國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

於視覺遮蔽環境下針對地面目標之空中協同軌跡追蹤定位與監視系統

Cooperative Aerial Trajectory Tracking, Localization and Surveillance System toward a Ground Target with Vision Occlusion

許芳源

Fang-Yuan Hsu

指導教授: 連豐力 博士

Advisor: Feng-Li Lian, Ph.D.

中華民國 113 年 7 月

July 2024

doi:10.6342/NTU202404031

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

於視覺遮蔽環境下針對地面目標之空中協同軌跡追蹤定位與監視系統

Cooperative Aerial Trajectory Tracking, Localization and Surveillance System toward a Ground Target with Vision Occlusion

本論文係許芳源（學號 R11921081）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國一百一十三年七月二十二日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Electrical Engineering on 22 July 2024 have examined a Master's thesis entitled above presented by Fang-Yuan Hsu (ID R11921081) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

連豐力 (指導教授 Advisor) _____

李後燦 _____

黃正民 _____

江明理 _____

系主任 Director　　李建模 _____

doi:10.6342/NTU202404031

誌謝

歡樂的時光(？)總是過得特別快，兩年很快就過去了，回想當初剛進實驗室還很菜的樣子，就覺得很好笑，卻又同時覺得不堪回首。這兩年歷經了一開始懵懵懂懂，問實驗室的大家問題，都還要敬畏三分，到現在大家能夠分享除了研究以外的生活瑣事，只能說各位真的改變我很多。在這期間也經歷了不少低潮，例如感情上的問題、課業上的不順，甚至是家裡的狀況，但有了大家的扶持，在研究所這兩年期間，讓我感觸很多。

首先感謝我的家人，在說長不長，說短也不短的碩士生涯，對於不管是經濟上的支持，或是生活方面的鼓勵，我了解家中的經濟上，並不如大多數台大學生般寬裕，也必須靠著省吃儉用才能打平，這點我其實一直很難為情，但有您們的支持，包含每個禮拜的電話以及鼓勵，也促使我對於課業與研究方面更有責任感。

接著也感謝實驗室的各位，感謝有田在剛進入實驗室，就給了我三十分鐘演講的當頭棒喝，也感謝您在研究上的幫助及建議，感謝雁丞 aka. Professor Yen 在各種實驗室的事務上，當大家的保母，感謝晁維不管是在無人機研究上，或是生活瑣事上，都給了我很大的幫助，感謝高逵對於我往後研究方向以及工作上的幫助，之後再麻煩您罩了，感謝暄坐在給我不少有關之後職涯發展上的建議，感謝之蕙作為實驗室的氣氛大師，還不忘鼓勵我們，解救我們期中期末的心情，感謝昱翔這兩年的幫助，或許我們也可以在遊戲上有一些交流，感謝宜儒在碩士研究後期，對於研究上眾多的協助，感謝凱正擔任新一代的氣氛大師，在實驗室注入了更多能量，感謝好久不見的瑋恩，對不起占用您的座位這麼久，也感謝 NCS 的傳奇杜哥，不知道什麼時候能再見面，看來之後要再去美國找您了。

感謝旅青、昀誠、芳緯、敬祥、昱彣，以及校外才遇得到的偉銘，有了各位

doi:10.6342/NTU202404031

同儕的互相扶持，是促進我進步的最大原動力，也感謝祐誠、林靖，以及勝凱各位學弟妹，之後實驗室的研究台柱還有氣氛就靠你們了。希望實驗室的各位，已經畢業的也能夠傳承 NCS 的精神，有事就肝，沒事也肝，發揮 NCS Labers 的工作效率與合作精神，也希望還沒畢業的各位，能夠準時畢業，在未來闖出自己的一片天。

感謝口試委員李後燦博士、黃正民博士、江明理博士對於本人論文的指教，給予了我很多寶貴的意見，以及最後也最感謝我的指導教授連豐力博士，除了在研究上循循善誘，給了我莫大的幫助以外，也讓我時時刻刻提醒自己：週報每天寫，進度每週報、FLL is watching you、學術品質若要好，工作時數不可少。這些標語深深烙印在我心中，我也會持續延續這份精神，在未來繼續堅持下去。

騎河濱、Ubike 爬山、深坑臭豆腐、衝陽明山而不是蟾蜍山、尾牙以及謝師宴、突然豐力的會議.....，各種在實驗室的點點滴滴，我想會一直長存在我心中。

doi:10.6342/NTU202404031

於視覺遮蔽環境下針對地面目標之空中協同軌跡追蹤定位與監視系統

研究生: 許芳源　　指導教授: 連豊力 博士

國立臺灣大學　電機工程學系

摘要

無人機可用於許多空中任務，例如探索、檢查、建圖、與環境互動與搜救。在這些無人機的應用中，其中一項是針對警隊。在市區中，追捕嫌疑人或目標是市區警隊的重要任務之一，此方式無需消耗過多人力。然而，長期的訓練過程是非常勞力密集的。同時，在任務過程中，一些畫面的目標估測失誤可能也會發生，這可能是由於障礙物或訊號傳輸故障造成的遮擋情形引起的。

因此，在這篇論文中，我們使用一四旋翼無人機隊解決針對單一目標之空中多群體自主追蹤問題，而此無人機隊會受到障礙物或無法預測之暫時訊號傳輸故障遮擋的影響，造成目標估測暫時失靈。此系統包含使用卡爾曼濾波器 (Kalman Filter) 融合機載圖像與慣性測量單元 (IMU) 測量數據進行無人機的全局定位，根據與無人機相對位置估測目標位置，透過基於軌跡之動作預測解決目標位置暫時的目標估測失效，提取唯一目標狀態的平均演算法，以及基於共識 (Consensus) 之多群體系統隊形追蹤控制演算法。此外也提供了實驗結果來展示系統的可用性。

關鍵字：

四旋翼無人機、多群組系統、卡爾曼濾波器、動作預測、軌跡追蹤、基於共識之隊形控制

doi:10.6342/NTU202404031

Cooperative Aerial Trajectory Tracking, Localization and Surveillance System toward a Ground Target with Vision Occlusion

Student: Fang-Yuan Hsu Advisor: Feng-Li Lian, Ph.D.

Department of Electrical Engineering

National Taiwan University

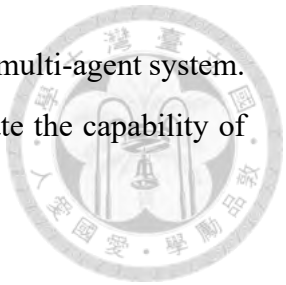
ABSTRACT

UAVs (Unmanned Aerial Vehicles) or drones can be used for many purposes in a lot of aerial tasks, such as exploration, inspection, mapping, interaction with the environment, or even search and rescue. Among these applications of the UAVs, one of them is the police team. Chasing suspects or targets can be one of the most important task for the police in the city without much effort. However, the long process of training would be labor-intensive. Simultaneously, during the tasks, there might be some image-based estimation failure of the target, which may be caused by occlusion from the obstacles or signal transmission failure.

As a result, in this thesis, we solve the autonomous aerial multi-agent tracking problem toward a single target using a team of quadrotors under the occlusion by obstacles or unpredicted temporary signal transmission failure. The system include global localization of quadrotors using data fusion of onboard image and IMU measurement by Kalman filter (KF), target position estimation with the positions relative to the quadrotors, trajectory-based target motion prediction for solving temporarily unpredictable estimation failure of the target position, averaging algorithm for extracting the unique state of the target

doi:10.6342/NTU202404031

and consensus-based formation tracking algorithm for controlling the multi-agent system. Furthermore, We also provide some experiment results to demonstrate the capability of the system.



Keywords:

Quadrotor, Multi-Agent System, Kalman Filter (KF), Motion Prediction, Trajectory Tracking, Consensus-Based Formation Control

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

CONTENTS

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

LIST OF FIGURES

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

LIST OF TABLES

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Chapter 1

Introduction

In this chapter, we will introduce the motivation of the research, problem formulation, contributions and the organization of this thesis.

1.1 Motivation for Using Multi-Agent System

UAVs (Unmanned Aerial Vehicles) or drones can be used for many purposes in a lot of aerial tasks, such as exploration, inspection, mapping, interaction with the environment, or even search and rescue, as shown in Figure 1.1. According to Goldman Sachs Research [1: Goldman Sachs 2016], UAVs evolve into a $100 billion market by 2020. Simultaneously, 70% of the usage of them is in the military in the market, 17% is in the use for the consumers, which is usually for individuals, and 13% is in the use of B2B (business-to-business), such as agriculture, mining and health. Although there are a lot of usages in the military, the latter two are more common for us to see.

Among these applications of the UAVs, one of them is the police team. They use UAVs for mapping cites, chasing suspects, crime scene investigation, 3D reconstruction of accidents, traffic flow management, search and rescue, or even natural disaster relief [2:

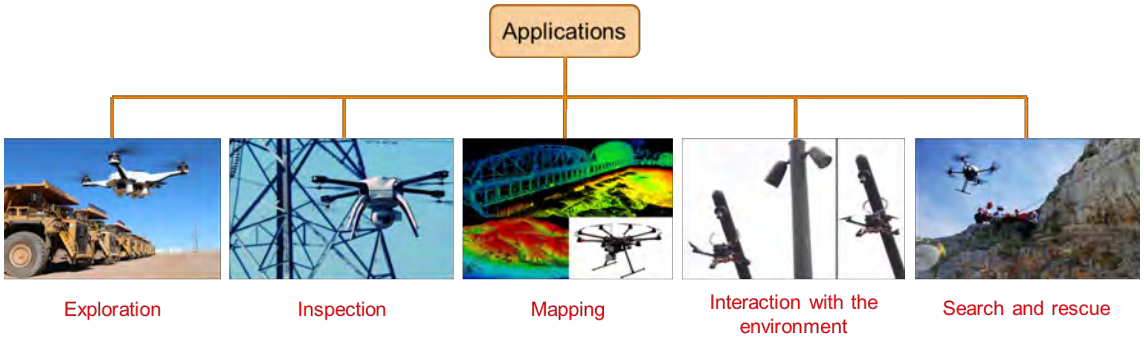


Figure 1.1: The applications of the UAVs.

doi:10.6342/NTU202404031

Droneblog 2021]. Chasing suspects can be one of the most important task for the police in the city. It can track the target without a lot of physical power and obtain the state of the target with efficiency, as shown in Figure 1.2 [3: 華視新聞 CH52].

(a) A news highlight. (b) The onboard image of the drone.

Figure 1.2: A scenario for chasing suspects using a single drone by police teams.

However, in order to operate the UAV in the environment with obstacles, the police have to go through a lot of training such as localization, control and emergency response using GPS (Global Positioning System), IMU (Inertial Measurement Unit) and the image, or even complete several kinds of mission training such as mapping and surveying, photographic documentation of scenes, tactical ISR (information, surveillance and reconnaissance), supply transportation and public relations [4: Police 1 2018], which is labor-intensive. Furthermore, due to the safety, a human can only operate and control a UAV, which will also increase the labor consumption. As a result, developing an autonomous UAV surveillance and tracking system with efficiency is the issue we can think of.

Nonetheless, due to the limitation of the hardware, an autonomous tracking system has lower confidence for most people than the system controlled by human in most of the cases because we may encounter the failure of the communication system [5: 洪哲政 2023] or some occlusion by the obstacles [6: Zhang 2019], which will decrease the confidence on it by people. Another example can be seen in the self-driving car market. Although the driving cars can have less errors than humans nowadays, most people tend

doi:10.6342/NTU202404031

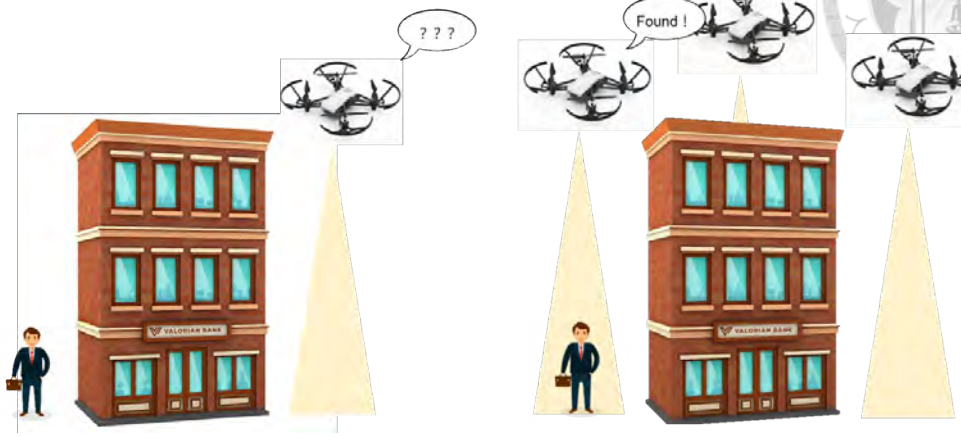
to drive by human [7: Yoshida 2021]. As a result, we need to construct a more reliable and efficient system to convince people that it is much more efficient than the system controlled by humans.

To address the problem, we need to consider issues such as motion prediction and viewing the target from different angles. For the former issue, we can recall the scenario in Figure 1.2(b). Once the target car is moving right, we can infer that it will move right even if it is behind the tree on the right side. For the latter issue, we can consider the behavior of the animals, and one we can consider are the animals living in a group. For example, some birds fly with the V-formation flight or cluster flocking [8: Portugal 2020], locusts and bees fly in a swarm [9: Tylus 2020], [10: Logan Berry Heritage Farm 2018], and hyenas can attack the prey in a formation to increase the perception [11: Young 2023]. These behaviors of the animals can be the capable ways for tracking the target in the real UAV system. There are also some research on the multi-agent UAV system. For example, in [12: Liu *et al.* 2022], the authors proposes the strategies to track the target according to the recognition rate using a team of drones, [13: Wu *et al.* 2022] proposes a real-time formation tracking control protocol with obstacle avoidance, and [14: Chen *et al.* 2022] proposes a decentralized H∞ PID team formation tracking strategy under external disturbance, intrinsic stochastic fluctuation and trailing vortex coupling.

As a result, after referring to these examples, a multi-agent UAV surveillance and tracking system can be a strategy we can consider. As shown in Figure 1.3, if we use a single UAV to track the target, the target may be occluded sometimes, which causes the failure of the task as Figure 1.3(a), but if we use multiple UAVs, it will enlarge the surveillance area and increase the perception of the system as the case of Figure 1.3(b). By doing some information exchange, we have a more reliable surveillance and tracking

doi:10.6342/NTU202404031

system of the UAVs.



(a) Single-agent surveillance. **(b) Multi-agent surveillance.**

Figure 1.3: Surveillance using multiple agents will increase the perception of the system.

1.2 Problem Formulation of Aerial Multi-Agent Tracking

Quadrotor is one of the most common kinds of UAVs. As shown in Figure 1.4 [15: Xie and Lynch 2017], it consists of a body and four sets of motors with propellers. The quadrotor move itself by the adjustment of the speed of each motor. The opposite motors rotate in the same direction. It can control the yaw by the adjustment of the speed of the opposite motors and also adopts the strategy of wheeled take-off and landing technology and integrated navigation mode, which is extremely dynamic to resist the impact of air currents, and has excellent advantages such as anti-surveillance, anti-signal detection and interference.

We first discuss the principle of it. As shown in Figure 1.5 [16: Murtaza's Workshop - Robotics and AI], in Figure 1.5(a), if the rotors exert the same forces and counteract the gravity, the quadrotor will hover. In the case of Figure 1.5(b), if the total force exerted by the rotors is larger or smaller than the gravity, the quadrotor will go up or down vertically,

doi:10.6342/NTU202404031

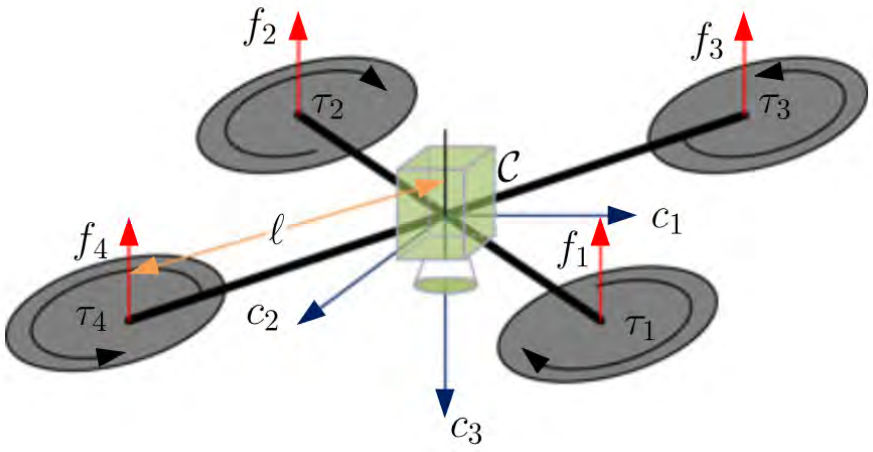


Figure 1.4: The schematic appearance of the quadrotor.

As for the case in Figure 1.5(c), if exerted forces cannot make the quadrotor stable, it will tilt and move sideways, and in Figure 1.5(d), if the opposite motors change their speed, in order to maintain conservation of angular momentum, the quadrotor will make a turn.

As for the dynamics of a quadrotor, we can use the concept discussed before. The total force exert on the quadrotor can affect the motion of the quadrotor and the total angular momentum exerted by the motors can affect the rotation of the quadrotor. As a result, the dynamic model of a quadrotor can be given by

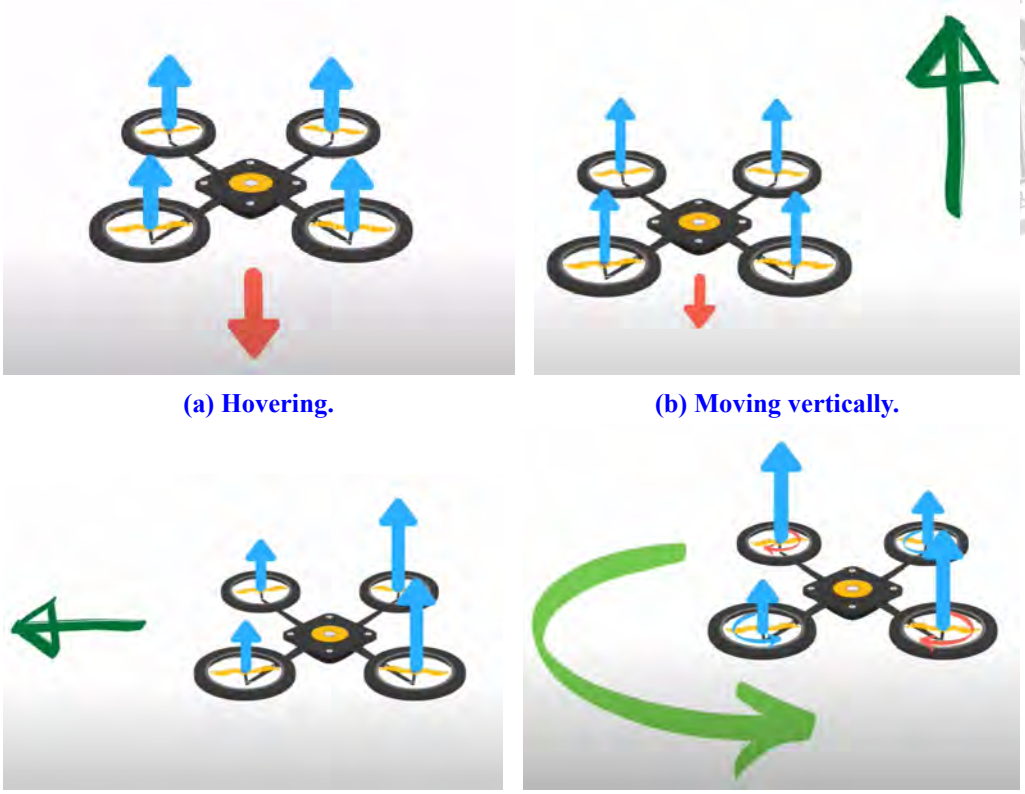
$$m\ddot{x} = -mge_3 + fRe_3, \tag{1.1}$$

$$\dot{R} = R\hat{\Omega}, \tag{1.2}$$

$$I\dot{\Omega} = M - \Omega \times I\Omega, \tag{1.3}$$

where m is the mass of the quadrotor, x is the position in the three dimensions, g is the gravity, $e_3 = [\ 0 \quad 0 \quad 1\ ]^T$, f is the total thrust exerting on the system, $R \in SO(3)$ is the orientation, Ω is the angular velocity expressed in the camera (robot) frame, I is the moment of inertia, M is the moment exerting on the system, and $\hat{\bullet}$ is the hat map of the vector. Among these notations, the thrust f and the moment M are the control inputs to

doi:10.6342/NTU202404031



(a) Hovering. (b) Moving vertically.

(c) Moving sideways. (d) Turning.

Figure 1.5: The principle of flight of a quadrotor.

the system.

In the scenario, we want to design the control command properly to let the formation of the quadrotors keep tracking toward the target with some unpredictable occlusion. As shown in Figure 1.6, the field of view of the triangular formation should cover the target in the ideal case so that at least one agent could see the target and predict the target motion. In the later chapters, we will discuss how to achieve this task.

1.3 Contributions of the Thesis

In the previous section, we mentioned some research about multi-agent tracking and surveillance frameworks. There are some issues we can discuss including localization and tracking strategies.

For the issue of localization, Global Positioning System (GPS) is an important ele-

doi:10.6342/NTU202404031

Figure 1.6: The schematic scenario in this thesis.

ment in the localization. For instance, the authors of [17: Qu and Zhang 2011] use cooperative localization algorithm to solve the unpredictable malfunction of GPS receiver. Nonetheless, there is no GPS device on our drone so that we cannot obtain the global position with it directly. The onboard vision is also one of the strategies. For instance, the authors of [18: Ma *et al.* 2023] proposed the vision-based relative localization as well as the formation tracking framework with YOLOv7. Nonetheless, the frequency of the image is only about 30Hz, which is not suitable for our scenario. The other common device for localization is IMU. Nonetheless, the IMU on the drone we use has severe error accumulation, which is not suitable for use as a single sensor, as described in Appendix G. Among the disadvantages above, more research focus on the sensor fusion algorithms.

As for the sensor fusion for localization of the drone, the authors of [19: Wang *et al.* 2022] use UAV swarm localization system based on probabilistic data association (PDA) for vision-based relative measurements and integrate with the IMU measurements and broadcast information with extended Kalman filter (EKF). In [20: Zheng *et al.* 2023], the authors fuse the information of multi-UAV ranging and user-side IMU module using unscented Kalman filter (UKF) to solve the partial GNSS-denied environment. These

doi:10.6342/NTU202404031

fusion algorithms can be the references of the localization frameworks in our system.

Furthermore, for the issue of drone tracking toward the target, the authors of [21: Thomas *et al.* 2017] proposed an trajectory error-based cost function to make a quadrotor track with efficiency. This paper gives us the concept of trajectory-based prediction but lacks the solution of multi-agent problem. The authors of [12: Liu *et al.* 2022] proposed a multi-agent tracking method. This paper gives us the concept of the formation of drone teams but has lack of the elementary concepts in dynamics. In [22: Li *et al.* 2019], the authors proposed a consensus-based formation tracking control algorithm. It solves the problem of slow convergence speed and low accuracy of the formation but only validate with simulation, which means it assume the conditions are all ideal including localization accuracy and target estimation. That is, the strategies of these papers about drone tracking consider the localization framework of the agents is always correct, which will not suitable for our case.

