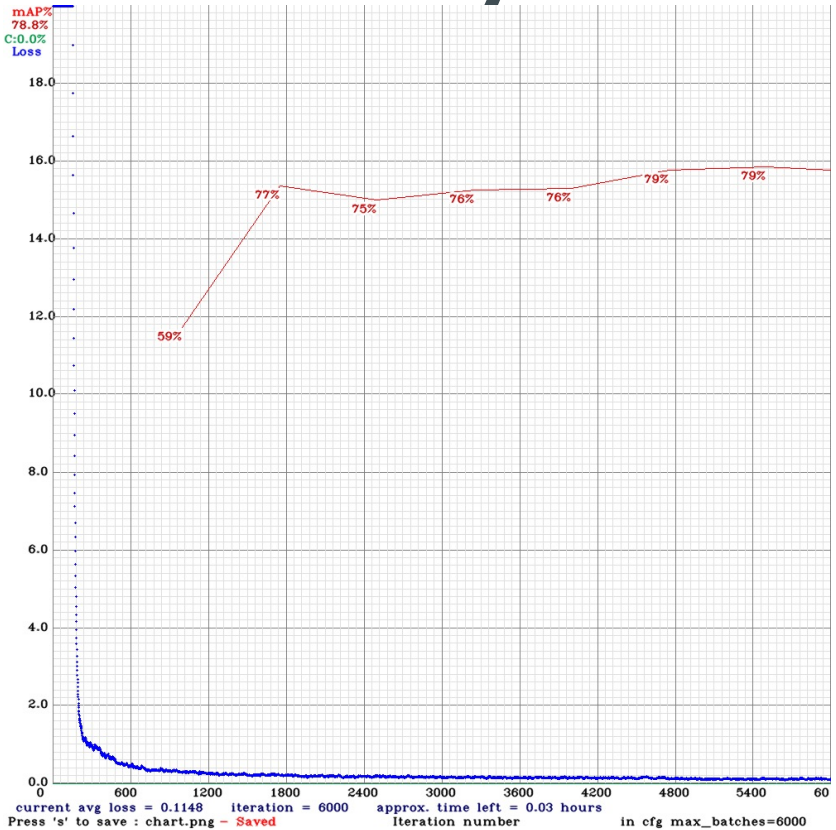
YOLOv4-Tiny 320x320



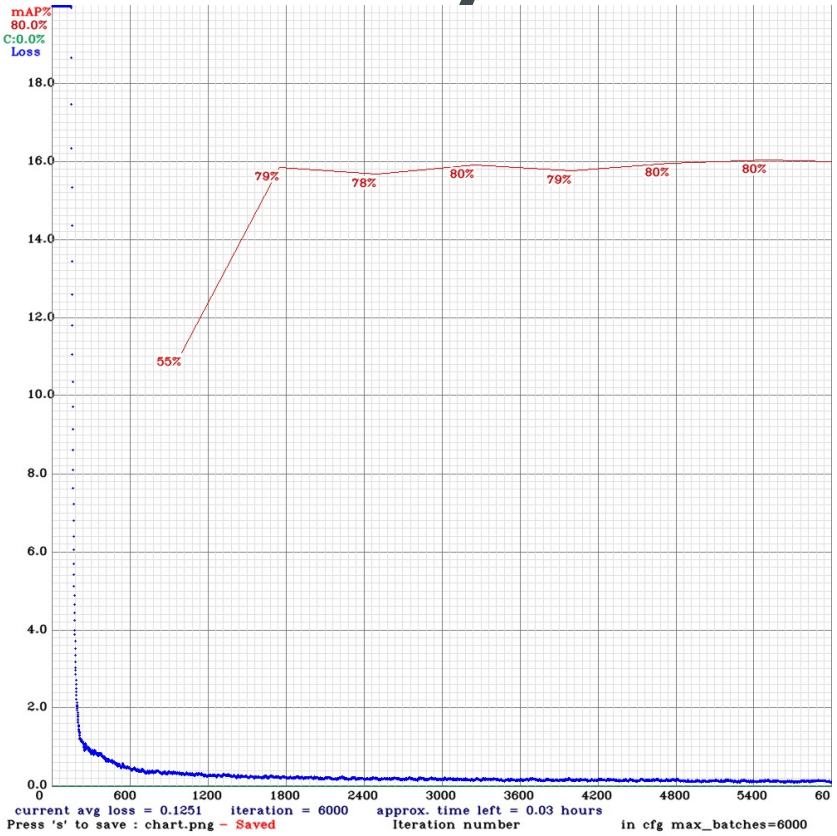
YOLOv4-Tiny 416x416



YOLOv4-Tiny 512x512



YOLOv4-Tiny 608x608

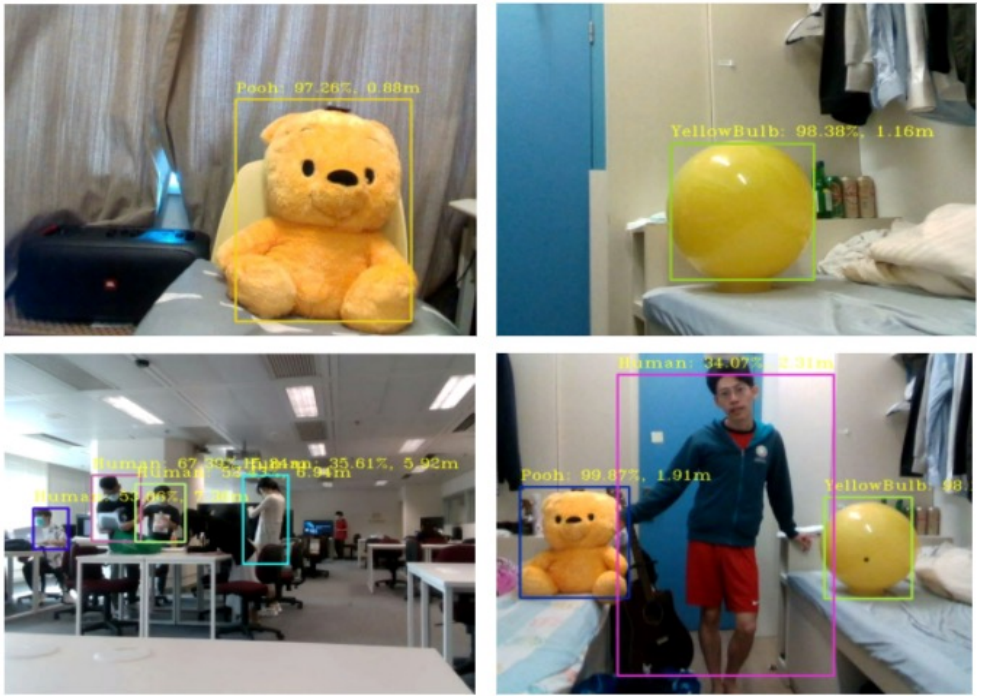


Trained Model

Method	Backbone	Size	mAP@0.50 (AP_{50})	FPS (@TX2)
YOLOv4-tiny	CSPDarknet-53-tiny	(a) 320 × 320	74.85%	5.2539
		(b) 416 × 416	77.21%	5.3657
		(c) 512 × 512	79.36%	5.5503
		(d) 608 × 608	80.20%	5.1557
YOLOv4	CSPDarknet-53	416 × 416	97.09%	1.0521