The contributions of the thesis are that it integrates and solve the problems when using low-cost quadrotors such as IMU accumulation, estimation failure of the target and occlusion from the obstacles or transmission failure and propose a workflow to solve these problems.

All in all, this thesis aims to develop a quadrotor surveillance and tracking system that efficiently tracks a single target. The system encompasses localization, estimation, and motion prediction of the target, as well as extracting the unique reference state of the target, formation tracking, and control of the quadrotors. We validate the enhanced tracking algorithms by using obstacles with known positions in the world frame. The target trajectories are utilized to predict its motion, and an averaging method is employed for

doi:10.6342/NTU202404031

determining the unique reference state of the target. Additionally, the formation tracking and control algorithm is used to track the target in a formation, ensuring group discipline and avoiding collisions. These strategies will also be discussed in the later chapters.

1.4 Organization of the Thesis

In this thesis, there are some main objectives we can consider. Chapter 1 introduces some issues of the current problems. Chapter 2 presents a literature survey of different methods. Chapter 3 covers the related works that forms the basis for our study. Chapter 4 are the proposed the methodologies we use in the system. Chapters 5 and 6 present experimental results and validations with conclusions and future works.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Chapter 2

Literature Survey

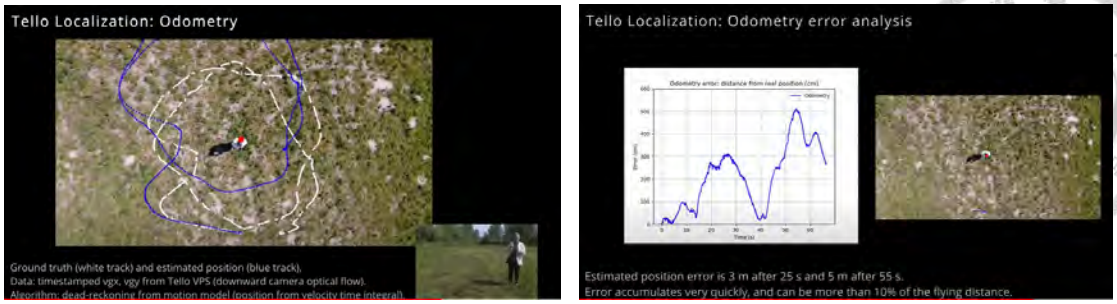
This thesis focuses on constructing a multi-agent quadrotor tracking and surveillance system. Some references including localization, visual servoing and formation tracking will be organized in this chapter.

2.1 Accurate Localization with Vision Assistance

We use the quadrotor "Tello" as our control plant. As shown in Figure 2.1, this kind of quadrotor has an IMU and a monocular camera in front of it. IMU can measure the velocity and the acceleration and obtain the position by integration. However, the IMU in this kind of quadrotor has a fatal flaw that it has a severe IMU error accumulation. As shown in Figure 2.2 [23: NightjarOne], as the quadrotor travels longer, the error measured by the IMU will accumulate, where the white curve in Figure 2.2(a) is the ground truth and the blue curve is the position estimation of the IMU. Simultaneously, the graph shown in Figure 2.2(b) demonstrates the error can be up to 500 centimeters within 50 seconds of traveling. In order to solve this problem, we can use the onboard image to improve the localization problem.

Figure 2.1: The onboard sensors on Tello.

doi:10.6342/NTU202404031



(a) Trajectory visualization in 2D space. **(b) Odometry error (cm) within time.**

Figure 2.2: The effect of error accumulation.

There are several strategies for localization. For example, as shown in Figure 2.4, we can do localization using visual odometry (VO), visual inertial odometry (VIO) or by the landmark. These techniques can be applied on the localization of the quadrotors with GPS and LiDAR unavailable.

2.1.1 Localization with Visual Odometry (VO) with Vision Directly

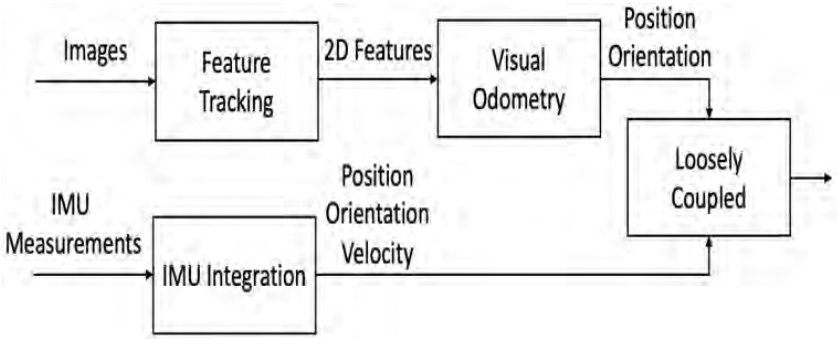
Visual odometry (VO) is a technique to obtain the pose of the plant using the information of the image. By calculating the motion of the images in the time sequence in [24: Nister *et al.* 2004], the autors use VO for pose estimation. However, VO has the problem of accuracy if the texture of the environment is not such obvious. Simultaneously, the sampling frequency is also the problem. The frequency of the video is about 30Hz generally, which is not sufficient, so we can use IMU whose the sampling frequency can be up to 1000Hz to compensate the disadvantage.

2.1.2 Localization with Visual Inertial Odometry (VIO) Based on Fusion of Vision and IMU Information

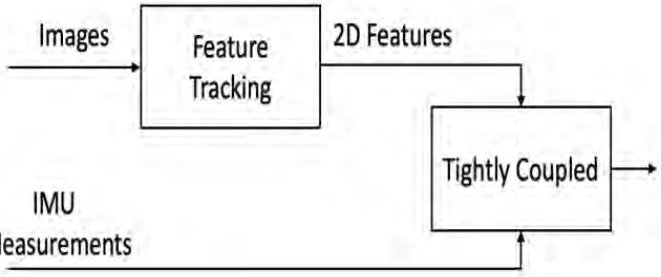
Visual inertial odometry (VIO) combines the visual odometry and inertial odometry together to take the advantages of these two methods. Generally, it can be divided into two

doi:10.6342/NTU202404031

categories: loosely coupled and tightly coupled VIO. As shown in Figure 2.3 [25: Tang *et al.* 2020], loosely coupled VIO conbines the position and orientation calculated by visual odometry and the position, orientation and velocity calculated by IMU odometry together [26: Liu and Shen 2017], [27: Weiss *et al.* 2013], [28: Weiss and Siegwart 2011]. It has lower computation cost but is hard to fix the error of inertial with image. Tightly coupled VIO combines the features of the image in 2D [29: Forster *et al.* 2017], [30: Caruso *et al.* 2017] or 3D [31: Mourikis and Roumeliotis 2007], [32: Palezieux *et al.* 2016] with the IMU measurement directly. It has higher computation cost but higher accuracy as well.



(a) Loosely coupled.



(b) Tightly coupled.

Figure 2.3: The workflow of loosely and tightly coupled VIO.

2.1.3 Landmark-Based Localization, a Modified Version of VIO

Landmark is also an information we can think of. It has the advantage that we can localize the plant without preliminary information on the environment. For example, authors of [33: Han *et al.* 2013] propose a novel landmark-based particle localization algorithm

doi:10.6342/NTU202404031

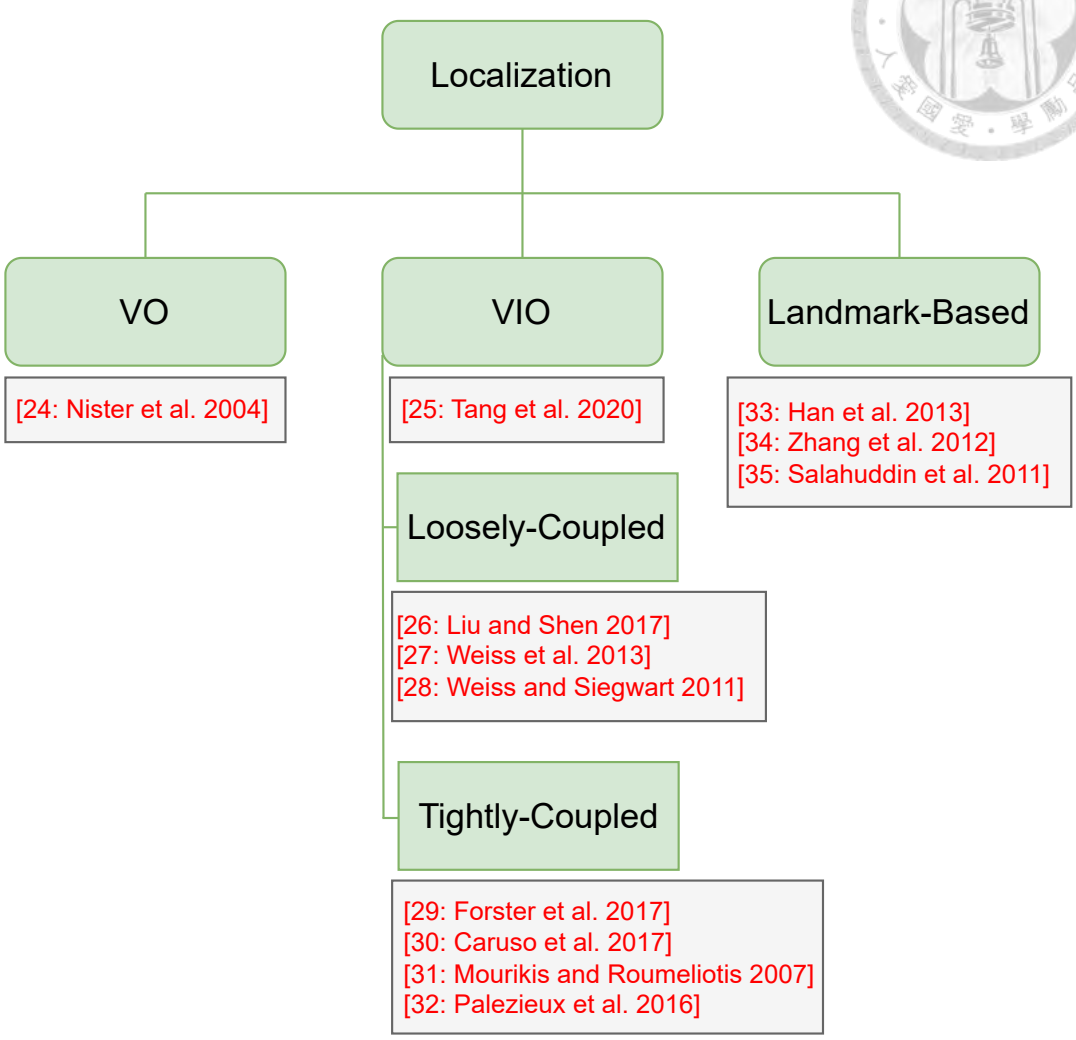


Figure 2.4: Literature survey of localization.

for the global localization problem called relocation, authors of [34: Zhang *et al.* 2012] present a localization algorithm of indoor mobile robot based on landmark, and authors of [35: Salahuddin *et al.* 2011] construct a novel Artificial Landmark-Based Identification System (ALIS) and make the computation faster with better distance estimation. If we use the landmark with known pose, by the estimation of the camera image, we can update the pose of the quadrotor to correct the problem of error accumulation of the IMU and make the pose of the quadrotor more accurate.

doi:10.6342/NTU202404031

2.2 Visual Servoing for Robots

Visual servoing is a technique to control the plant using the information of image. Camera is the most common vision sensor in the applications. Generally, it can be divided into to categories including position-based visual servoing (PBVS) and vision-based visual servoing (IBVS) as shown in Figure 2.5.

2.2.1 Position-Based Visual Servoing (PBVS) Based on Position Feedback

Position-based visual servoing (PBVS) uses the estimated position or pose of the target to control the plant. For example, authors of [36: Popova and Liu 2016], [37: Zhao *et al.* 2020] and [38: Liu *et al.* 2021] develop a controller based on the position to track the target. However, the quadrotor system is not a stable system, so during the flight, the vibration would occur, which could make the recognition of the target unstable. As a result, recording the trajectory and predicting the motion of the target can solve the problem. The authors of [21: Thomas *et al.* 2017] and [39: Chen *et al.* 2016] use the positions of the target with time to construct the trajectory of the target and use the sequential quadratic programming (SQP) method to construct an objective function under some constraints and optimize the trajectory and control the quadrotor(s).

2.2.2 Image-Based Visual Servoing (IBVS) Based on Image Features Feedback

Image-based visual servoing (IBVS) uses the information of the image based on the 2D image feature on the image plane directly on the control law design. Authors of [40: Serra *et al.* 2015] use IBVS to design the proposed nonlinear controller. In [41: Xie and Lynch 2017], the authors use this technique to construct an adaptive output feedback

doi:10.6342/NTU202404031

controller. Furthermore, in [42: Chesi and Shen 2012], the authors also use it for path planning of the aerial vehicles.

2.2.3 Hybrid Approach, Combination of PBVS and IBVS

As for the pros and cons of these two strategies, PBVS has the advantage of the accuracy of the target pose, but it is computationally expensive. IBVS has the advantage of low computation cost but has difficulties with very large rotations of the camera, and we call it camera retreat.

As a result, some papers also propose the hybrid approach. In [43: Malis *et al.* 1999] and [44: Corke and Hutchinson 2001], the authors propose the new visual servoing strategies by combining these two methods together to improve the advantages of them.

Among these methods, we will use PBVS for controlling the quadrotors because we need to obtain the exact state of the target in the world frame to share the information with each agent and accomplish the surveillance and tracking task.

2.3 Formation Tracking for Multi-Agent Systems

The reason we want to do the formation control during tracking is that we want to make the multi-agent quadrotor system track with discipline and maintain the distance of safety. In [45: Chang 2020], the author divided the formation tracking control into several categories, including leader-follower, potential function-based, virtual structure, consensus-based, behavior-based and intelligent methods, etc. Leader-follower is the method to use a leader and several followers [46: Wang 1991], [47: Desai *et al.* 1998], [48: Do and Pan 2007]. It is easier to implement but collision problem may occur due to the lack of communication between the followers and may fail with the failure of the leader.

doi:10.6342/NTU202404031

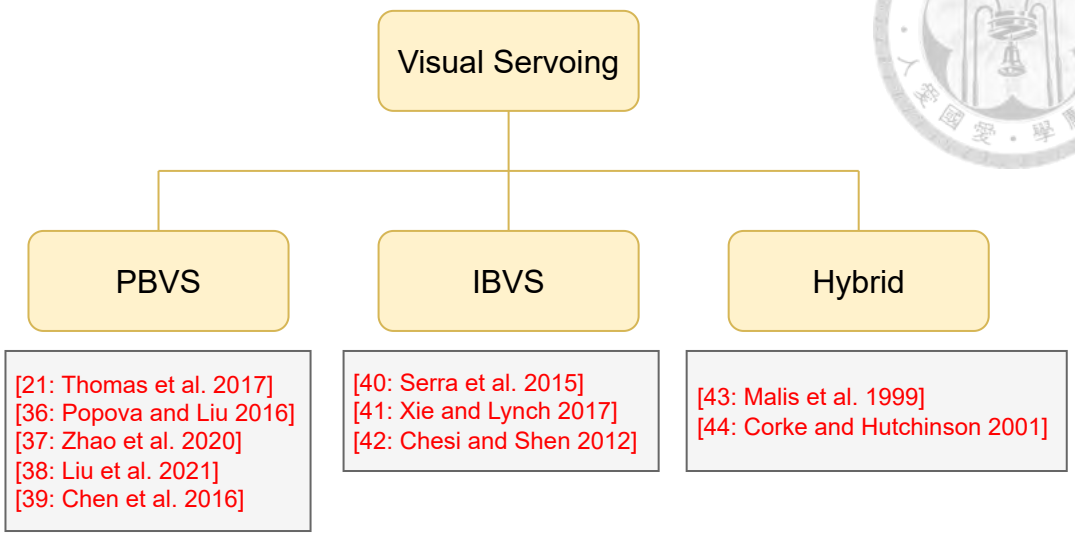


Figure 2.5: Literature survey of visual servoing.

Potential function-based method [49: Do 2008], [50: Yu *et al.* 2019], [51: Pang *et al.* 2021] uses the potential function to make the agents interact with attractive and repulsive forces but may cause the deadlock. Virtual structure method [52: Beard *et al.* 2000], [53: Kang *et al.* 2000], [54: Leonard and Fiorelli 2001] considers the formation as a rigid body but it is the open-loop control without feedback. Consensus-based method [55: Chen *et al.* 2021], [56: Bae *et al.* 2021], [57: Liu and Huang 2021] uses the consensus problem to maintain and move the formation but cannot prevent the collision with each other during formation configuration or reconfiguration phases. Behavior-based method [58: Balch and Arkin 1998], [59: Schneider-Fontan and Mataric 1998], [60: Lawton *et al.* 2003] uses several desired behaviors for each agent, and the control is from the weighting of the importance of each behavior but may cause destructive interference. Intelligence method [61: Lu 2020], [62: Huang *et al.* 2022], [63: Er *et al.* 2024] controls the formation by learning. However, because it is a data-driven method, we cannot analyze the system by

doi:10.6342/NTU202404031

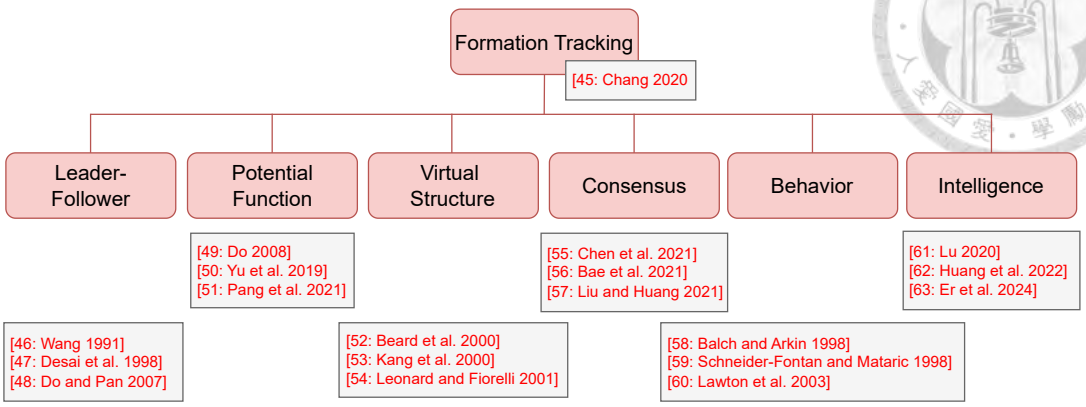


Figure 2.6: Literature survey of formation tracking.

the mathematical derivation, which is not easy to derive the equations and prove the stability of the system. The summary of formation tracking for multi-agent system is shown in Figure 2.5. In Chapter 3, we will further introduce the objective of these algorithms.

As a result, this thesis take the emphasis on the localization with vision and IMU information, visual servoing and formation tracking with vision information as the literature usages. Each issue has several strategies, and we can take these strategies into consideration to design our system. The summary of this chapter is shown in Figure 2.7.

doi:10.6342/NTU202404031

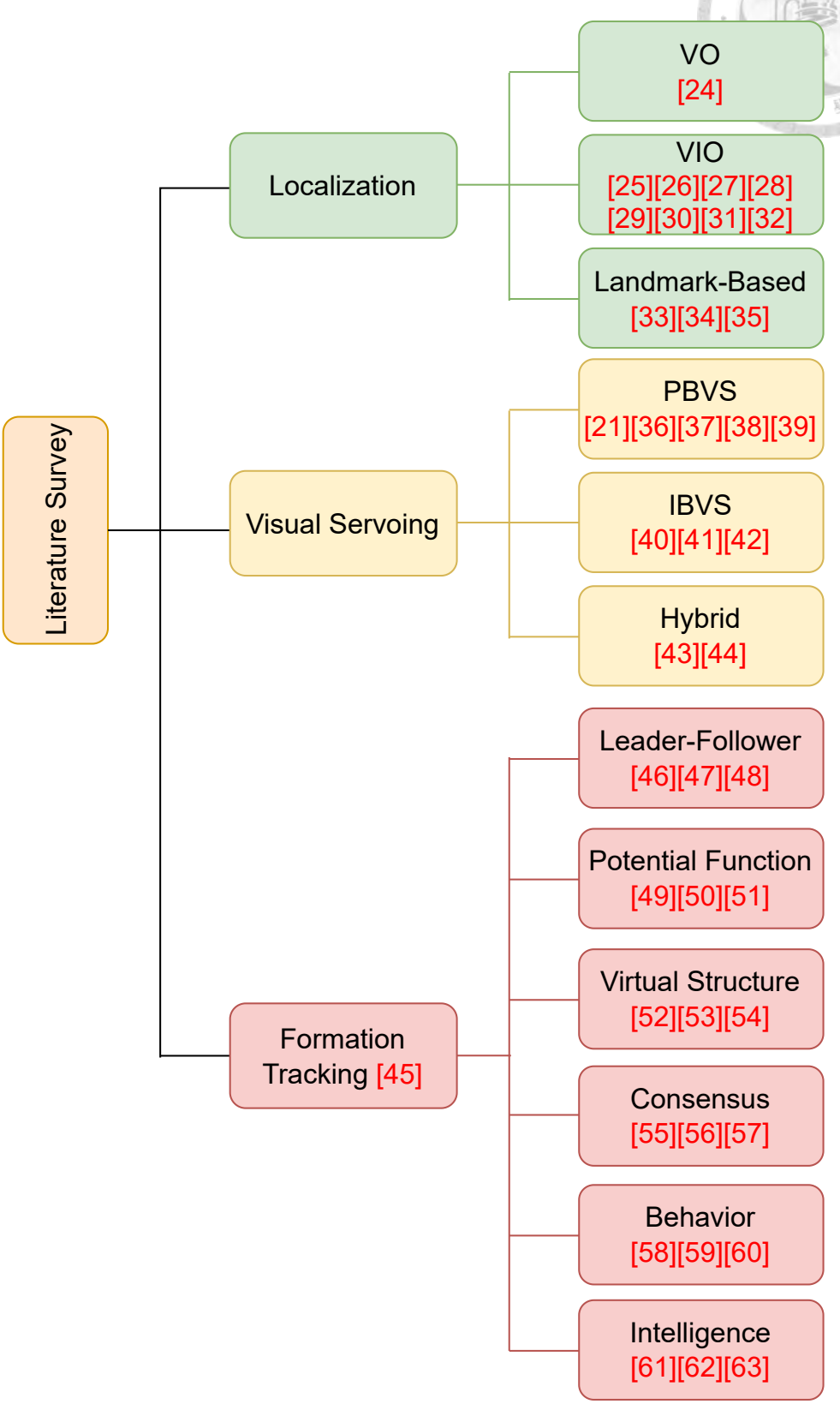


Figure 2.7: Summary of literature survey.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Chapter 3

Related Works of the System

3.1 Camera Pinhole Model, a Coordinate Transformation between Camera Image and World Frame

Camera is a common sensor we can use in the applicaitions of robotics. The imaging principle of a camera is based on several fundamental principles in optics. When light passes through the camera's lens, it is focused onto a photosensitive component. Pinhole camera model is a simplified model representation. As shown in Figure 3.1 [64: Ortiz *et al.* 2017], as light rays from different points in the scene pass through the pinhole, they converge to form an inverted image on the imaging plane, typically a sensor or film.

According to [65: Tsai 1987] referred to Figure 3.2, there are four steps to do the transformation of the camera coordinate, including

1. Rigid body transformation from the object world coordinate system.

2. Transformation from 3D camera coordinate to ideal (undistorted) image coordinate.

3. Distortion adjustment.

4. Real image coordinate to computer image coordinate coordinate transformation.

First we define the world frame $\{W\}$, camera frame $\{C\}$ and image frame $\{i\}$, they

doi:10.6342/NTU202404031

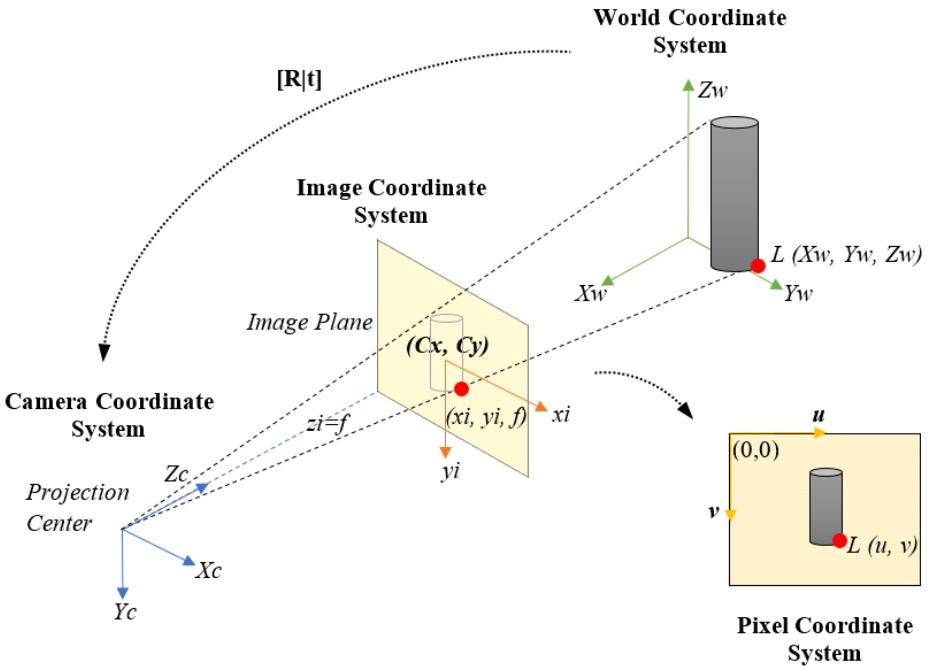


Figure 3.1: Camera pinhole model.

have coordinate transformation below.

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = R \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + T, \tag{3.1}$$

where R is the rotation matrix and T is the translation from frame $\{W\}$ to frame $\{C\}$.

After defining the transformation from world frame to camera frame, we then project the perception from camera frame $\{C\}$ to image frame $\{i\}$. According to the characteristics of homothetic triangles in pinhole geometry, we can define

$$x_i = f_x \frac{X_C}{Z_C}, \tag{3.2}$$

$$y_i = f_y \frac{Y_C}{Z_C}, \tag{3.3}$$

where f_x and f_y are the focal lengths of the camera in x_i and y_i directions in pixel.

doi:10.6342/NTU202404031

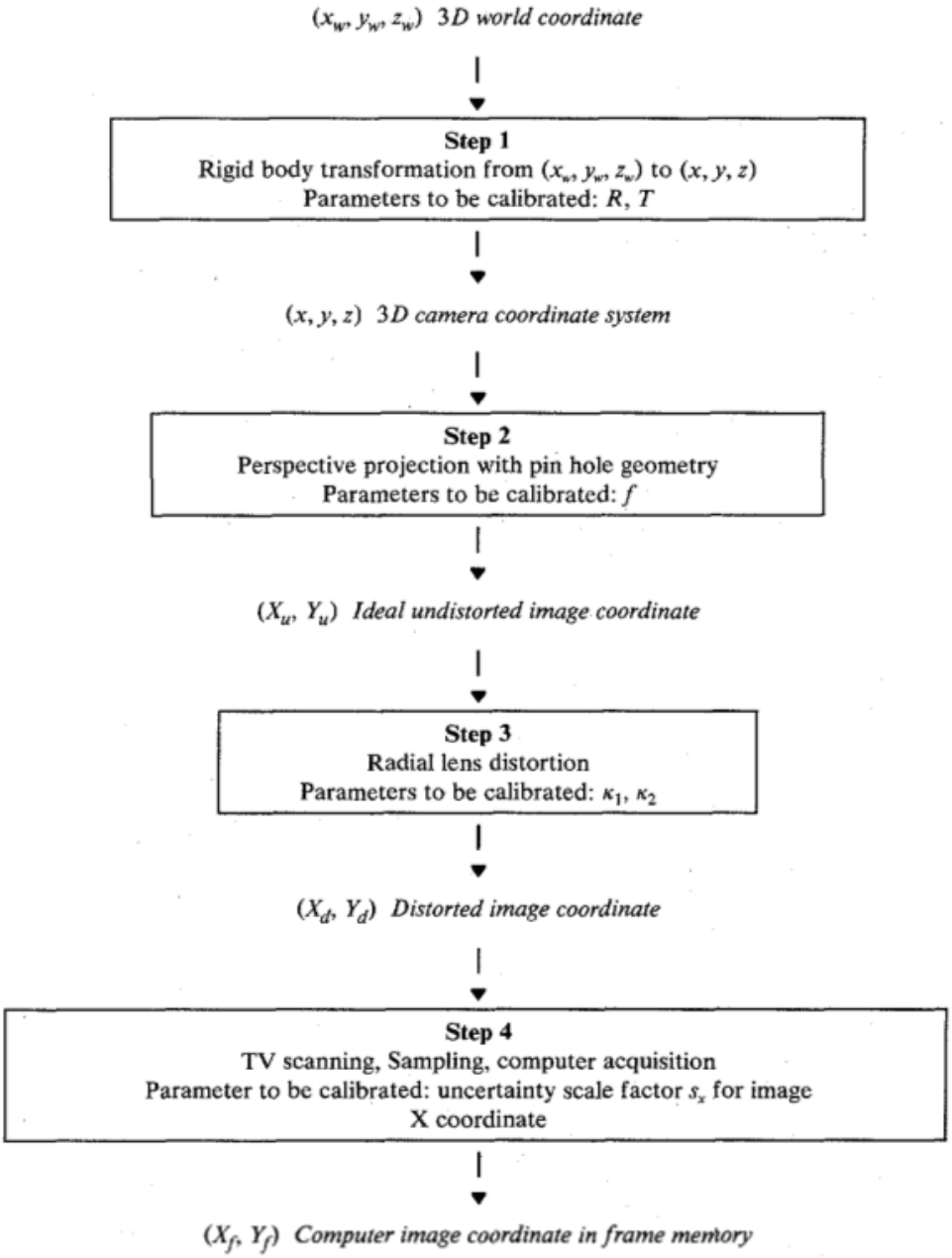


Figure 3.2: The steps for camera coordinate transformation.

doi:10.6342/NTU202404031

Figure 3.3: Radial distortion on the edge of the image.

Third, radial distortion is a phenomenon when the lens is not ideal, which would make the image distort on the edge of the image. As shown in Figure 3.3, we can observe that the fan on the upper left corner is partly distorted. As a result, we need to solve this kind of problem by radial distortion correction. We define the radial distortion correction of the lens.

$$\begin{bmatrix} x_{id} \\ y_{id} \end{bmatrix} = L(r) \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \tag{3.4}$$

In this equation, the left hand side is the coordinate after distortion correction. We use the function $L(r) = 1 + k_1 r^2 + k_2 r^4 \dots$, where $r = \sqrt{(x_i - x_{ic})^2 + (y_i - y_{ic})^2}$ is the Euclidean distance between the distorted image point and the distortion center to do radial distortion correction.

Finally, we transform the image coordinate (x_{id}, y_{id}) to the computer image coordi-

doi:10.6342/NTU202404031

nate (x_f, y_f), as shown below.

$$x_f = s_x d'^{-1}_x x_{id} + C_x, \tag{3.5}$$

$$y_f = d'^{-1}_y y_{id} + C_y, \tag{3.6}$$

where (C_x, C_y) is the row and column numbers of the center of computer frame memory, $d'_x = d_x \frac{N_{cx}}{N_{fx}}$, $d'_y = d_y \frac{N_{cy}}{N_{fy}}$ where (d_x, d_y) are the center to center distance between adjacent sensor elements in x and y (scan line) directions, (N_{cx}, N_{cy}) are the numbers of sensor elements in the x and y directions, and (N_{fx}, N_{fy}) are the numbers of pixels in a line as sampled by the computer.

Among the equations above in this section, the parameters we want to calibrate are the rotation matrix R and translation vector T from frame $\{W\}$ to frame $\{C\}$, the focal length of the camera (f_x, f_y), the coefficients $k_1, k_2 \ldots$ for solving the radial distortion and the uncertainty image scale factor s_x. However, the framework in step 4 is not the element we want to discuss since it is not the work we can modify. As a result, in the later chapter, we will discuss the frameworks from step 1 to step 3 for camera calibration to solve the parameters.

3.2 Kalman Filter for State Estimation and Sensor Fusion

Kalman Filter (KF) is an algorithm used for state estimation in systems that are subject to noise and uncertainty. It can be widely used in sensor fusion or state observer. As shown in Figure 3.4 [66: Welch and Bishop 2006], it has two main steps including prediction and correction. Prediction is the process to project the state and error covariance

doi:10.6342/NTU202404031

ahead. We first define the equation in the digital state space model and the prediction the error covariance

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}, \tag{3.7}$$

$$P_k^- = AP_{k-1}A^T + Q, \tag{3.8}$$

where Q is the process noise covariance matrix.

On the other hand, correction is the process to compute the Kalman gain. Simultaneously, we update the estimated state according to the measurement z_k and update the error covariance using Kalman gain. The equations can be defined as

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}, \tag{3.9}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-), \tag{3.10}$$

$$P_k = (I - K_k H)P_k^-, \tag{3.11}$$

where R is the measurement noise covariance matrix, H is the Jacobian matrix of the observation model, and I is the identity matrix. We use the definition of the equations to do the process of the algorithms.

3.3 Mathematical Optimization Problems

Mathematical optimization is a general term of all optimization problems. In this chapter, we will introduce some mathematical optimization problems, especially in convex

doi:10.6342/NTU202404031

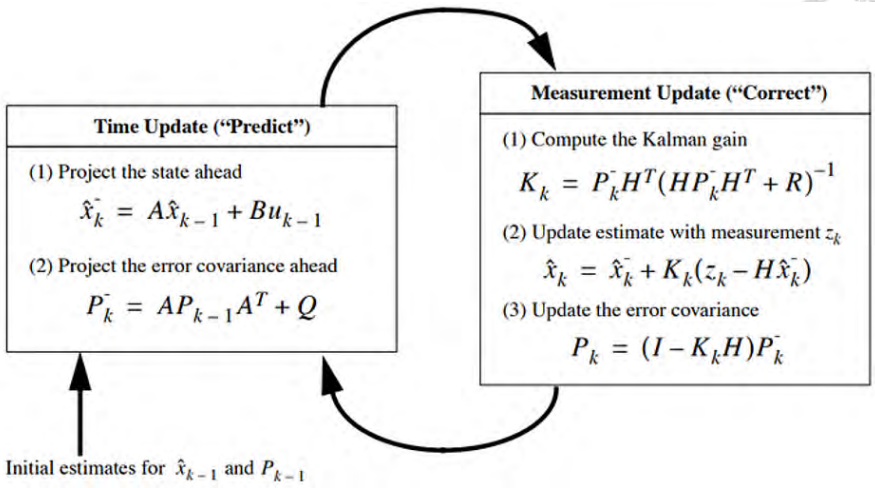


Figure 3.4: The workflow of the Kalman filter in mathematical interpretation.

optimization. It has the form as

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, p. \end{aligned}$$

In this optimization problem, $f_0 : \mathbf{R}^n \to \mathbf{R}$ is the objective function which we want to minimize, $f_i : \mathbf{R}^n \to \mathbf{R}$ are the inequality constraints, and $h_i : \mathbf{R}^n \to \mathbf{R}$ are the equality constraints. We want to minimize the objective function under some inequality and equality constraints.

Convex optimization is a problem to minimize/maximize a convex function under some constraints. It is widely used on signal processing, finance, communications, energy

doi:10.6342/NTU202404031

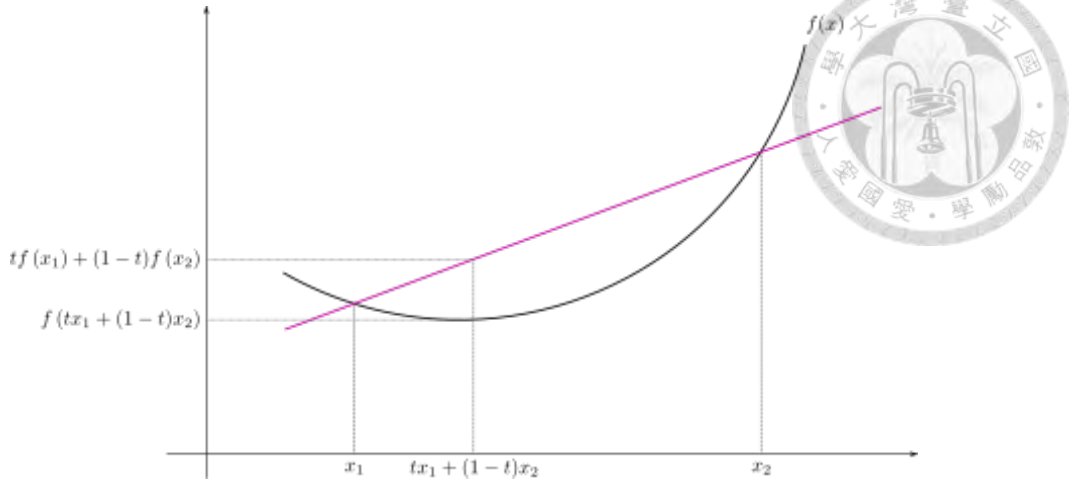


Figure 3.5: The interpretation of a convex function.

and control systems, etc. The standard form of convex programming can be written as

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, p, \end{aligned}$$

where $f_i(x)$ are convex functions and $h_i(x)$ are affine, and $x \in R^n$.

Convex function has the characteristic that for any t satisfying $0 \leq t \leq 1$ and any x_1, x_2 in the domain of convex function f, the below inequality should holds.

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2). \tag{3.12}$$

That is, as shown in Figure 3.5 [67: Wikipedia], for any line segment formed by x_1 and x_2, the function values of the line segment should larger or equal to the value of the convex function. As for the affine function, it has the characteristic that it is the sum of a linear function and a constant, as the form below.

$$h(x) = Ax + b, \tag{3.13}$$

doi:10.6342/NTU202404031

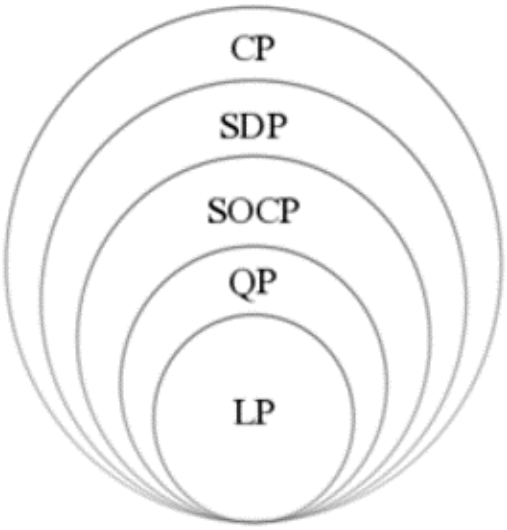


Figure 3.6: Research fields in convex optimization.

where $A \in R^{p\times n}$ and $b \in R^p$. That is, it has equality in (3.28).

$$f(tx_1 + (1-t)x_2) = tf(x_1) + (1-t)f(x_2). \tag{3.14}$$

The categories of the convex optimization problems including linear programming (LP), quadratic programming (QP), second-order cone programming (SOCP), semidefinite programming (SDP) and conic programming (CP) will obey this rule, as shown in Figure 3.6. Besides of QP, other problems are formulated in Appendix C.

3.3.1 Quadratic Programming (QP)

In this thesis, we solve quadratic programming problems for curve fitting of the target trajectory. Quadratic programming (QP) is the problem when the objective function is convex quadratic and the constraint functions are affine, as shown below.

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T P_{qp} x + q_{qp}^T x + r \\ \text{s.t.} \quad & G_{qp} x \preceq h_{qp} \\ & A_{qp} x = b_{qp}, \end{aligned}$$

doi:10.6342/NTU202404031

where $P_{qp} \in S_+^n$ (positive semidefinite symmetric matrix), $G_{qp} \in R^{m \times n}$, $A_{qp} \in R^{p \times n}$, and $b_{qp} \in R^p$.

Convex optimization can be solved by several methods. For instance, Newton's method can be used to solve the optimization problem with no constraints, interior point method (barrier method) uses a logrithmic barrier to optimize the objective function and eliminate the inquality constraints. Besides, gradient descent uses the local gradient to find the minimum whose gradient approaches zero. In fact, there are more than these methods. These methods by iteration can be used to solve the optimization problems in our research.

3.4 Consensus-Based Formation Tracking Control Strategy

Consensus-based formation tracking control is a method to control the multi-agent system according to the states and the communication protocol of the agents. The objective of this algorithm is to make the states, including position and velocity, to converge to a desired value. That is, we want to let $||r_i(t) - r_j(t) - d_{ij}|| \to 0$ and $||\dot{r}_i(t) - \dot{r}_j(t)|| \to 0$ as $t \to \infty$ where d_{ij} is the position offset between i^{th} and j^{th} agents, and $r_i(t), r_j(t)$ and $\dot{r}_i(t), \dot{r}_j(t)$ are the positions and velocities of the i^{th} and j^{th} agents, which will be satisfied for any initial states $r_i(0)$ and $\dot{r}_i(0)$. Other kinds of formation tracking control algorithms are introduced in Appendix D.

doi:10.6342/NTU202404031

Chapter 4

Methodologies

4.1 System Overview for Vision-Based Multi-Agent Trajectory Tracking System with Landmark-Based Localization

Before we introduce the techniques we use. We first introduce the system overview. As shown in Figure 4.1, we take the drone Tello as the control plant. We can do state estimation using onboard IMU and camera, and do sensor fusion using linear Kalman filter (KF) to make the state more accurate. After obtaining the estimated drone state, we can obtain the state of the target, and after recording it with time, we can obtain the trajectory with time to predict the motion. We use the message of current state of the drone and the trajectory of the target as the input of the trajectory planner to obtain the reference trajectory. Finally, a position-based visual servoing (PBVS) controller is applied to send the velocity command to control the drone, which achieves the purpose of feedback control.

We first define the coordinates of the system. As shown in Figure 4.2, we assume the camera frame and the quadrotor frame are the same at frame $\{C\}$. Simultaneously, the world frame $\{W\}$ has only translation with frame $\{C\}$. Thus, there is no rotation between them. Additionally, to validate the capability of the system, an implement of a motion capture system is needed. There is an RBG camera with frame $\{M\}$ which has the rotation of -90 degrees and opposite z axis.

doi:10.6342/NTU202404031

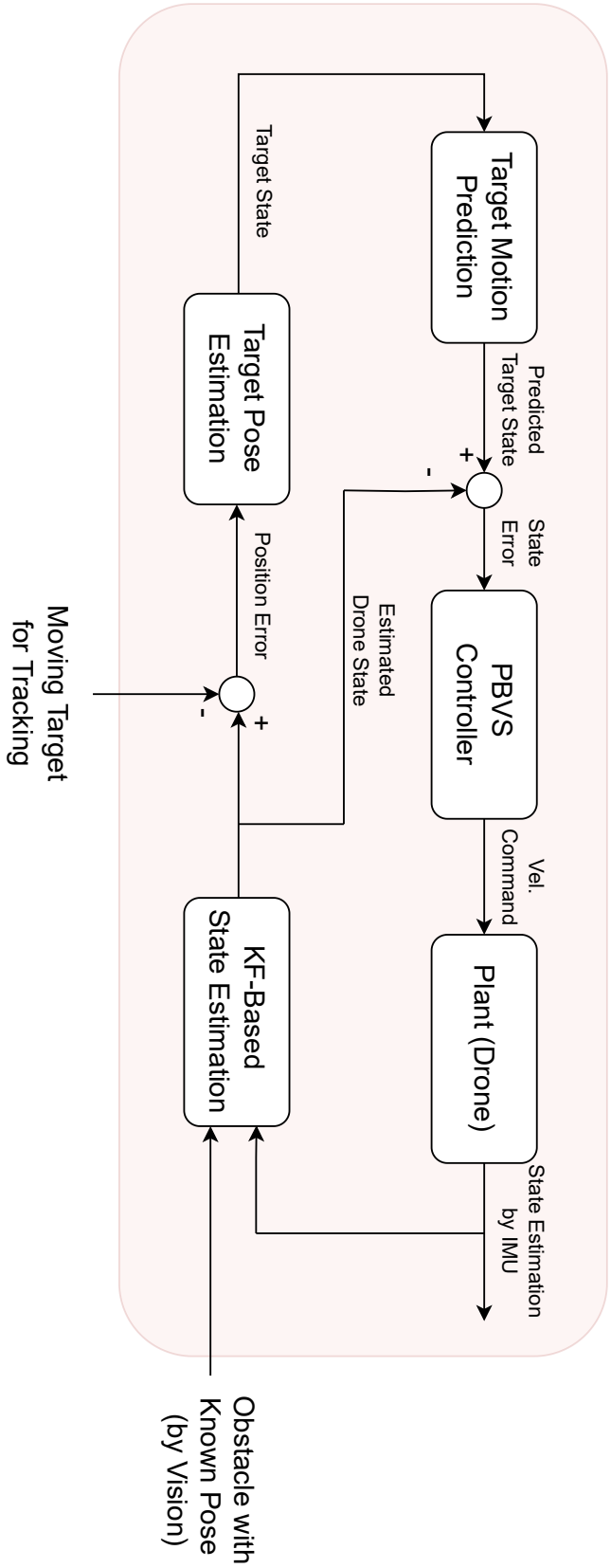


Figure 4.1: System overview.

doi:10.6342/NTU202404031

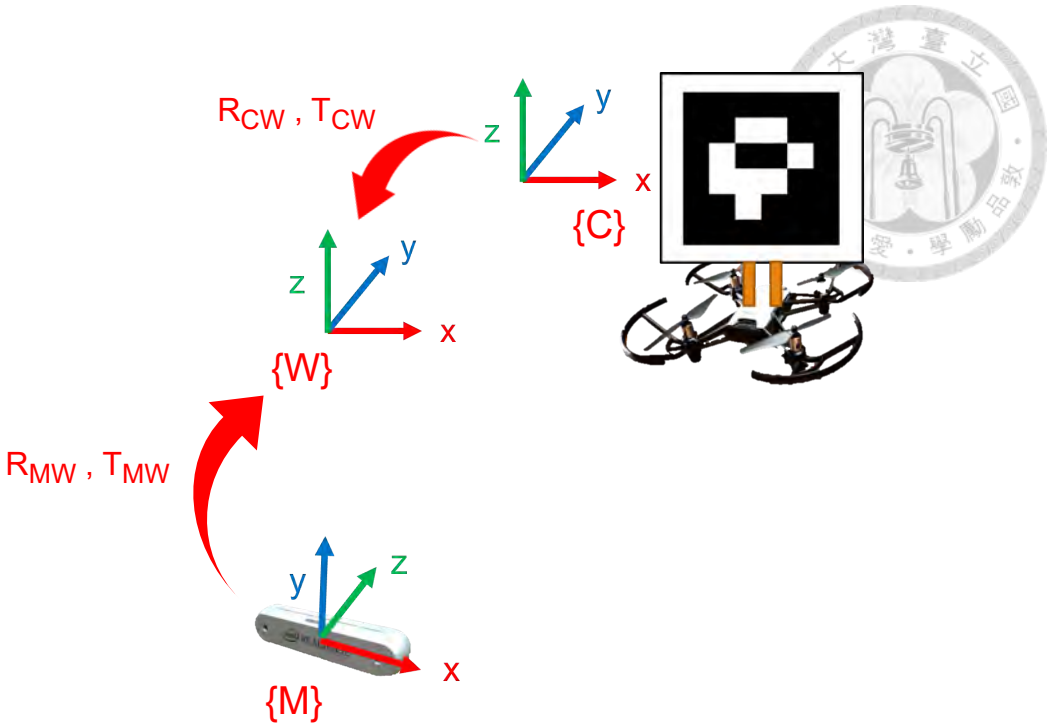


Figure 4.2: Relationship of the coordinates.