Chose 512 × 512 as a good balance between detection accuracy and inference speed

Trained Model



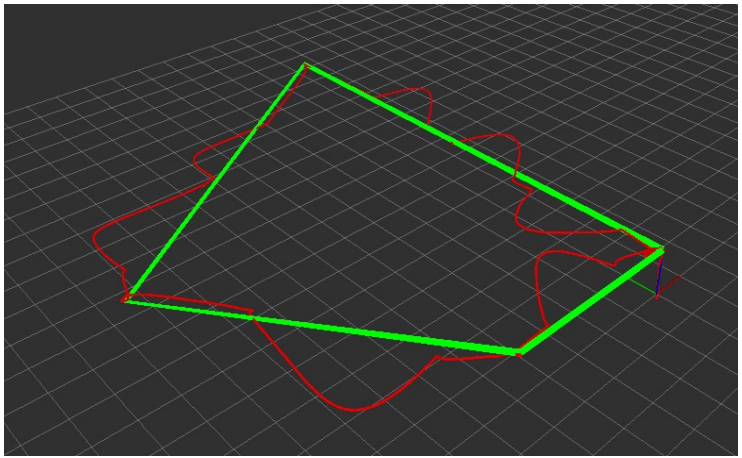
Object Searching Flight Tests

1. Obstacle avoidance in Gazebo simulation

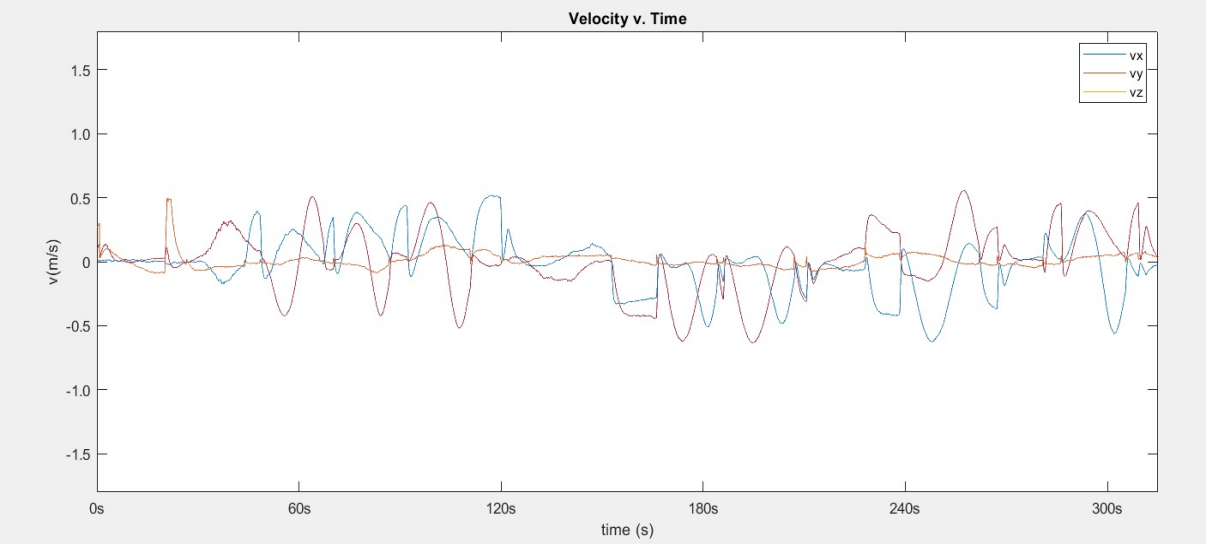
Video

Trajectory shown in RVIZ

- Waypoints are set as:
  - (0,0,1.2)
  - (14,0,1.2)
  - (-10,7,1.2)
  - (-5,0,1.2)
  - (0,0,1.2),



Velocity v Time during flight test

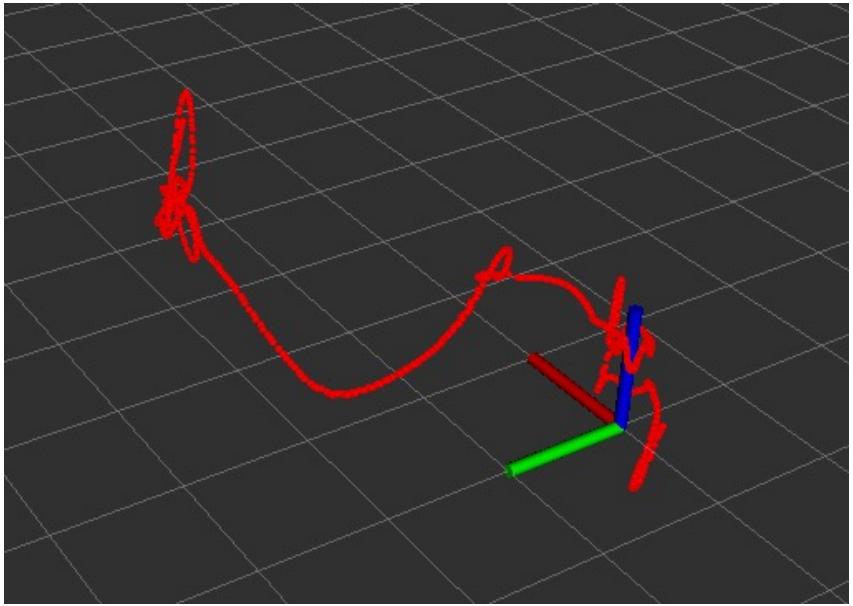


- Successfully passed through all waypoints
- Avoided all obstacles
- Well-controlled velocity (under 0.5 m/s)
- The average computational time: 71.50 ms
  (collision check time + path finding time = 71.50 ms)