The overall coordinate transformation of the experiment is listed below.

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = R_{CW} \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} + T_{CW}, \tag{4.1}$$

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = R_{MW} \begin{bmatrix} X_M \\ Y_M \\ Z_M \end{bmatrix} + T_{MW}, \tag{4.2}$$

where R_{CW} and R_{MW} can be defined as

$$R_{CW} = I_3, \tag{4.3}$$

doi:10.6342/NTU202404031

where

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.4}$$

and

$$R_{MW} = R_{z=-z} \times R_x(-90^\circ) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{4.5}$$

where

$$R_x(-90^\circ) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-90^\circ) & -\sin(-90^\circ) \\ 0 & \sin(-90^\circ) & \cos(-90^\circ) \end{bmatrix}, \; R_{z=-z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{4.6}$$

We will use this coordinate system in the experiment validations.

4.2 Proposed KF-Based Localization with Landmarks

As mentioned in chapter 2 and Appendix G, the onboard IMU of the quadrotor Tello has a severe error accumulation, so we need to improve the problem with downward vision using the onboard camera. The method of downward vision processing is shown in Appendix A. This section will focus on the strategies to make the localization more accurate.

4.2.1 Workflow of State Estimation According to Landmarks with Vision

Localization is a vital technique in PBVS control. We use the Kalman filter with a constant speed model to do the localization using the landmarks with known positions in

doi:10.6342/NTU202404031

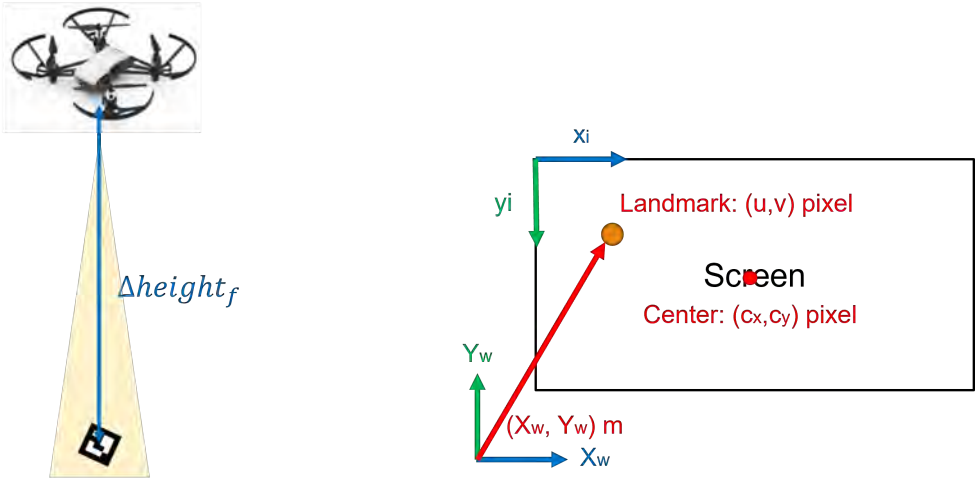
the world frame $\{W\}$. The linear model of the Kalman filter can be defined as

$$A(k) = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.7}$$

$$H(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.8}$$

where $dt = 0.1(s)$ is the runtime of a loop in the algorithm.

First, according to [68: Wu 2019], we can estimate the relative position $(\Delta X_W, \Delta Y_W)$ between the target and itself in the world frame according to the relative height $\Delta height_f$ and pixel value in the image, as shown in Figure 4.3. The equation to estimate its position



(a) Representation of the relative height. **(b) Relative position estimation in the 2D image.**

Figure 4.3: Schematic scenario in state estimation according to the landmark.

doi:10.6342/NTU202404031

can be interpreted as

$$\Delta X_W = -(v - c_x) \times \Delta\text{height}_f \times \frac{h_{\text{img}}}{\alpha_v}, \tag{4.9}$$

$$\Delta Y_W = (u - c_y) \times \Delta\text{height}_f \times \frac{w_{\text{img}}}{\alpha_u}, \tag{4.10}$$

where (u, v) is the pixel value of the landmark center in the image, (c_x, c_y) is the center of the image in pixel, (h_{img}, w_{img}) are the height and width of the image in pixel, and $(\alpha_u, \alpha_v) = (1.04, 0.77)m$ is the FOV in x_i and y_i direction at the height of 1 meter.

Thus, the position of the quadrotor in the world frame $\{W\}$ is

$$X_{W,quad} = X_{W,landmark} + \Delta X_W, \tag{4.11}$$

$$Y_{W,quad} = Y_{W,landmark} + \Delta Y_W, \tag{4.12}$$

where $(X_{W,landmark}, Y_{W,landmark})$ is the position of the landmark in the world frame $\{W\}$ with known position.

4.2.2 KF-Based State Estimation According to Landmarks

Recalling the workflow in Figure 3.4, when the quadrotor detect the target, it will use the landmark information directly. If there is no landmark, it will take the last position information as the initial state input of the KF and run the KF algorithm to update the position. If the quadrotor detect the landmark again, it will use the landmark information again. The workflow is shown in Figure 4.4. The simulation and experiment result will be shown in Appendix I.

doi:10.6342/NTU202404031

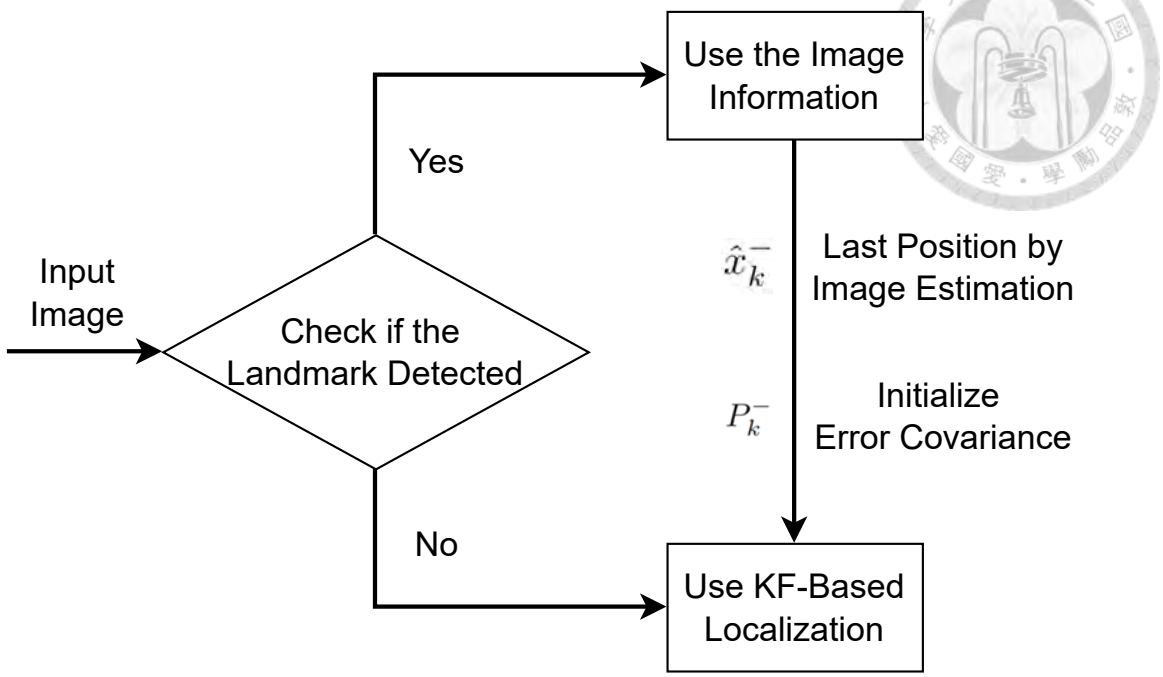


Figure 4.4: The workflow of proposed KF-based localization framework.

4.3 Trajectory-Based Motion Prediction of the Target

The purpose why we need to obtain the trajectory is that because the vision will not be always stable. That is, the target is not always detected through the image. As a result, trajectory is an important essence for tracking. Furthermore, motion prediction is also a vital method. If we track the target without prediction, the drone will lose some target information unpredictably. In this section, we will introduce the strategies to obtain the trajectories of the target in x and y directions using onboard vision of the quadrotors with more accurate KF-based localization.

4.3.1 Target State Estimation with Downward Vision

Similar to 4.2.1, the representation of the equation to estimate relative position with

doi:10.6342/NTU202404031

the target can be interpreted as

$$\Delta X_{W,t} = (v_t - c_x) \times \Delta\text{height}_{f,t} \times \frac{h_{img}}{\alpha_v}, \tag{4.13}$$

$$\Delta Y_{W,t} = -(u_t - c_y) \times \Delta\text{height}_{f,t} \times \frac{w_{img}}{\alpha_u}, \tag{4.14}$$

where the notations with subscript t denote the relative information with the target. We can obtain the target position as

$$X_{W,t} = X_{W,quad} + \Delta X_{W,t}, \tag{4.15}$$

$$Y_{W,t} = Y_{W,quad} + \Delta Y_{W,t}, \tag{4.16}$$

for estimating the position of the target. Different from section 4.2.1 to estimate the drone position itself using the landmark, the direction of these equations is opposite. In this section, we estimate the quadrotors themselves according to the external object, but in this section, it estimates the external object using its information.

4.3.2 Trajectory Function Representation

After we have the positions of the target within time, we can fit the target trajectories in X_W and Y_W directions. It can be defined as [21: Thomas *et al.* 2017]

$$g(t) = \sum_{k=0}^{n} c_k b_k(t). \tag{4.17}$$

In the representation of the element in the vector, it can be formed as

$$g_i(t) = {c_i}^T b(t), \tag{4.18}$$

where the target trajectory $g(t)$ is the function of time, c_k is the coefficients of the poly-

doi:10.6342/NTU202404031

nomial trajectory, and $b(t)$ can be the standard power basis. That is,

$$b(t) = \begin{bmatrix} 1 & t & t^2 & \dots & t^n \end{bmatrix}^T. \tag{4.19}$$

4.3.3 Target Motion Prediction with Curve Fitting

For the motion prediction of the target, we first estimate the position of the target within time and obtain the trajectories of the target which are functions of time.

When the target is detected, we record its position as well as the time. As shown in Figure 4.5, we collect the most recent points to do curve fitting. If the number of points is equal to 10, the algorithm will discard the first data to perform curve fitting. That is, we maintain a record of the most recent 10 sets of data to avoid the effect of the previous data. After discarding the first point of the set, we check whether the new point be added or not. If there is new point detected in the next loop, the algorithm would utilize these 10 points to perform curve fitting. Else, the algorithm would utilize the remaining 9 points to perform curve fitting until the new point detected. As for the curve fitting, we can formulate it into an unconstrained convex optimization (QP) problem.

$$\min F(c) = \sum_{k=1}^{N} (c^T X_{W,k} - Y_{W,k})^2,$$

where $F(c)$ is the quadratic function we want to optimize, $X_{W,k}$ are the X positions of the points, $Y_{W,k}$ are the Y positions of the points in the world frame $\{W\}$, and coefficients of the polynomial we want to fit can be defined as

$$c = argmin F(c). \tag{4.20}$$

doi:10.6342/NTU202404031

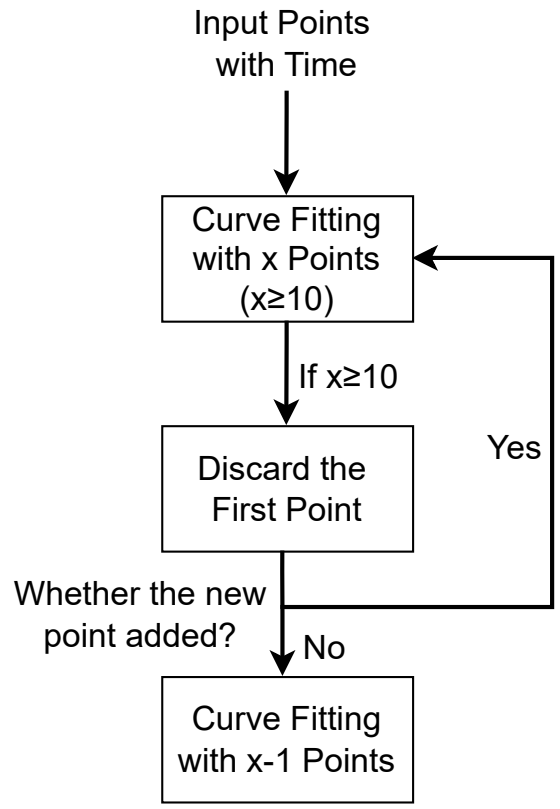


Figure 4.5: The workflow of sliding-window points selection for curve fitting.

Here $c = [c_0\ c_1\ c_2 \ldots c_n]$ is the vector consisting of all the elements of c_k in Equation 4.17. In this paper, we utilize first-degree polynomials to fit the polynomial trajectories. That is, $c \in \mathbf{R}^2$ and $g(t)$ is a linear function.

4.4 Control Strategies of the Multi-Quadrotor System

4.4.1 Planar Consensus-Based Formation Tracking Control for the Multi-Agent System

In this section, we will introduce the control law in X_W and Y_W directions to make the team of quadrotors to track the ground target with formation.

4.4.1.1 Reference State Extraction

According the curve fitting algorithm, we can predict the motion of the target for each quadrotors. As shown in Figure 4.6, each quadrotor plans a trajectory and predicts a

doi:10.6342/NTU202404031

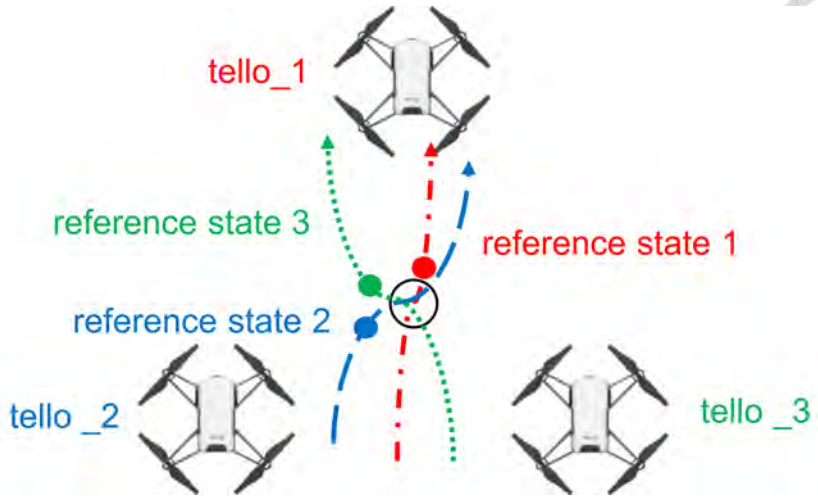


Figure 4.6: The schematic scenario for individual trjaectory of prediction.

reference state including position and velocity terms at current time.

$$r_{i,rs} = [r_{i,r}, \dot{r}_{i,r}], \tag{4.21}$$

where $r_{i,r}$ is the position of the reference from trajectory-based motion prediction, which is also the function value of $g(t)$ at current time and $\dot{r}_{i,r}$ is the velocity of the reference, which is also the function value of $\dot{g}(t)$ at current time.

Nonetheless, according to the motion prediction algorithm, if the quadrotor cannot obtain the new target information for a long time, it will utilize the last trajectory to predict the current state of the target, which will make the information of the target inaccurate. Thus, we take the threshold of the predicted position to obtain acceptable state. The equation for selecting the acceptable reference states can be formulated as

$$r_{i,rs,accep} = \begin{cases} r_{i,rs}, & \text{if } |r_{i,r}| \leq thresh \\ None, & \text{otherwise.} \end{cases} \tag{4.22}$$

doi:10.6342/NTU202404031

As a result, the overall unique reference state can be formulated as

$$r_{overall,rs} = mean(r_{i,rs,accep}) \quad \text{for} \quad r_{i,rs,accep} \neq None, \tag{4.23}$$

where $r_{overall,rs} = [p_{ref}, \dot{p}_{ref}]$ include position and velocity terms.

4.4.1.2 Planar Consensus-Based Formation Tracking Control

Next, because for the current package we use, we need to design the velocity command for the quadrotors. We consider a single-integrator dynamic of a quadrotor

$$\dot{r}_i = u_i, \quad i = 1, ..., n, \tag{4.24}$$

where $n = 3$ is the number of agents, and $u_i \in \mathbf{R}^2$ is the control input of the i^{th} agent. In our control scheme referring to [69: Ren and Beard 2007], the consensus-based velocity control input can be designed as

$$u_i = \dot{r}_i^d - \alpha_i(r_i - r_i^d) - \sum_{j=1}^{n}[(r_i - r_i^d) - (r_j - r_j^d)], \tag{4.25}$$

where $\alpha_i > 0$ is a scalar, r_i^d is the desired state of the i^{th} agent. Here the states r_i and r_i^d are the position information with the velocity control input u_i. That is, $r_i = [X_{Wi}, Y_{Wi}]$ and $r_i^d = [X_{Wi}^d, Y_{Wi}^d] = p_{ref} + d_i$ where d_i is the distance of position offset, as shown in Figure 4.7. Furthermore, the desired velocity $\dot{r}_i^d = [\dot{X}_{Wi}^d, \dot{Y}_{Wi}^d] = \dot{p}_{ref}$. The stability analysis of the formation controller will be presented in Appendix M.

Nonetheless, due to the hardware limitations such as signal transmission and the weights of the ArUco markers on the quadrotors for the motion capture system, collision will sometimes occur among the quadrotors with the design command u_i. As a result, for the designed formation control command, we revise the command velocity u_i and set a

doi:10.6342/NTU202404031

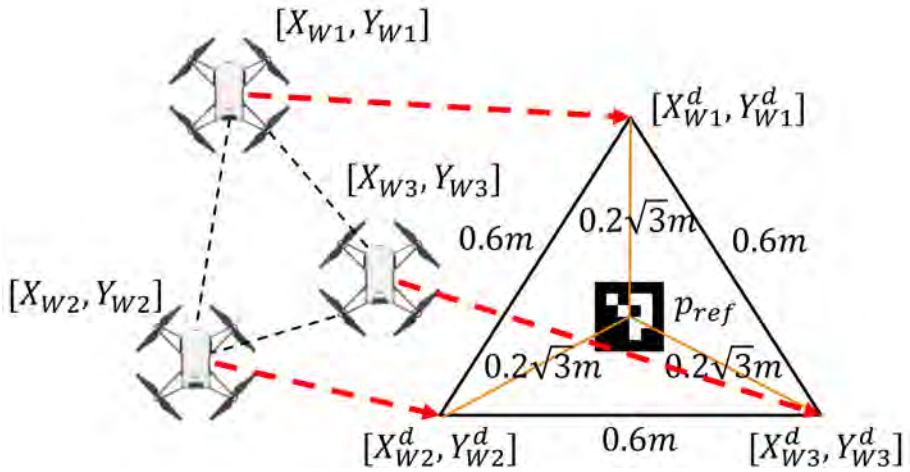


Figure 4.7: The relationship between the current states and desired formation of the agents.

saturation projection for it. That is,

$$\mathbf{u}_{\text{if}} = \begin{cases} \mathbf{u}_i, & \text{if } \|\mathbf{u}_i\| \leq 0.3 \\ 0.3\frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}, & \text{if } \|\mathbf{u}_i\| > 0.3. \end{cases}$$

Here $\mathbf{u}_{\text{if}}$ is the final designed control command (m/s) in the experiment. Note that the command velocity is also larger than the target speed.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Chapter 5

Experiment Results and Validations of the System

In this chapter, we will discuss the multi-agent localization and consensus-based formation tracking controller toward a moving ground target. We also compare some KF-based localization results of the quadrotors with the ground truth, the predicted target position of each drone using curve fitting with estimated data, the averaged unique predicted position of the target with the actual position, the positions of all moving objects including the quadrotors and the target, and executing time interval of the command velocities.

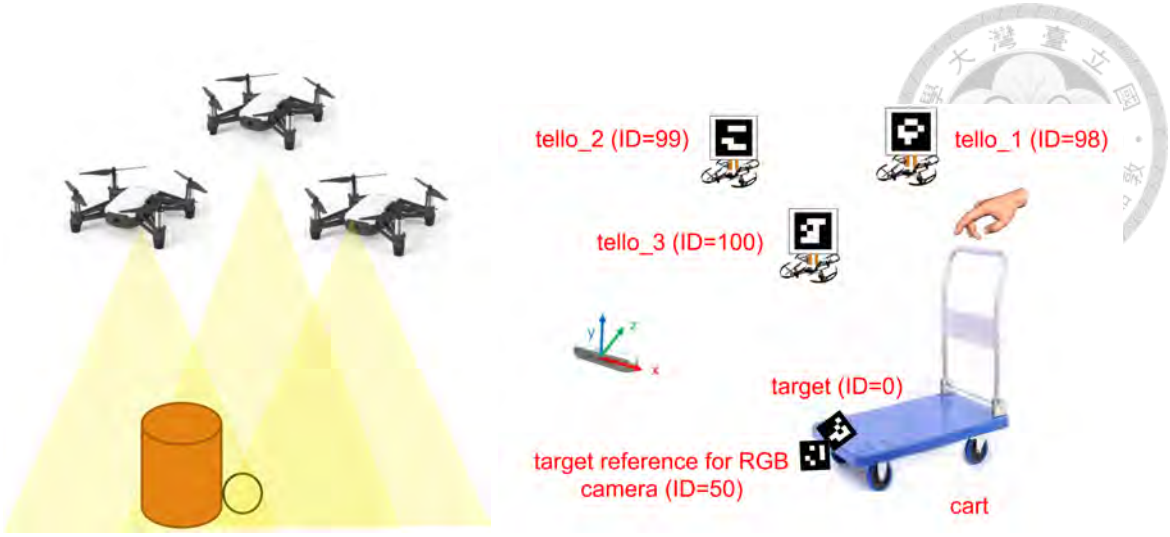
5.1 Experiment Setup

We use three quadrotors as our control plant. The objective is to localize the quadrotors with landmarks and obtain the state of the ground moving target under partially occlusion from the landmarks by the tracking framework with trajectory assistance. The external objects including landmarks and the target are detected by means of the ArUco markers on them. An RGB camera from the third perspective is applied for capturing the motions of quadrotors (ID: 98, 99, 100) and the target reference (ID: 50). The scenario is shown in Figure 5.1.

5.2 Task Overview

For the scenario design, there are 3 non-control tasks and 3x3 control tasks for evaluating the effect of the proposed algorithms, as shown in Table 5.1. The purpose is to

doi:10.6342/NTU202404031



(a) Target scenario. (b) Schematic target scenario.

Figure 5.1: Experiment scenario.

demonstrate that control of the group will make them detect the target even when the target is moving. We divide the cases into three cases according to the obstacles to imitate the conditions of simple, normal and complex scenario in the cities. Furthermore, the velocity of the target is also a vital criteria to evaluate the capability of the tracking algorithms because there might be some difference for target surveillance efficiency. We will also show the results in the later sections. The velocities of the target in these cases are

1. Stationary: $v = 0$ (m/s) along Y_W direction
2. Slow: $v = 0.03 \sim 0.07$ (m/s) along Y_W direction
3. Fast: $v = 0.15 \sim 0.30$ (m/s) along Y_W direction.

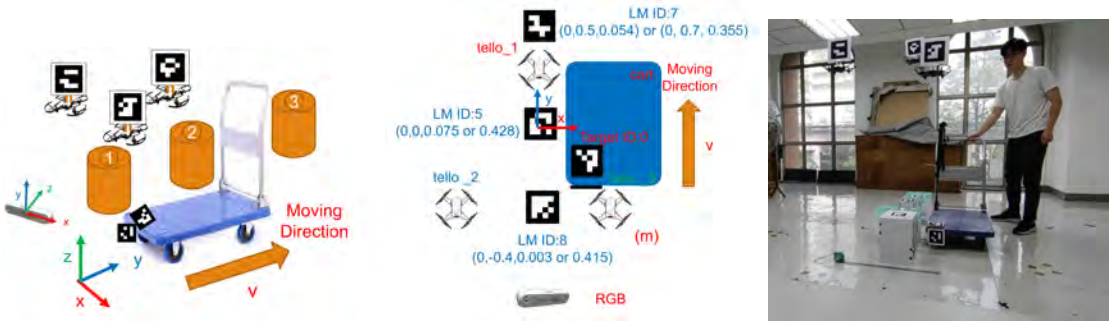
Table 5.1: Case study of the experiments.

	No control	Control, no obs.	Control, 1 obs.	Control, 3 obs.
Stationary	0-1	1-1	2-1	3-1
Slow	0-2	1-2	2-2	3-2
Fast	0-3	1-3	2-3	3-3

In Figure 5.2, it shows the experiment setup and Figure 5.3 shows the real object in Figure 5.2(a). For tasks 0-1 $\sim$ 0-3 and 1-1 $\sim$ 1-3, the 3D positions of the landmarks with IDs 7,

doi:10.6342/NTU202404031

5, and 8 are $(0, 0.5, 0.054)$, $(0, 0, 0.075)$, and $(0, -0.4, 0.003)$ (m) respectively. For tasks 2-1 $\sim$ 2-3, the 3D positions of the landmarks with IDs 7, 5, and 8 are $(0, 0.5, 0.054)$, $(0, 0, 0.075)$, and $(0, -0.4, 0.415)$ (m) respectively. For tasks 3-1 $\sim$ 3-3, the 3D positions of the landmarks with IDs 7, 5, and 8 are $(0, 0.7, 0.355)$, $(0, 0, 0.428)$, and $(0, -0.4, 0.415)$ (m) respectively. Note that all the positions are defined in frame $\{W\}$ Simultaneously, the height of the target (ID=0) is 0.2 (m) for all cases and the height of the drones can be obtained by IMU information. As a result, in Figure 5.3, the objects in the red boxes are the obstacles higher than the target, which may result in some occlusions of the target from the perspective of the drones. The process will go through 4 steps, including



(a) Experiment setup from the side view. **(b) Experiment setup from the top view.** **(c) Experiment setup in real scenario.**

Figure 5.2: Experiment setup.

1. Tello motor on (10 sec)
2. Takeoff and hovering (5 sec)
3. Controlling the planar motion in X_W and Y_W (30 sec)
4. Hovering (5 sec)
5. Landing.

Each process can be identified in the figures of this chapter.

doi:10.6342/NTU202404031

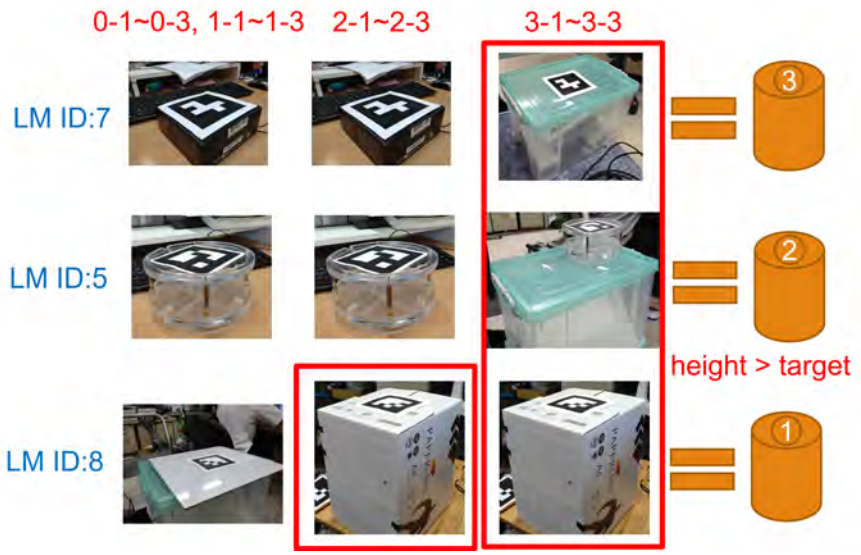


Figure 5.3: The real objects as the landmarks and obstacles in Figure 5.2(a).

5.3 Quadrotor Positions by Onboard Localization/RGB Camera Estimation within Time

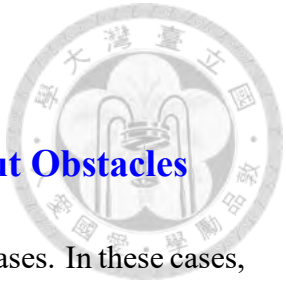
In this section, we will demonstrate accuracy of the onboard localization framework using KF-based localization with the landmarks and IMU velocity measurement. Furthermore, we will compare the localization result with the result of the ground truth with RGB camera motion capture system using the evaluation of the error, where the error is defined as

$$error(\%) = \frac{position(onboard) - position(RGB)}{height} \times 100\%. \tag{5.1}$$

Note that in the first two graphs for each case, the curves formed by (red) circles, (blue) crosses and (green) triangles are the ground truth of tello_1, tello_2 and tello_3 respectively, and the (red) dashed, (blue) dotted and (green) dashed-dotted curves are the position result of onboard localization using the fusion of landmark and IMU information with KF. Furthermore, for the last two graphs in each case, the the curves formed by (red) circles, (blue) crosses and (green) triangles are the errors of tello_1, tello_2 and tello_3

doi:10.6342/NTU202404031

respectively.



5.3.1 Results of 0-1 ∼ 0-3, Non-Control Cases, without Obstacles

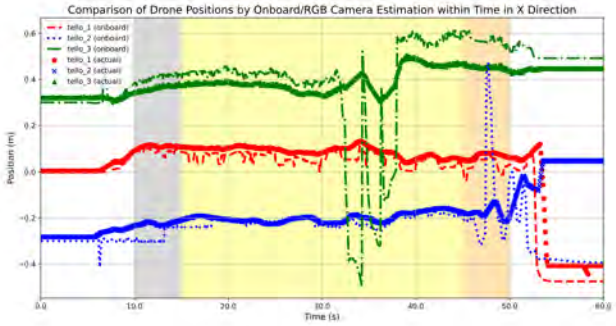
In this section, we will demonstrate the result of the non-control cases. In these cases, the result will show that without control of the agents, they will lose the information of the target after certain time period. The results of non-control cases are shown in Figure 5.4, Figure 5.5 and Figure 5.6.

For the case 0-1 referring to Figure 5.4, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.4(a) and Figure 5.4(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.4(c) and Figure 5.4(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error is in ± 10 % range in X_W direction and in ± 15 % range in Y_W direction.

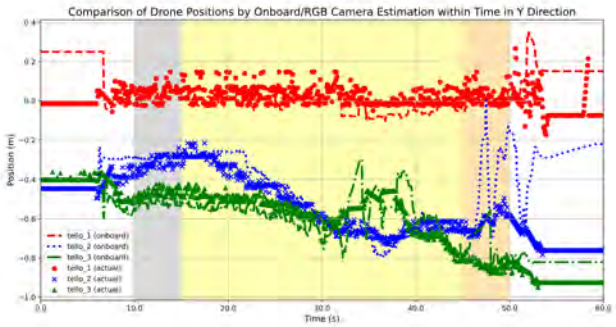
For the case 0-2 referring to Figure 5.5, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.5(a) and Figure 5.5(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.5(c) and Figure 5.5(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in -5% to 10% range in X_W direction and in -15% to 10% range in Y_W direction, which has not much difference between case 0-1.

For the case 0-3 referring to Figure 5.6, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the

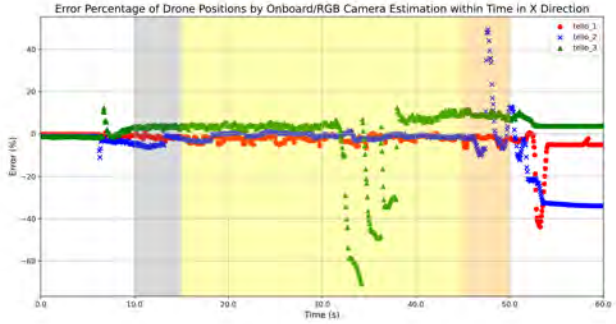
doi:10.6342/NTU202404031



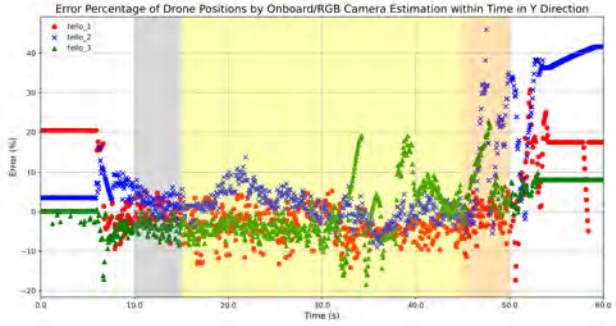
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



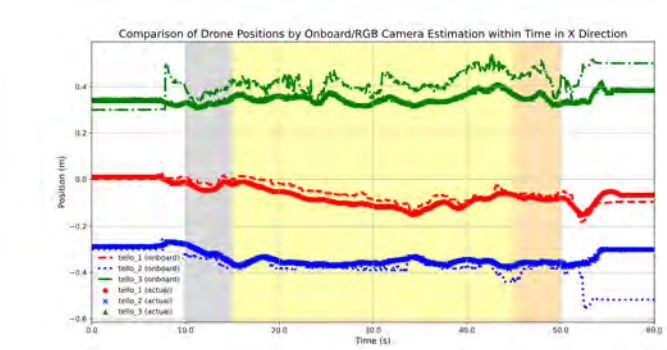
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



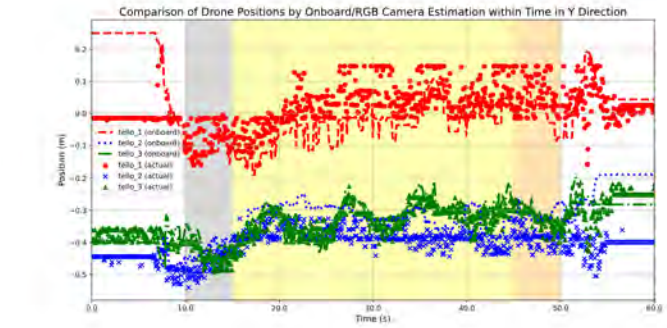
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.4: The comparison of drone positions by onboard/RGB camera estimation within time of case 0-1.

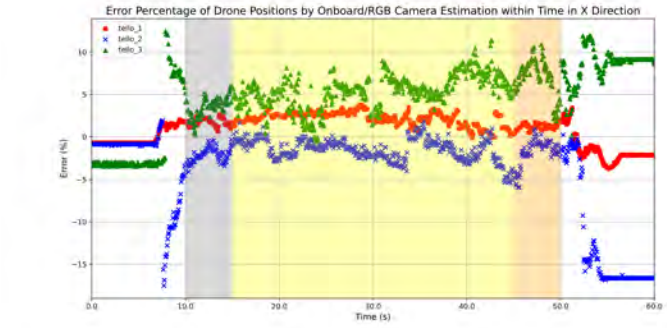
doi:10.6342/NTU202404031



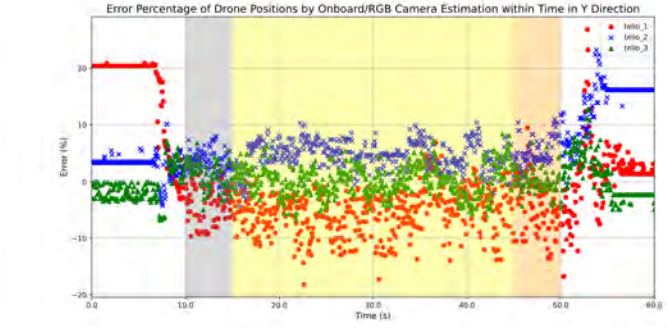
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



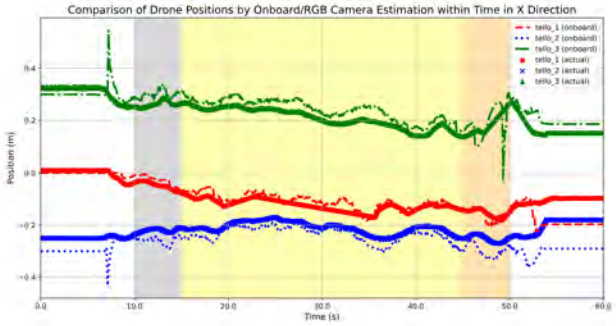
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



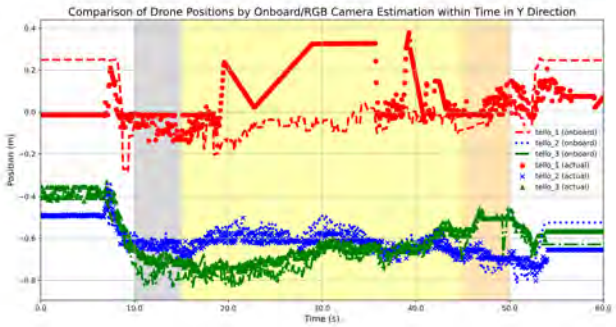
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.5: The comparison of drone positions by onboard/RGB camera estimation within time of case 0-2.

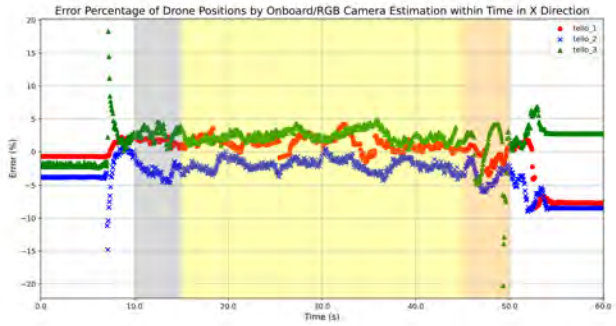
doi:10.6342/NTU202404031



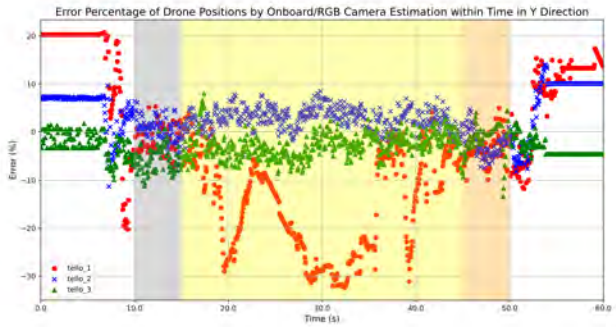
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.6: The comparison of drone positions by onboard/RGB camera estimation within time of case 0-3.

doi:10.6342/NTU202404031

ground truth. In Figure 5.6(a) and Figure 5.6(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.6(c) and Figure 5.6(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in ± 5 % range in X_W direction and in -30% to 10% range but mostly ± 10 % in Y_W direction, which has not much difference between case 0-1 and 0-2.

In summary, for case 0-1 $\sim$ 0-3, we can observe that because there is no active motion of the drones, the motions are similar among the cases, and the errors are about $\pm 5 \sim \pm 15$ % for most cases.

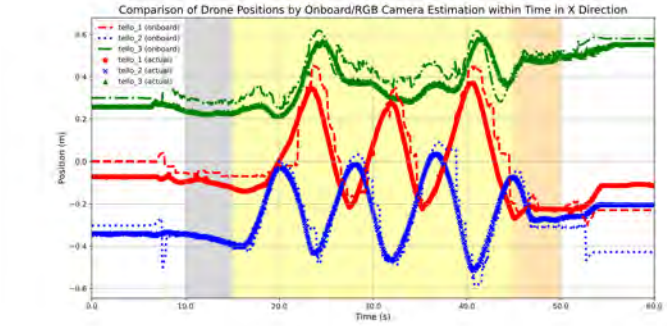
5.3.2 Results of 1-1 $\sim$ 1-3, Control Cases, without Obstacles

In this section, we demonstrate the onboard localization according to ground landmarks and the height of ground landmarks are lower than the target so that the landmarks will not occlude the target as in the empty scenario. The results the control cases of 1-1, 1-2 and 1-3 from stationary to fast speed of the target without obstacles are shown in Figure 5.7, Figure 5.8 and Figure 5.9.

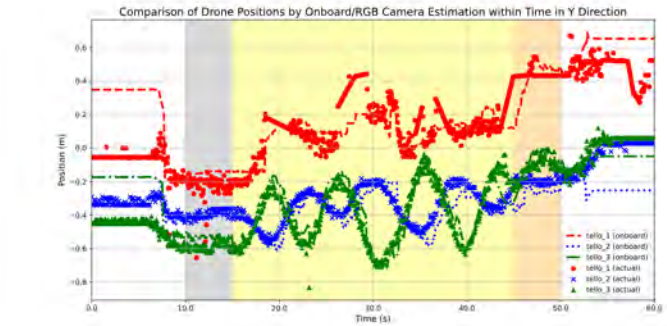
For the case 1-1 referring to Figure 5.7, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.7(a) and Figure 5.7(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.7(c) and Figure 5.7(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in -10% to 15% range in X_W direction and in -20% to 10% range in Y_W direction.

For the case 1-2 referring to Figure 5.8, there is the comparison of the onboard local-

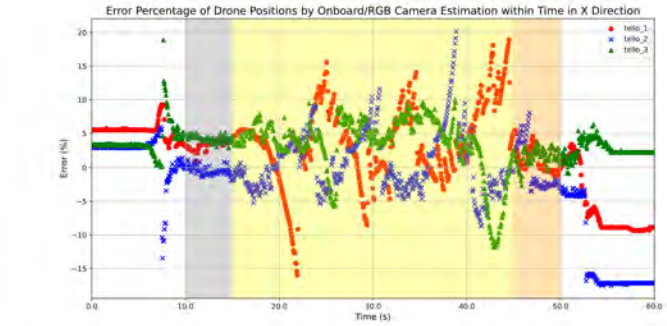
doi:10.6342/NTU202404031



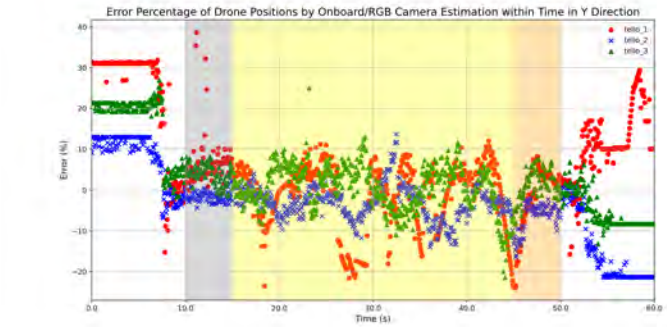
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



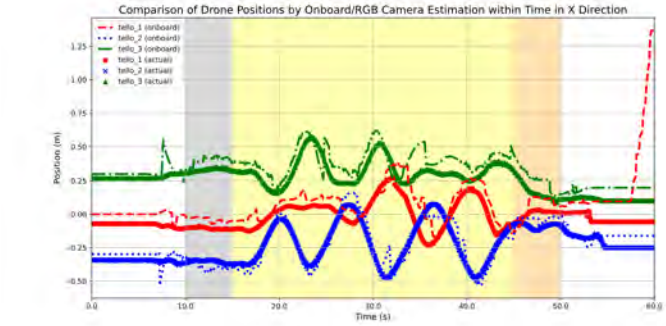
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



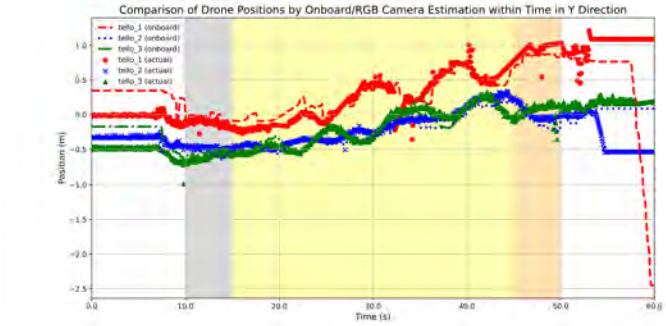
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.7: The comparison of drone positions by onboard/RGB camera estimation within time of case 1-1.