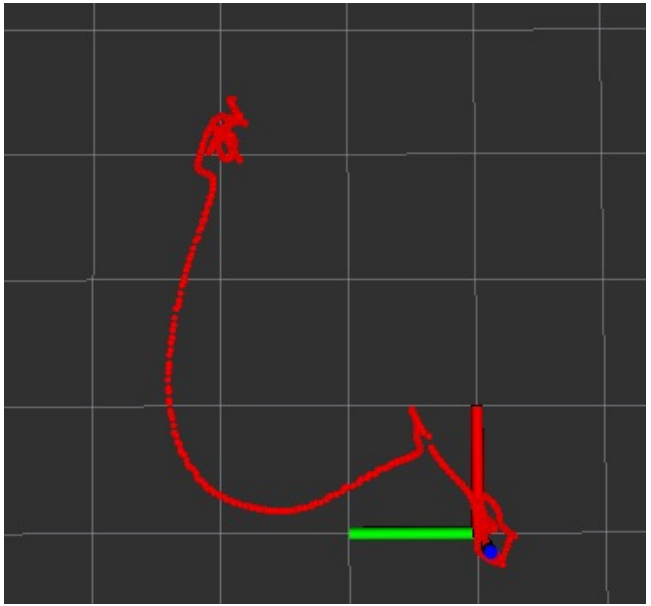
2. Object searching flight test in VICON

Video

Trajectory visualization



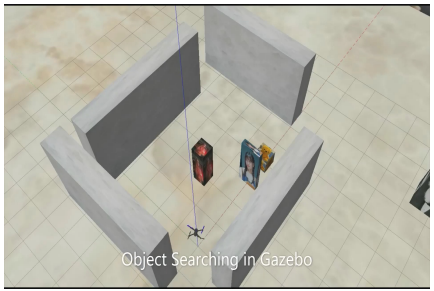
(a) Side view of the trajectory



(b) Top view of the trajectory

Discussion

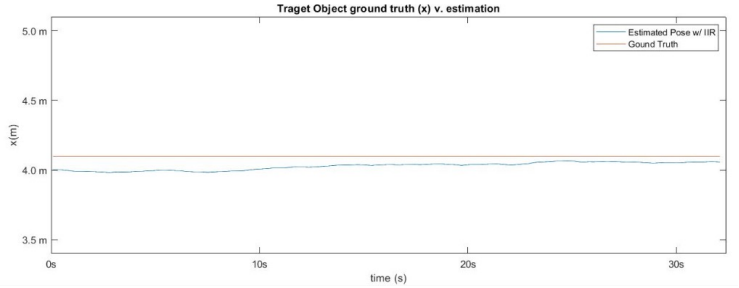
- Object successfully found and obstacle safely avoided
- Yet: UAV oscillated severely when hover and sway
- Recorded object position was not accurate and was thus discarded
- Reconducted in GAZEBO
- Technical issues existed (VICON)