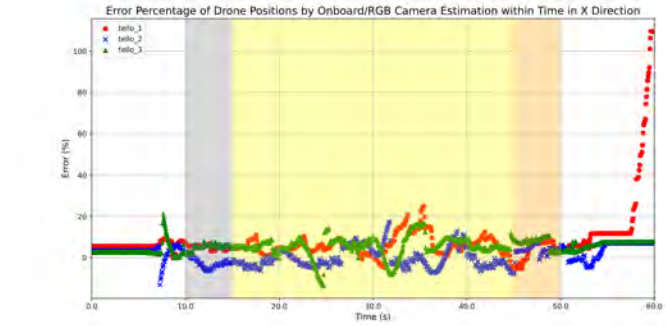
doi:10.6342/NTU202404031



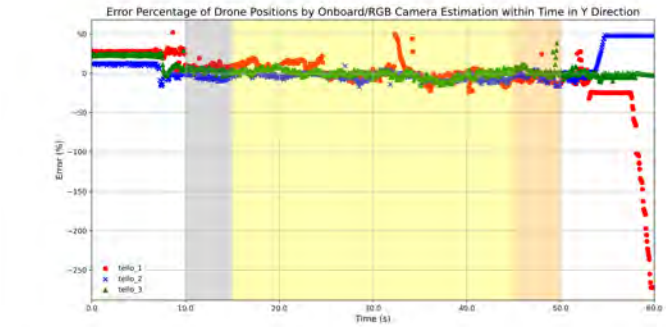
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.8: The comparison of drone positions by onboard/RGB camera estimation within time of case 1-2.

doi:10.6342/NTU202404031

ization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.8(a) and Figure 5.8(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.8(c) and Figure 5.8(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in -10% to 20% range in X_W direction and in $\pm$20 % range in Y_W direction.

For the case 1-3 referring to Figure 5.9, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.9(a) and Figure 5.9(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.9(c) and Figure 5.9(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in $\pm$10 % range in X_W direction and in -20% to 10% range in Y_W direction.

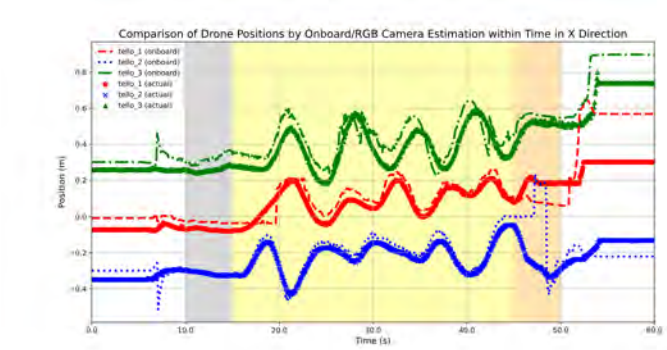
We can observe that although there is nonzero formation control speed, the accuracy of onboard localization is not such obvious. In this case, for the cases with control and without obstacles (1-1 $\sim$ 1-3), there is not much difference among these cases.

5.3.3 Results of 2-1 $\sim$ 2-3, Control Cases, with an Obstacle

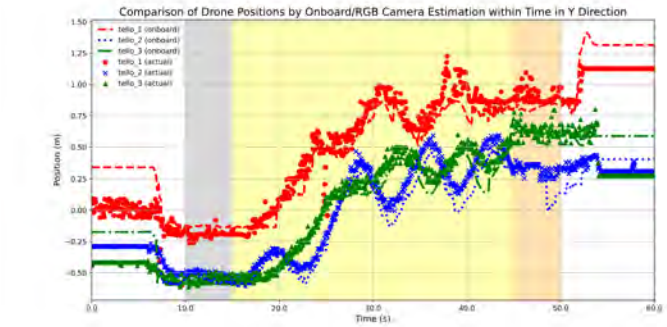
The objective of this section is to imitate the scenario with an obstacle which will sometimes occlude the target. In these cases, the flight of the quadrotors will be more unstable than the cases without obstacles (1-1 $\sim$ 1-3). The onboard localization accuracy results of the control cases of 2-1, 2-2 and 2-3 with an obstacle are shown in Figure 5.10, Figure 5.11 and Figure 5.12.

For the case 2-1 referring to Figure 5.10, there is the comparison of the onboard

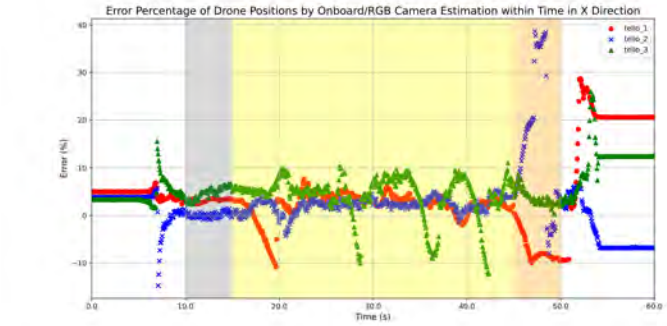
doi:10.6342/NTU202404031



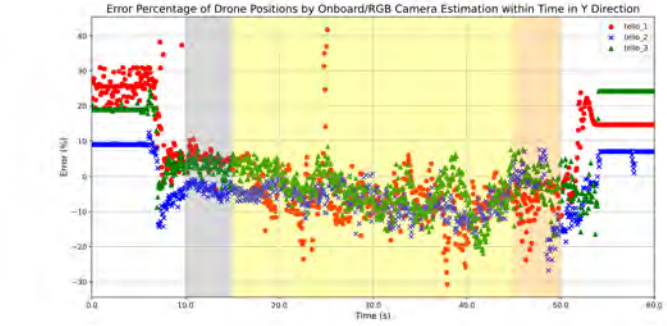
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



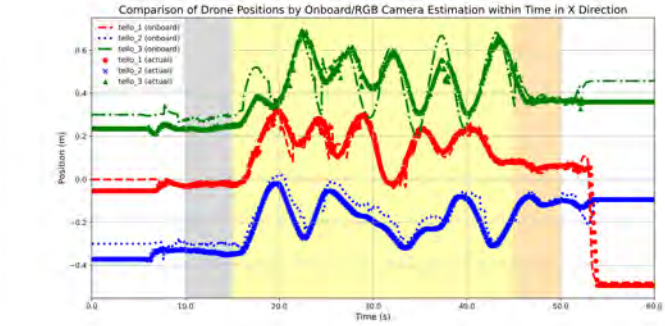
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



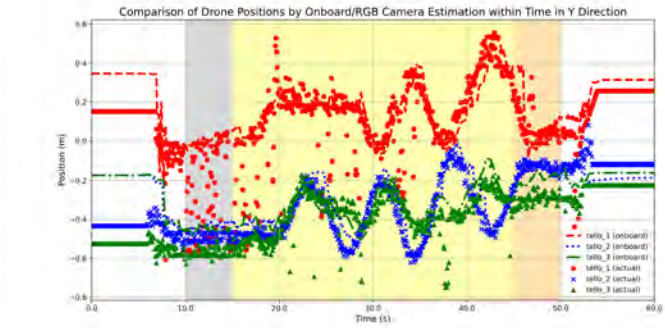
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.9: The comparison of drone positions by onboard/RGB camera estimation within time of case 1-3.

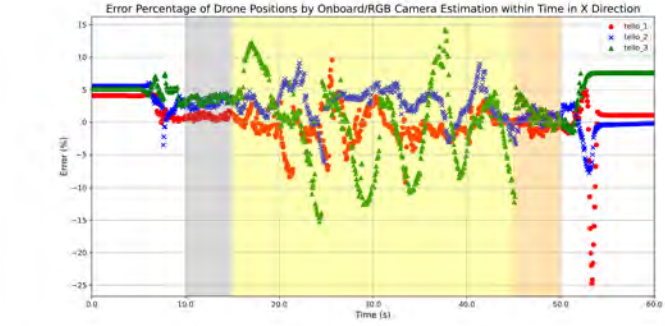
doi:10.6342/NTU202404031



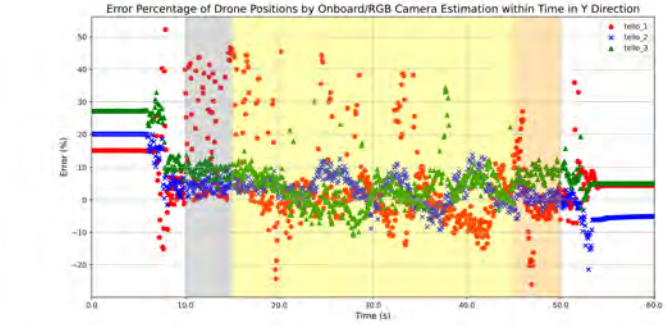
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.10: The comparison of drone positions by onboard/RGB camera estimation within time of case 2-1.

doi:10.6342/NTU202404031

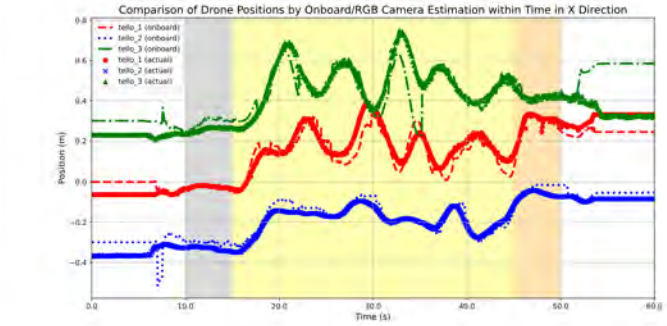
localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.10(a) and Figure 5.10(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.10(c) and Figure 5.10(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in ± 15 % range in X_W direction and in ± 10 % range in Y_W direction.

For the case 2-2 referring to Figure 5.11, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.11(a) and Figure 5.11(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.11(c) and Figure 5.11(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in -20% to 10% range in X_W direction and in -20% to 15% range in Y_W direction.

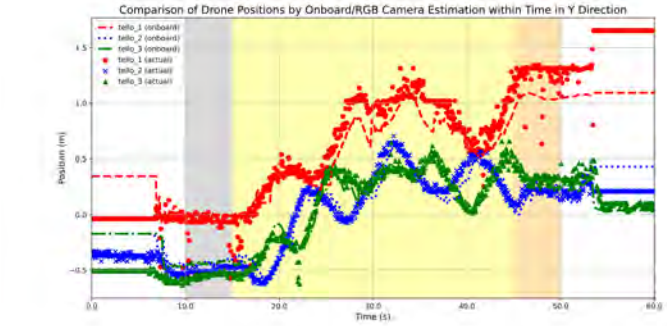
For the case 2-3 referring to Figure 5.12, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.12(a) and Figure 5.12(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.12(c) and Figure 5.12(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in -20% to 15% range in X_W direction and in ± 20 % range in Y_W direction.

Obviously, for case 2-1$\sim$ 2-3, we can observe that if there is nonzero formation control speed, the accuracy of onboard localization decreases, which means the error increases

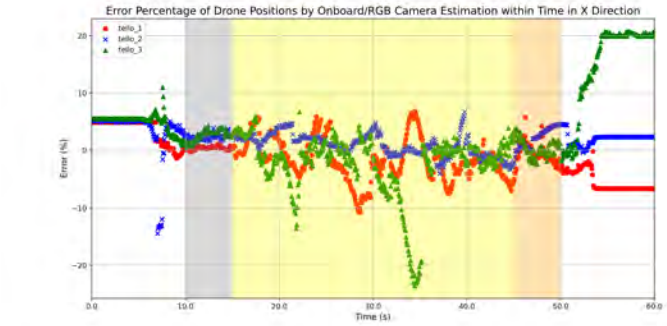
doi:10.6342/NTU202404031



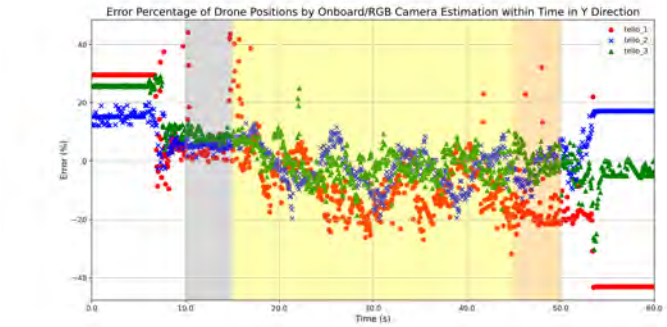
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



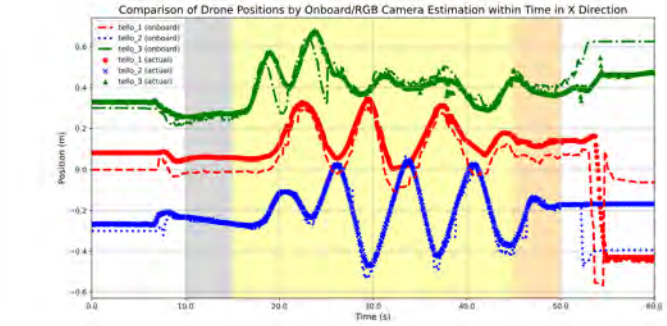
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



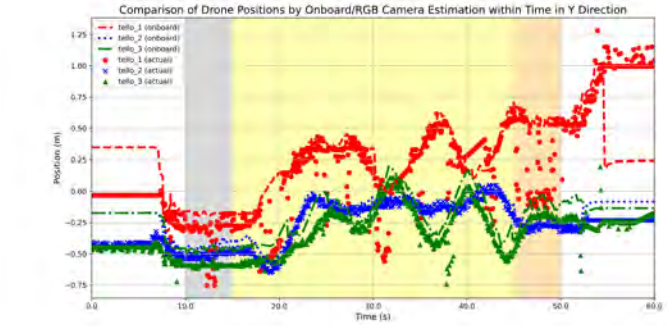
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.11: The comparison of drone positions by onboard/RGB camera estimation within time of case 2-2.

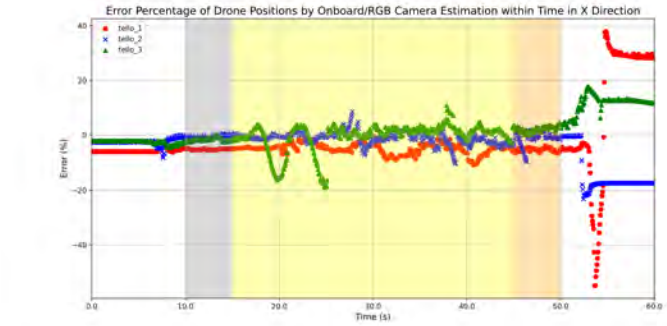
doi:10.6342/NTU202404031



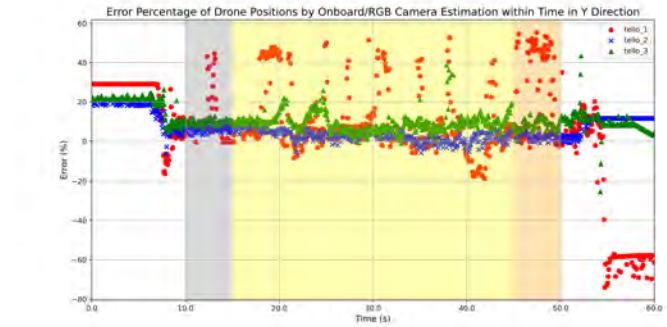
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.12: The comparison of drone positions by onboard/RGB camera estimation within time of case 2-3.

doi:10.6342/NTU202404031

because of more unstable flight.

5.3.4 Results of 3-1 $\sim$ 3-3, Control Cases, with three Obstacles

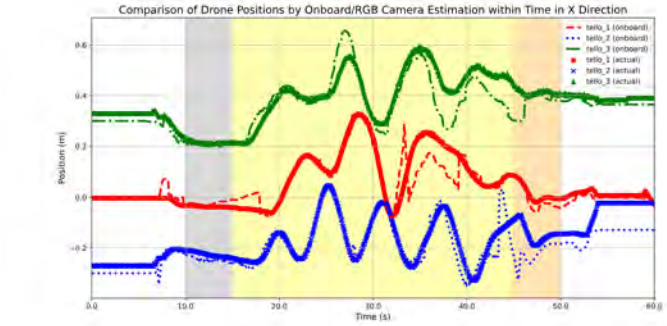
In these cases of 3-1 $\sim$ 3-3 for imitating the complex scenario with multiple obstacles, the flight of the quadrotors will be more unstable than the previous cases including 1-1 $\sim$ 1-3 and 2-1 $\sim$ 2-3 due to the less information of the target in FOV. The results of the control cases of 3-1, 3-2 and 3-3 with three obstacles are shown in Figure 5.13, Figure 5.14 and Figure 5.15.

For the case 3-1 referring to Figure 5.13, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.13(a) and Figure 5.13(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.13(c) and Figure 5.13(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in $\pm$15 % range in X_W direction and in $\pm$20 % range in Y_W direction.

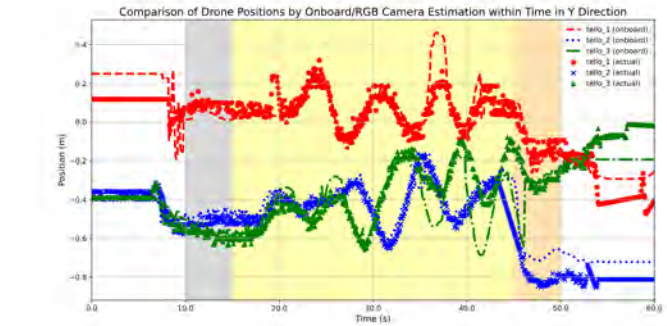
For the case 3-2 referring to Figure 5.13, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which is the ground truth. In Figure 5.14(a) and Figure 5.14(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.14(c) and Figure 5.14(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in $\pm$20 % range in X_W direction and in -30% to 20% range in Y_W direction.

For the case 3-3 referring to Figure 5.15, there is the comparison of the onboard localization with KF algorithm and the position result estimated by RGB camera, which

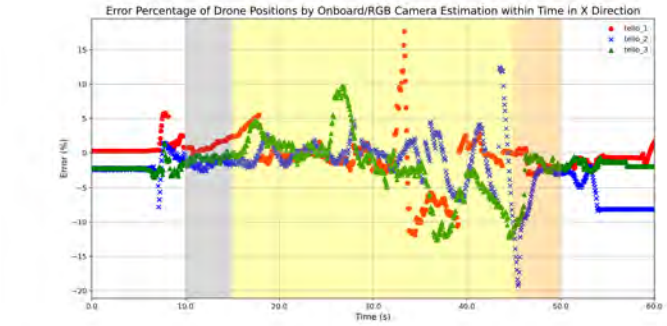
doi:10.6342/NTU202404031



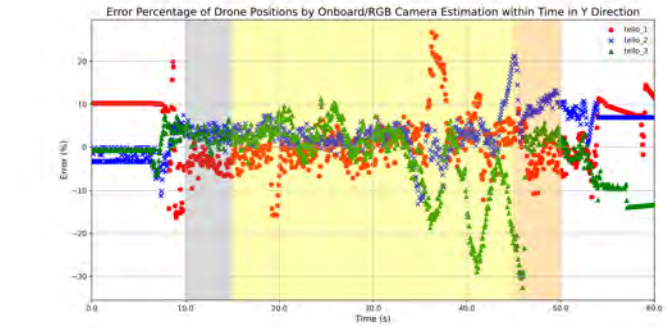
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



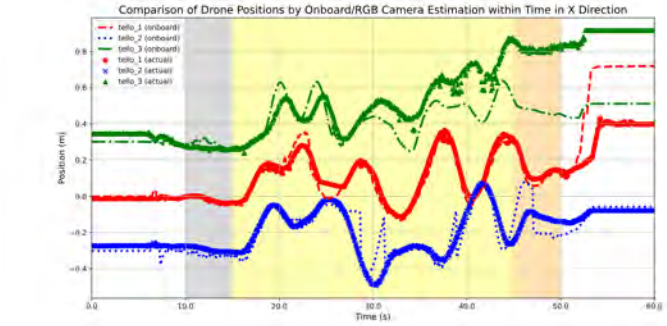
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



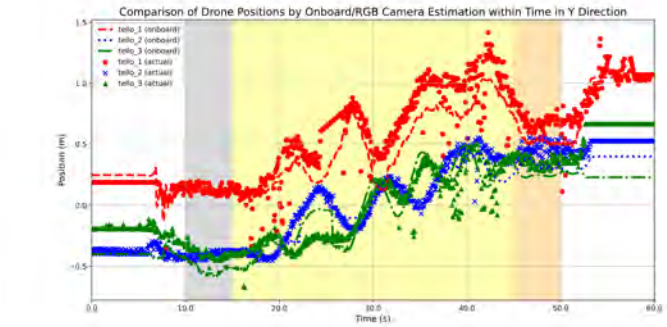
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.13: The comparison of drone positions by onboard/RGB camera estimation within time of case 3-1.

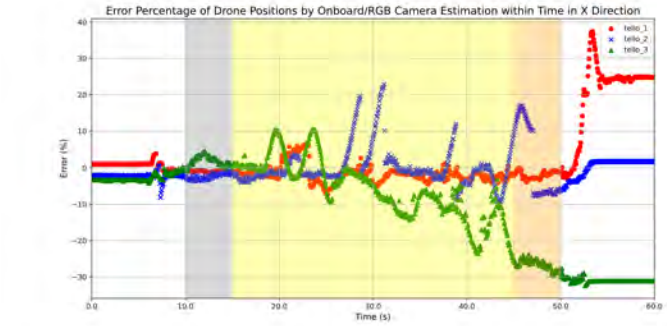
doi:10.6342/NTU202404031



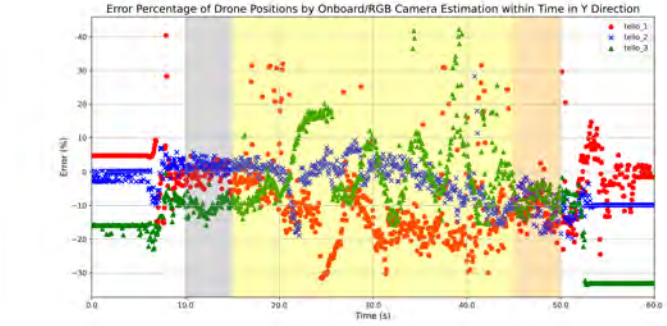
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



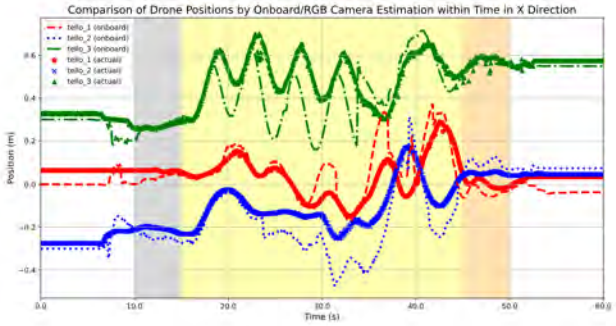
(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



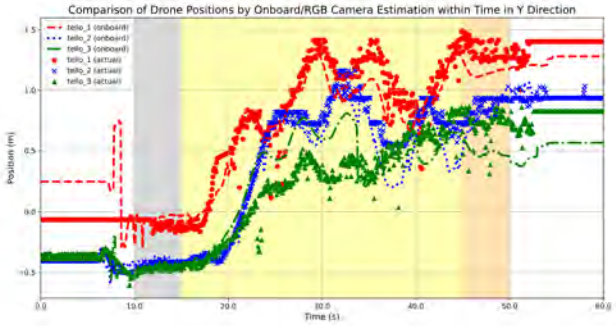
(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.14: The comparison of drone positions by onboard/RGB camera estimation within time of case 3-2.

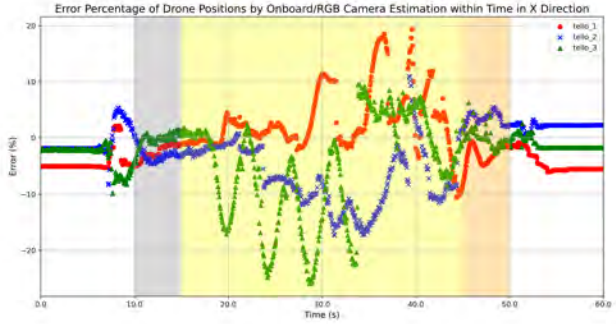
doi:10.6342/NTU202404031



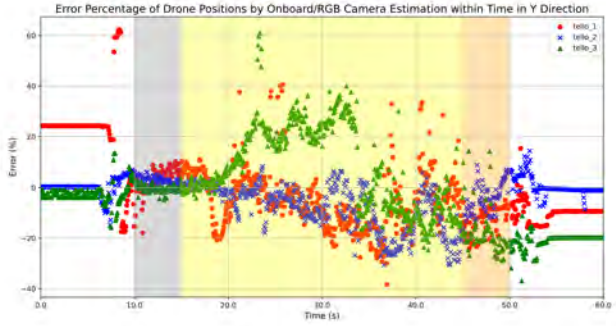
(a) Drone positions by onboard/RGB camera estimation in X_W direction.



(b) Drone positions by onboard/RGB camera estimation in Y_W direction.



(c) Drone position errors by onboard/RGB camera estimation in X_W direction.



(d) Drone position errors by onboard/RGB camera estimation in Y_W direction.

Figure 5.15: The comparison of drone positions by onboard/RGB camera estimation within time of case 3-3.

doi:10.6342/NTU202404031

is the ground truth. In Figure 5.15(a) and Figure 5.15(b), the result shows the position comparison in X_W and Y_W directions respectively. In Figure 5.15(c) and Figure 5.15(d), the result shows the position error (%) comparison in X_W and Y_W directions respectively. We can observe that the onboard localization error (%) is in -25% to 20% range in X_W direction and in $\pm$30 % range in Y_W direction.

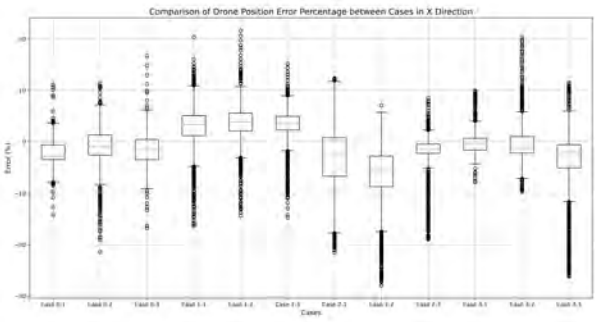
Similar to the previous cases, for case 3-1$\sim$ 3-3, we can observe that if there is more unstable flight or fast flight and less target information, the accuracy of onboard localization decreases, which means the error increases.

The comparison of the range of error for each case is shown in Figure 5.16. We can observe that from Figure 5.16(a) in X_W direction and Figure 5.16(b) in Y_W direction, cases with nonzero control command will cause larger localization error than the cases without control. Nonetheless, for control cases, the onboard localization error is not such obvious among these cases.

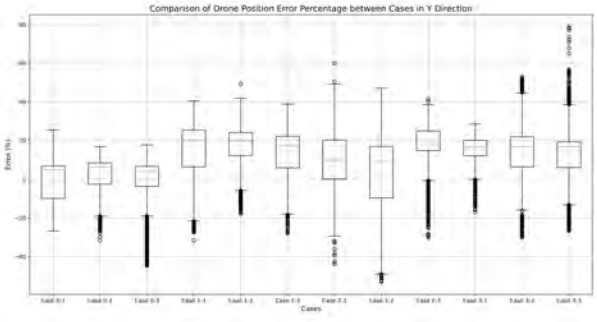
5.4 Estimated/Predicted Target Positions within Time without/with Curve Fitting

Prediction of the target motion is an vital essence in this system since we need to deal with the unpredictable estimation failure of the target result from unpredictable signal or obstacle occlusions. In this section, we will demonstrate the capability of curve fitting to predict the position of the target with some estimation failure from the drones. In each figure of this section, the first two subfigures are the directly estimation of the target from the image and the last two subfigures are the predicted position of the target using curve fitting. The (blue) circles, (green) crosses and (red) triangles are the estimated/predicted target positions of tello_1, tello_2 and tello_3 respectively. Furthermore, the prediction

doi:10.6342/NTU202404031



(a) Box plot of the onboard localization errors for every case in X_W direction.



(b) Box plot of the onboard localization errors for every case in Y_W direction.

Figure 5.16: The box plot comparison of the onboard localization errors for every case during control process.

doi:10.6342/NTU202404031

error of this method will be discussed in the next section.

5.4.1 Results of 0-1 $\sim$ 0-3, Non-Control Cases, without Obstacles

For case 0-1 $\sim$ 0-3 of the non-control cases, we will see the target out of all FOVs of the quadrotors if the target is moving. Nonetheless, we can also use curve fitting to predict the position of the target. The result of the estimated positions in X_W direction and Y_W direction, and predicted positions in X_W direction and Y_W direction for case 0-1 $\sim$ 0-3 is shown in Figure 5.17, Figure 5.18 and Figure 5.19.

For case 0-1 referring to Figure 5.17 without control and with stationary target, the estimation of the target is relatively stable. Although there might be some estimation in a small time interval, we can predict the target positions using curve fitting, as shown in Figure 5.17(c) and Figure 5.17(d).

However, for case 0-2 referring to Figure 5.18 without control and with moving target, the target will be out of FOVs of the drones because there is no control command. We can also predict the target positions using curve fitting, as shown in Figure 5.18(c) and Figure 5.18(d).

Similar to case 0-2, the result of case 0-3 referring to Figure 5.19 without control and with moving target, the target will be out of FOVs of the drones because there is no control command along X_W and Y_W directions. We can also predict the target positions using curve fitting, as shown in Figure 5.19(c) and Figure 5.19(d).

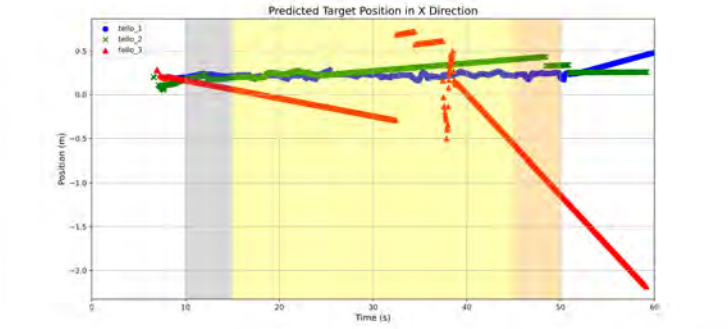
5.4.2 Results of 1-1 $\sim$ 1-3, Control Cases, without Obstacles

For case 1-1$\sim$ 1-3 with no obstacles, the situation only depends on the signal occlusion and positions of the quadrotors, which means there is no occlusion by obstacles in

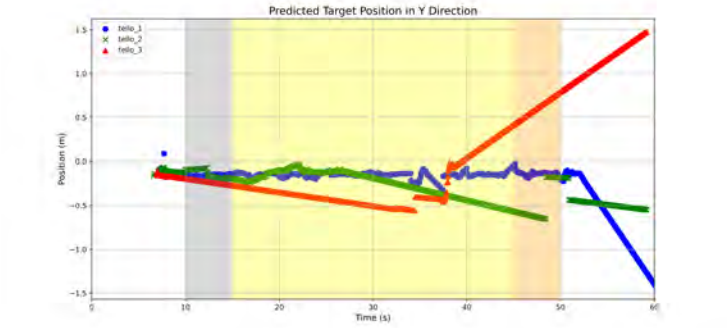
doi:10.6342/NTU202404031

(a) Target positions estimated by the drones in X_W direction.

(b) Target positions estimated by the drones in Y_W direction.



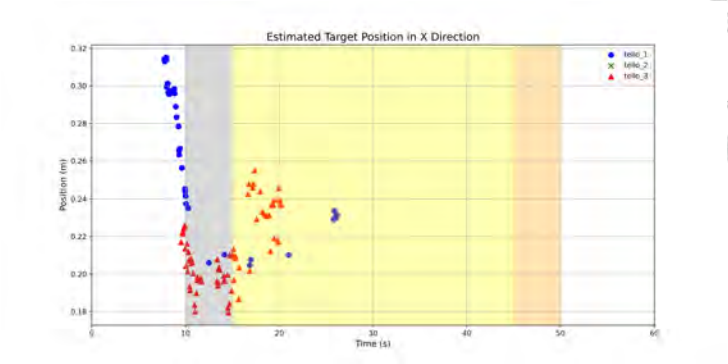
(c) Target positions predicted using curve fitting in X_W direction.



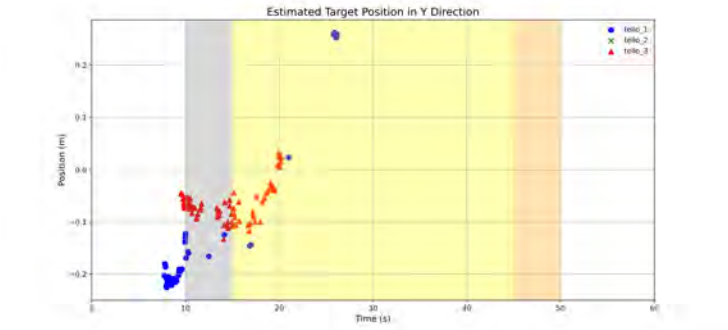
(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.17: The comparison of target positions using directly estimation and curve fitting within time of case 0-1.

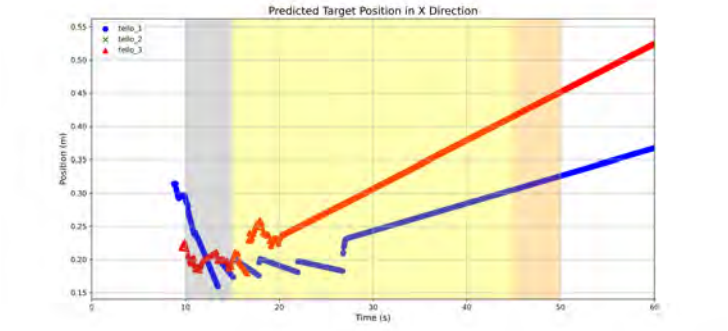
doi:10.6342/NTU202404031



(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



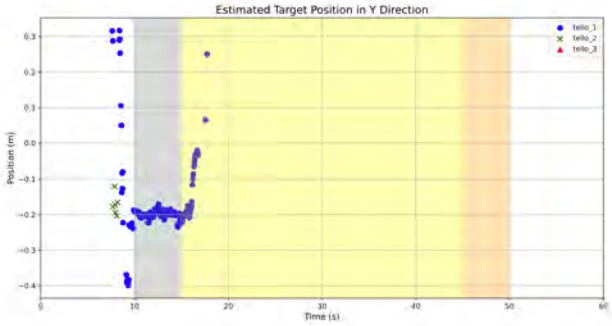
(c) Target positions predicted using curve fitting in X_W direction.

(d) Target positions predicted using curve fitting in Y_W direction.

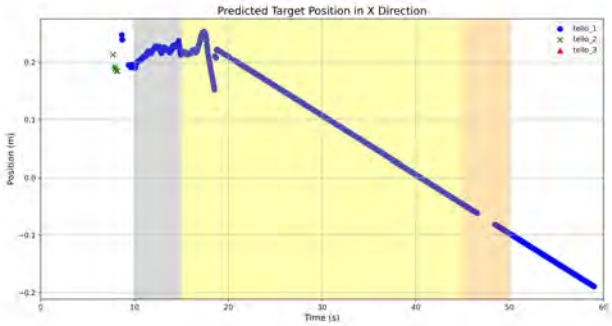
Figure 5.18: The comparison of target positions using directly estimation and curve fitting within time of case 0-2.

doi:10.6342/NTU202404031

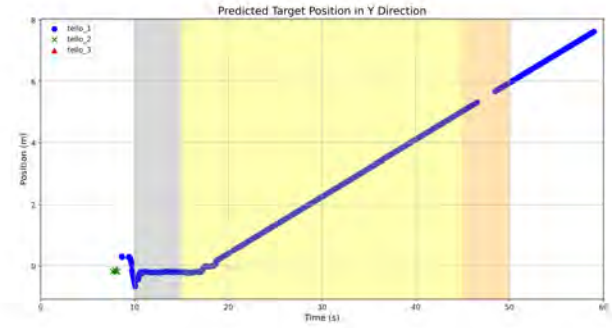
(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



(c) Target positions predicted using curve fitting in X_W direction.



(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.19: The comparison of target positions using directly estimation and curve fitting within time of case 0-3.

doi:10.6342/NTU202404031

these cases. The result of case 1-1 ∼ 1-3 is shown in Figure 5.20, Figure 5.21 and Figure 5.22.

For case 1-1 referring to Figure 5.20, we can observe that although there is no target detected at some time intervals such as 30s ∼ 32s in Figure 5.20(a) and Figure 5.20(b), we can compensate these estimation errors by curve fitting 1st order polynomials. As a result, the predicted trajectories will be 1st order polynomials if there is no new estimation point of the target, as shown in Figure 5.20(c) and Figure 5.20(d).

For case 1-2 referring to Figure 5.21, we can observe that although there is no target detected at some time intervals such as 20s ∼ 28s in Figure 5.21(a) and Figure 5.21(b), we can also compensate these estimation errors by curve fitting 1st order polynomials and if there is no new estimation point of the target, the predicted trajectories will be 1st order polynomials as well, as shown in Figure 5.21(c) and Figure 5.21(d).

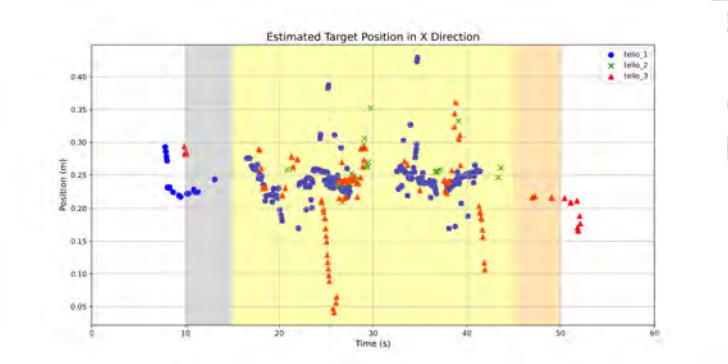
For case 1-3 referring to Figure 5.22, we can observe that although there is no target detected at some time intervals such as 17s ∼ 21s, 21s ∼ 23s and 23s ∼ 26s in Figure 5.22(a) and Figure 5.22(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.22(c) and Figure 5.22(d).

5.4.3 Results of 2-1 ∼ 2-3, Control Cases, with an Obstacle

For case 2-1 ∼ 2-3 with an obstacle, the data obtaining is harder than case 1-1 ∼ 1-3 since there is the situation of the occlusions by obstacles added. The result of case 2-1 ∼ 2-3 is shown in Figure 5.23, Figure 5.24 and Figure 5.25. Similarly, although there is some estimation failure at some time intervals, we can use curve fitting to solve this kind of problem and obtain the target positions in real time.

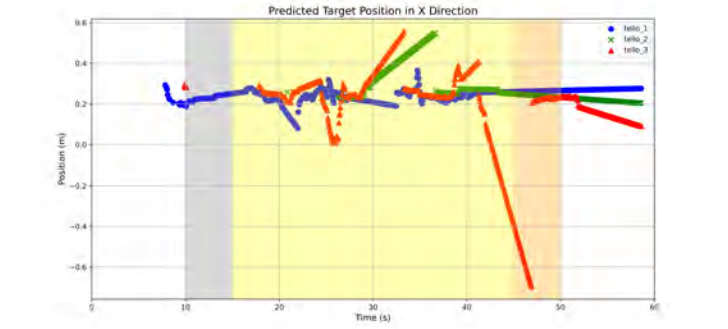
For case 2-1 referring to Figure 5.23, we can observe that although there is no target

doi:10.6342/NTU202404031



(a) Target positions estimated by the drones in X_W direction.

(b) Target positions estimated by the drones in Y_W direction.

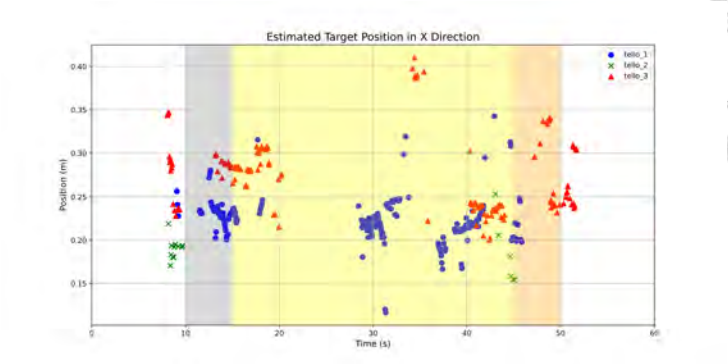


(c) Target positions predicted using curve fitting in X_W direction.

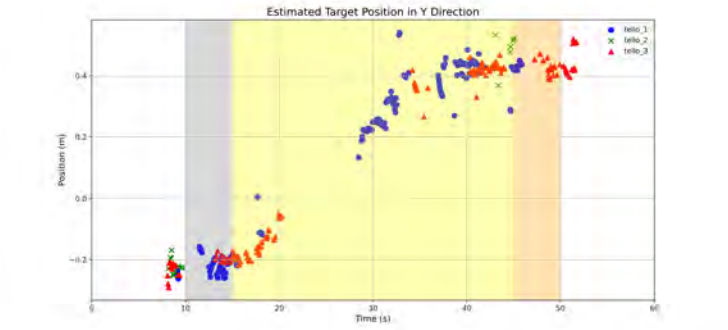
(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.20: The comparison of target positions using directly estimation and curve fitting within time of case 1-1.

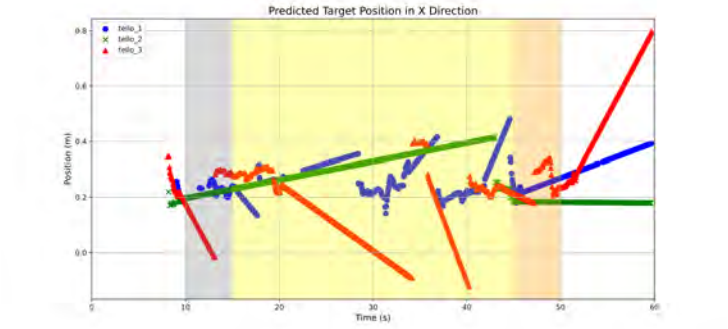
doi:10.6342/NTU202404031



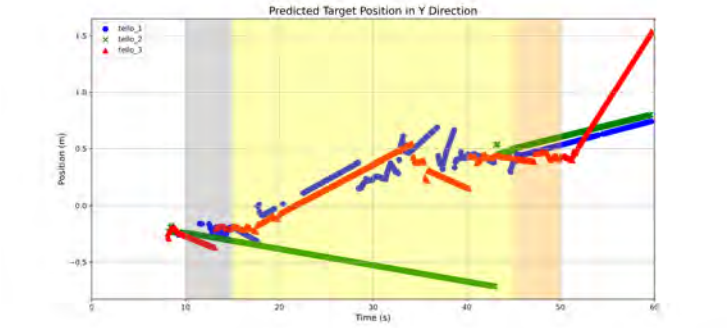
(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



(c) Target positions predicted using curve fitting in X_W direction.



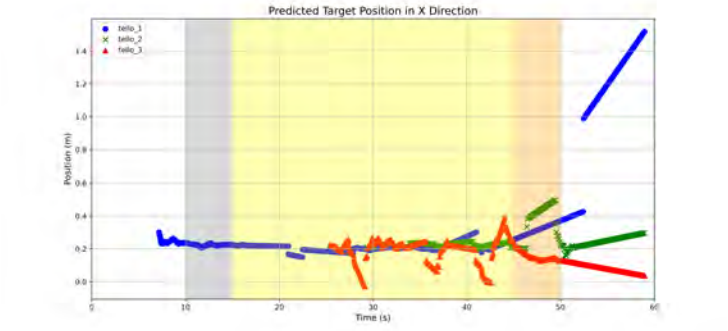
(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.21: The comparison of target positions using directly estimation and curve fitting within time of case 1-2.

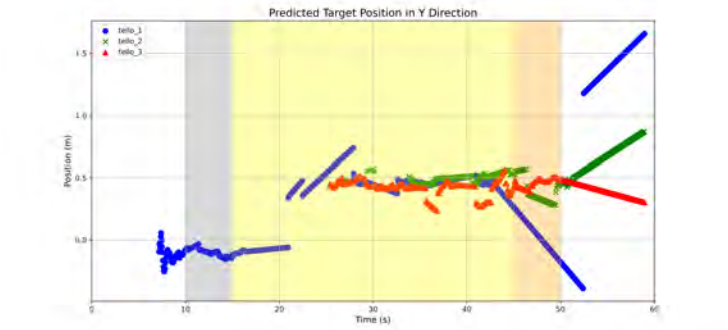
doi:10.6342/NTU202404031

(a) Target positions estimated by the drones in X_W direction.

(b) Target positions estimated by the drones in Y_W direction.



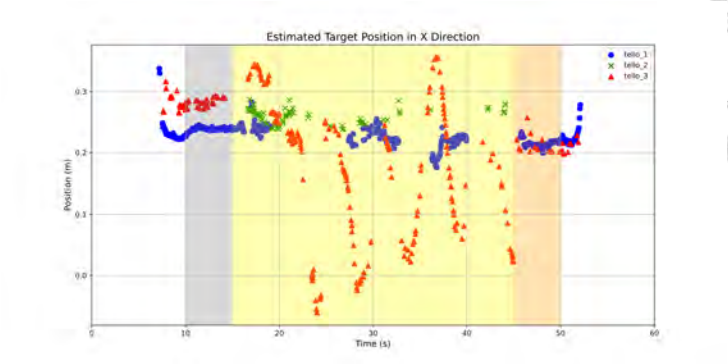
(c) Target positions predicted using curve fitting in X_W direction.



(d) Target positions predicted using curve fitting in Y_W direction.

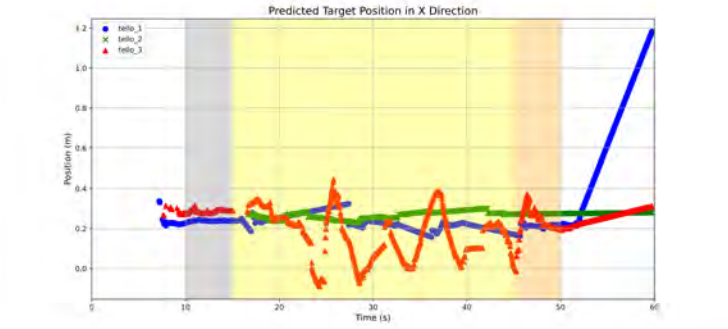
Figure 5.22: The comparison of target positions using directly estimation and curve fitting within time of case 1-3.

doi:10.6342/NTU202404031



(a) Target positions estimated by the drones in X_W direction.

(b) Target positions estimated by the drones in Y_W direction.



(c) Target positions predicted using curve fitting in X_W direction.