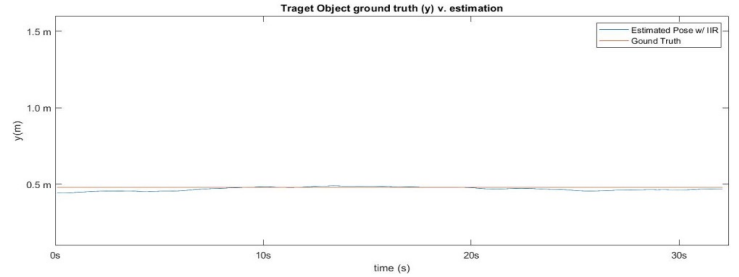


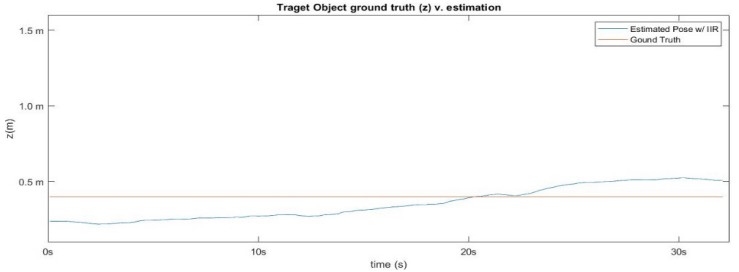
Object Pose Estimation in Gazebo

- IIR filter allows the estimation to converge eventually

- Object pose was located with satisfactory accuracy



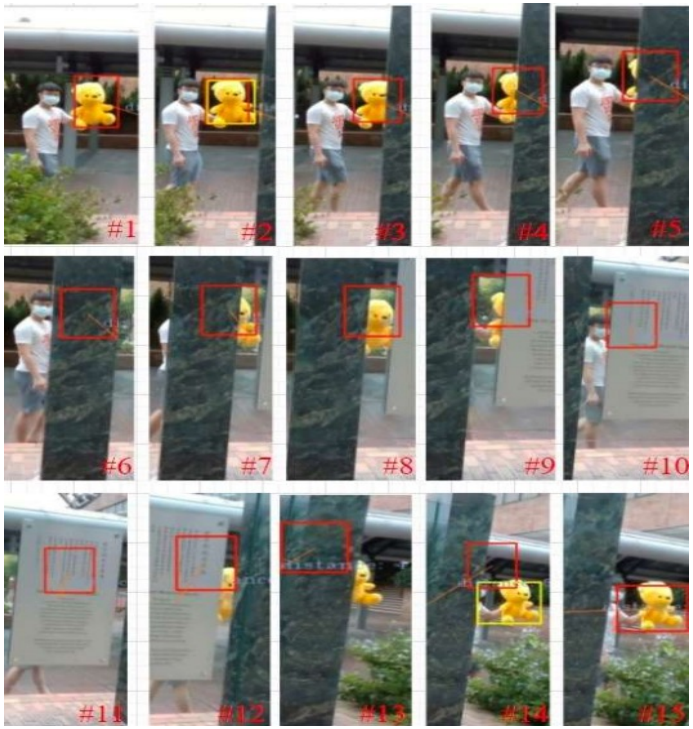




Object Tracking Flight Tests

1. Real-time tracking on image planes

(a) occlusion with short duration



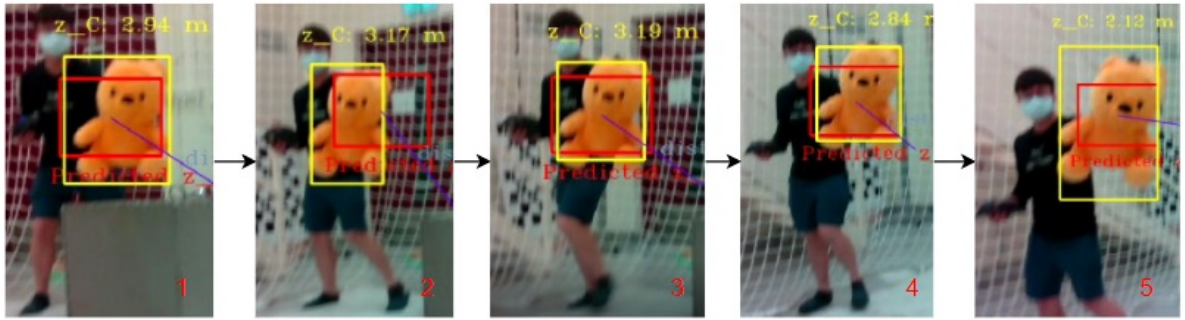
(b) occlusion with long duration

2. Object tracking flight test

Video

Performance Discussion

- UAV was able to track and follow the object steadily



Kalman Filter during tracking

- Despite object not detected by YOLO-Tiny
- Despite occlusion

Kalman Filter was still able to approximately follow the object!

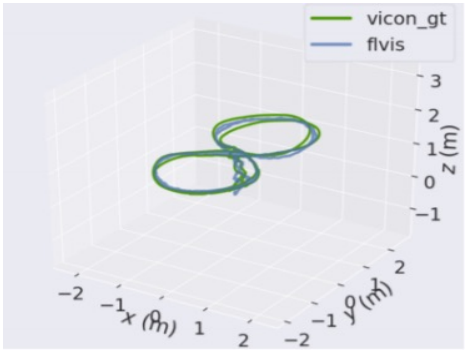
Department of
Aeronautical and Aviation Engineering
航空及民航工程學系

Conclusion

Achievements

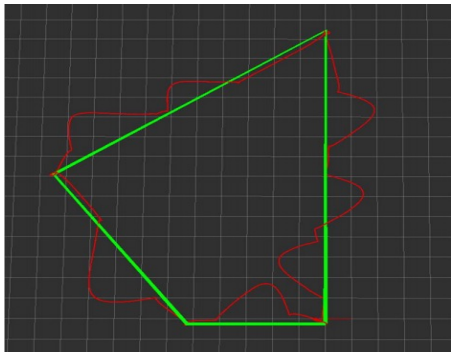
- The objectives were fulfilled.
- The proposed two systems were evaluated and validated.
- accurately search the occurrence and location of target objects in an unknown environment with an autonomous and collision-free flight
- consistently tracking and safely following the movement of a single target object without human intervention
- a step towards autonomous object searching and object tracking applications in real world scenarios of SAR operations or reconnaissance works

Future Recommendations



Independent Localization

Extended Kalman filter or Unscented Kalman filter



Inequality constraints (e.g., corridor check, feasibility check)

Thank you!

Special thanks to:

Prof. Chih-Yung WEN,
Dr. Boyang LI,
Mr. Yu Rong FENG
Dr. Sheng Yang CHEN
Mr. Jeremy Chang

Q&A