(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.23: The comparison of target positions using directly estimation and curve fitting within time of case 2-1.

doi:10.6342/NTU202404031

detected at some time intervals such as 40s ∼ 42s in Figure 5.23(a) and Figure 5.23(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.23(c) and Figure 5.23(d).

For case 2-2 referring to Figure 5.24 when the target is moving slowly, we can observe that although there is no target detected at some time intervals such as 32s ∼ 34s in Figure 5.24(a) and Figure 5.24(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.24(c) and Figure 5.24(d).

For case 2-3 referring to Figure 5.25 when the target is moving fast, In this case, the agents lose most of the target information before 30s except tello_1. We can observe that although there is no target detected at some time intervals such as 18s ∼ 20s, 30s ∼ 31s, 40s ∼ 41s and 42s ∼ 44s in Figure 5.25(a) and Figure 5.25(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.25(c) and Figure 5.25(d).

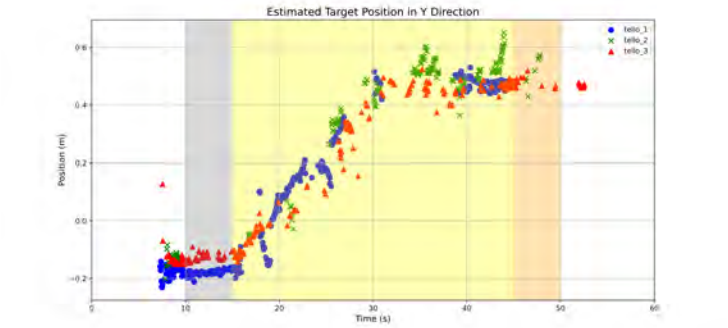
5.4.4 Results of 3-1 ∼ 3-3, Control Cases, with three Obstacles

The result of case 3-1 ∼ 3-3 is shown in Figure 5.26, Figure 5.27 and Figure 5.28. We can observe that the variety information of the target positions is less than the case without obstacles (1-1∼1-3) and with an obstacle (2-1∼2-3) . However, we can also use curve fitting to solve the problem of unpredictable estimation failure.

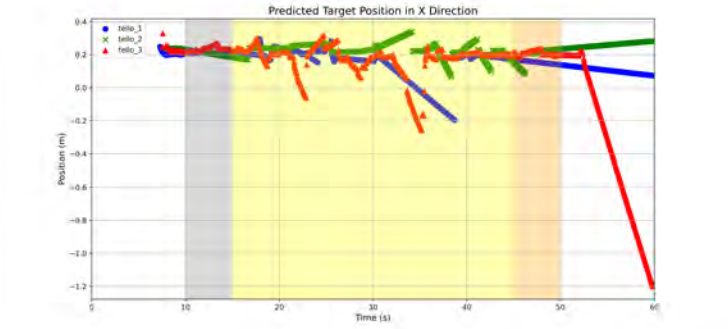
For case 3-1 referring to Figure 5.26, in this case, target information appears with intermittent. We can observe that although there is no target detected at some time intervals such as 15s ∼ 18s in Figure 5.26(a) and Figure 5.26(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.26(c)

doi:10.6342/NTU202404031

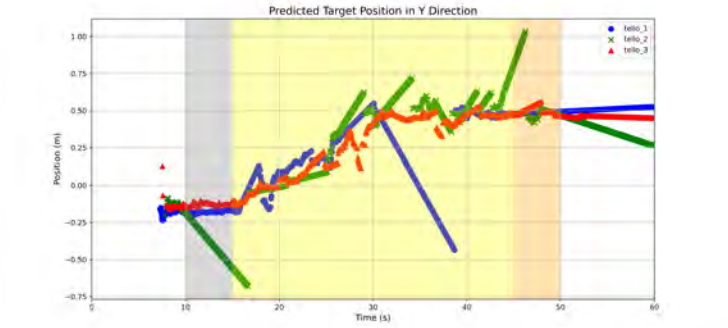
(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



(c) Target positions predicted using curve fitting in X_W direction.

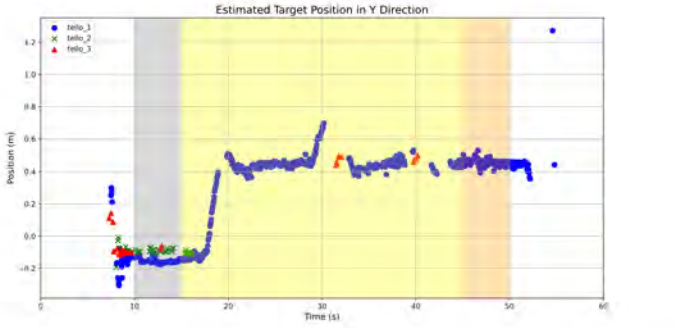


(d) Target positions predicted using curve fitting in Y_W direction.

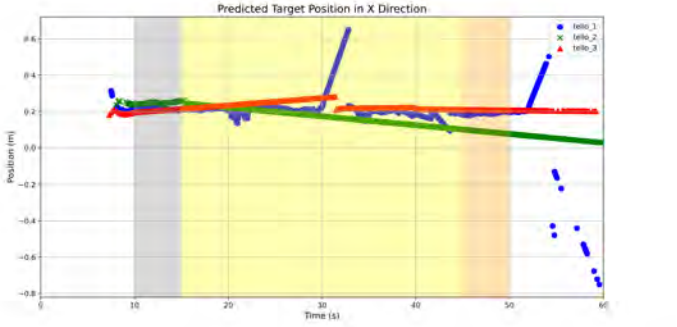
Figure 5.24: The comparison of target positions using directly estimation and curve fitting within time of case 2-2.

doi:10.6342/NTU202404031

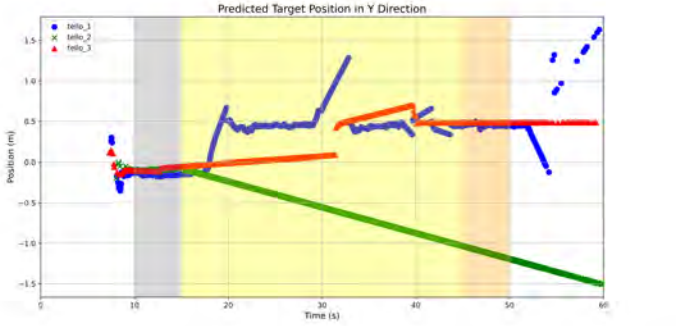
(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



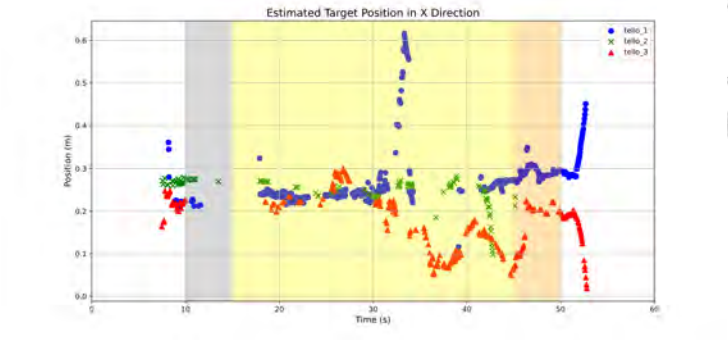
(c) Target positions predicted using curve fitting in X_W direction.



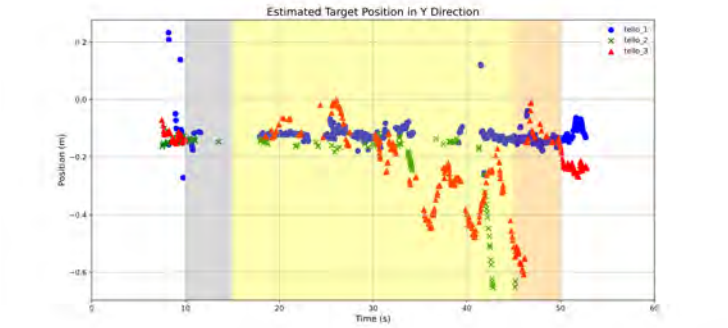
(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.25: The comparison of target positions using directly estimation and curve fitting within time of case 2-3.

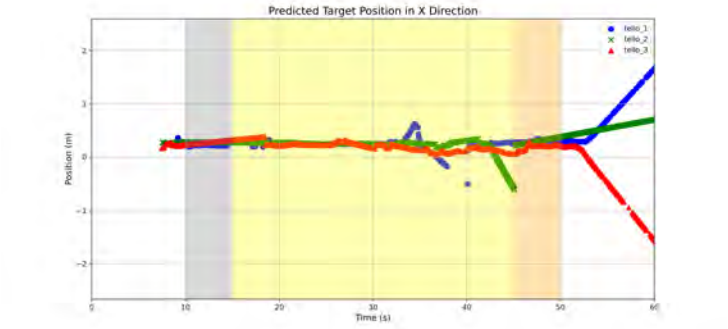
doi:10.6342/NTU202404031



(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.

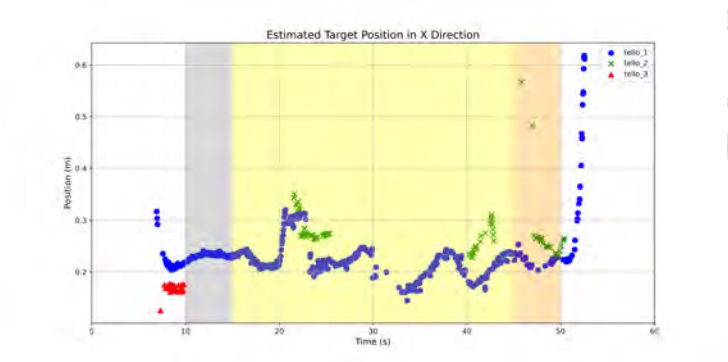


(c) Target positions predicted using curve fitting in X_W direction.

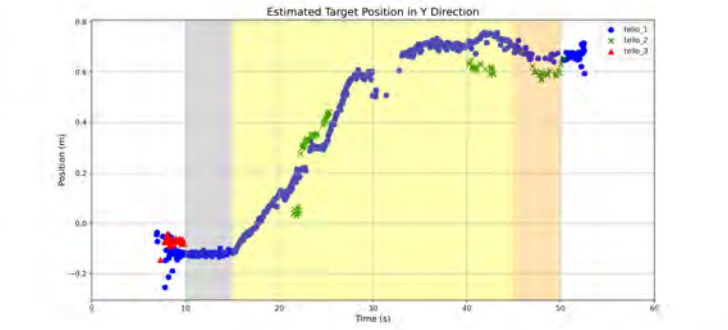
(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.26: The comparison of target positions using directly estimation and curve fitting within time of case 3-1.

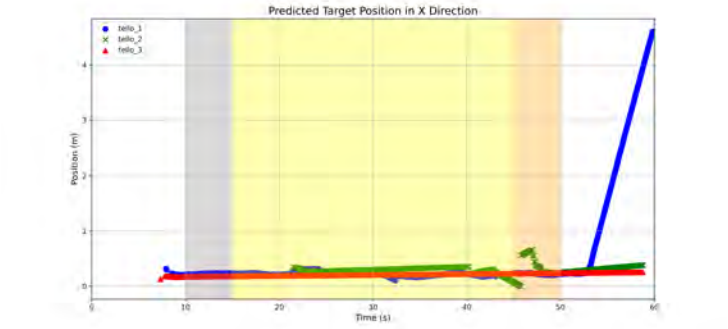
doi:10.6342/NTU202404031



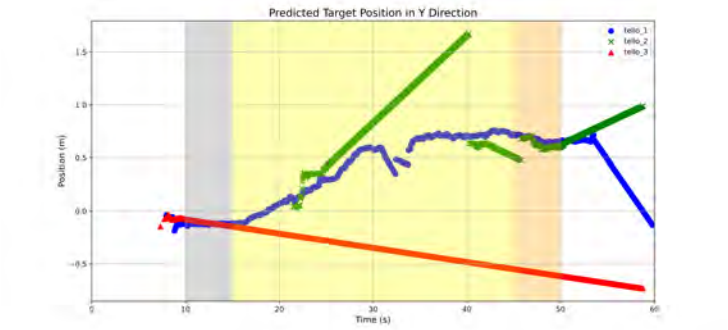
(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



(c) Target positions predicted using curve fitting in X_W direction.



(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.27: The comparison of target positions using directly estimation and curve fitting within time of case 3-2.

doi:10.6342/NTU202404031

and Figure 5.26(d).

For case 3-2 referring to Figure 5.27 for slow speed of the target, in this case, tello_2 and tello_3 could not obtain the target information due to occlusions by obstacles. We can observe that although there is no target detected at some time intervals such as 31s ∼ 33s in Figure 5.27(a) and Figure 5.27(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.27(c) and Figure 5.27(d).

For case 3-3 referring to Figure 5.28 for fast speed of the target, in this case, tello_2 and tello_3 are hard to obtain the target information due to occlusions by obstacles until 35s. We can observe that although there is no target detected at some time intervals such as 29s ∼ 32s and 34s ∼ 36s in Figure 5.28(a) and Figure 5.28(b), We can use curve fitting 1st order polynomials to compensate the data loss of the target position, as shown in Figure 5.28(c) and Figure 5.28(d).

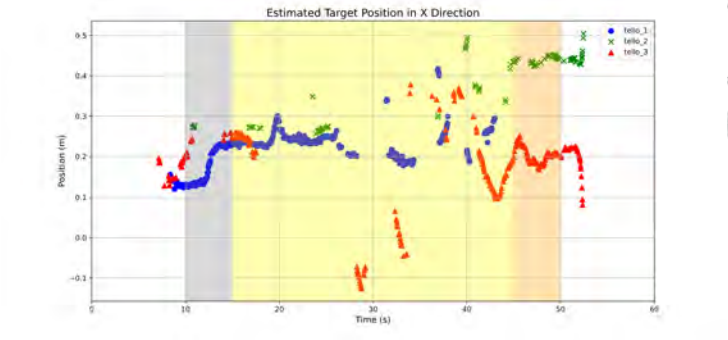
In this section, we demonstrate the solution of unpredictable estimation failure of the target using trajectory-based motion prediction method. We will discuss the accuracy of the onboard prediction with averaging method in the next section.

5.5 Onboard Prediction/RGB Camera Estimation of the Target Position within Time

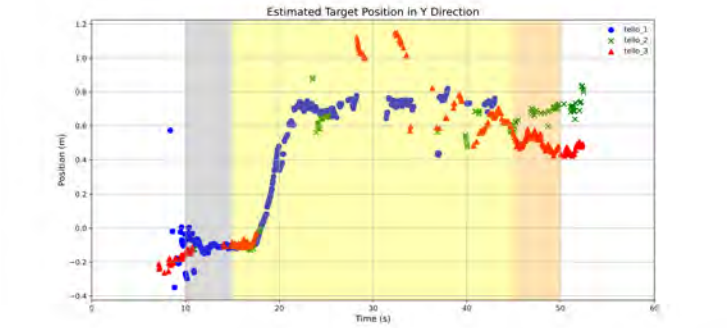
Obtaining the target position is one of the vital task for tactical ISR. With accurate target information, it will help and enhance the whole police team on the ground find the target more easily.

In the previous section, we discuss the trajectory-based method to predict the target

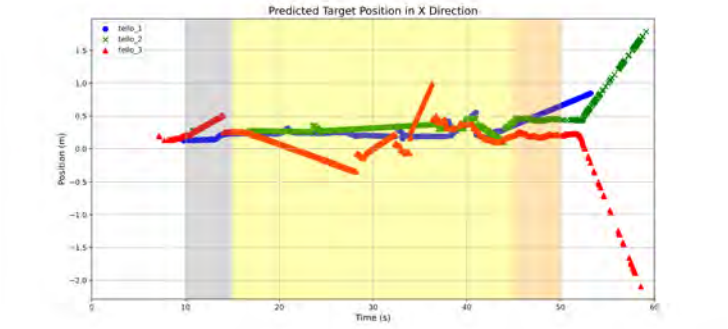
doi:10.6342/NTU202404031



(a) Target positions estimated by the drones in X_W direction.



(b) Target positions estimated by the drones in Y_W direction.



(c) Target positions predicted using curve fitting in X_W direction.

(d) Target positions predicted using curve fitting in Y_W direction.

Figure 5.28: The comparison of target positions using directly estimation and curve fitting within time of case 3-3.

doi:10.6342/NTU202404031

motion. In this section, we will discuss the accuracy of the unique position of the target by onboard prediction and averaging algorithm in Equation 4.20 and 4.21 as the formation control reference. For each case in this section, the first two subfigures are the comparison of the reference target position by onboard prediction and averaging (black solid curves) and the actual target position by RGB camera-based motion capture system from the third perspective (purple dashed curves). The last two subfigures are the onboard prediction error in X_W and Y_W direction respectively where the error is defined as

$$error(\%) = \frac{position(onboard) - position(RGB)}{height} \times 100\%. \tag{5.2}$$

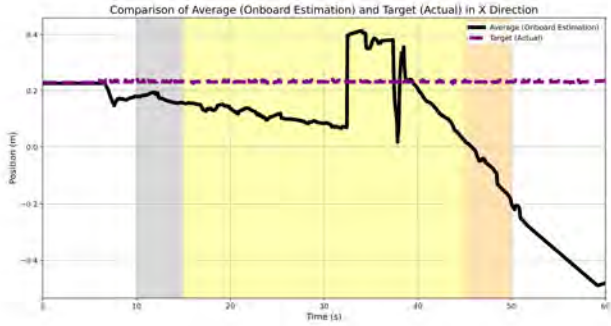
5.5.1 Results of 0-1 $\sim$ 0-3, Non-Control Cases, without Obstacles

For non-control cases of 0-1 $\sim$ 0-3, because the flight of quadrotors are relatively stable and the target is stationary or in nearly constant speed, the accuracy of the target position should be higher. The result of case 0-1 $\sim$ 0-3 is shown in Figure 5.29, Figure 5.30 and Figure 5.31.

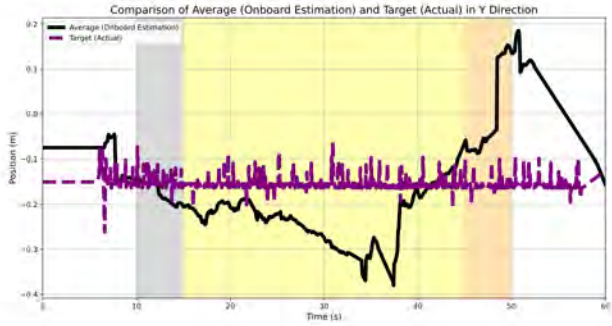
For case 0-1 referring to Figure 5.29, in this case, all agents are stationary. We can observe that in Figure 5.29(a) and Figure 5.29(b), the actual target position by RGB camera is nearly stationary, and the position by the algorithm for extracting the target reference is not such stable. Nonetheless, the error is about $\pm$15 % in X_W direction and -15% to 10% in Y_W direction, as shown in Figure 5.29(c) and Figure 5.29(d).

For case 0-2 referring to Figure 5.30, in this case, the target is moving slowly but the quadrotors are nearly stationary. We can observe that in Figure 5.30(a) and Figure 5.30(b), the actual target position by RGB camera is nearly stationary in X_W direction and moving slowly with constant speed in Y_W direction, and the position by the algorithm

doi:10.6342/NTU202404031



(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

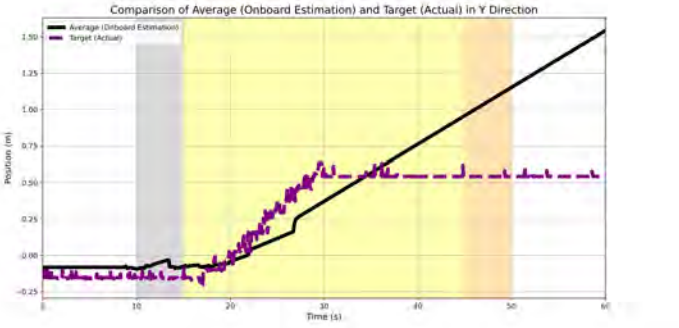
(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.

(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

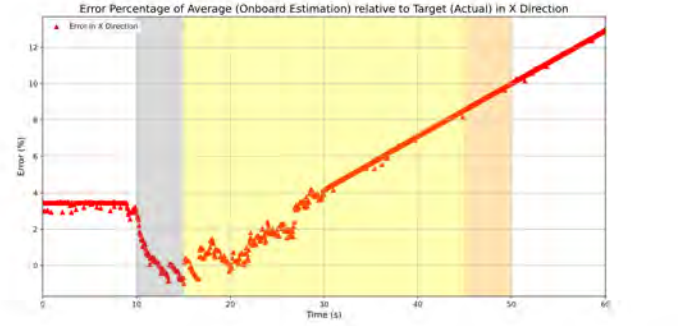
Figure 5.29: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 0-1.

doi:10.6342/NTU202404031

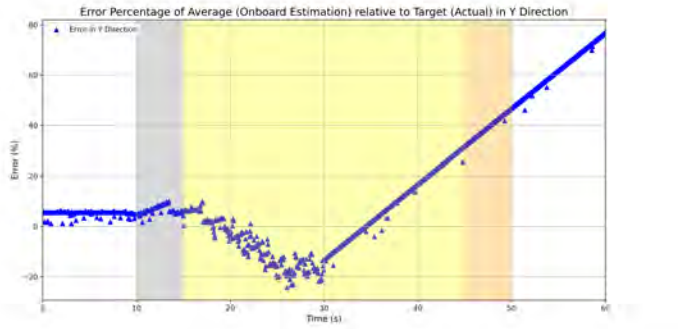
(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.30: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 0-2.

doi:10.6342/NTU202404031

for extracting the target reference is not such stable. The error is about -1% to 9% in X_W direction and -20% to 30% in Y_W direction. It has not much different in X_W direction but is much larger in Y_W direction than case 0-1 , as shown in Figure 5.30(c) and Figure 5.30(d).

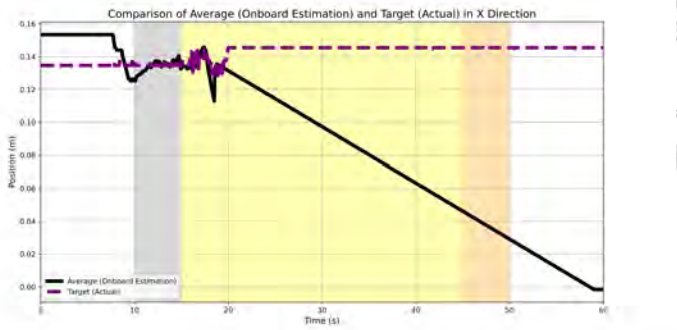
For case 0-3 referring to Figure 5.31, in this case, the target is moving fast but the quadrotors are nearly stationary. We can observe that in Figure 5.31(a) and Figure 5.31(b), the actual target position by RGB camera is nearly stationary in X_W direction and moving fast with constant speed in Y_W direction, and the position by the algorithm for extracting the target reference is not such stable. The error is about -8% to 1% in X_W direction and -45% to 75% in Y_W direction. It has not much different in X_W direction as Figure 5.31(c) but is much larger than case 0-1 and 0-2 in Y_W direction as Figure 5.31(d) since the quadrotors lose the target information in a short time , as shown in Figure 5.19(a) and Figure 5.19(b).

5.5.2 Results of 1-1 $\sim$ 1-3, Control Cases, without Obstacles

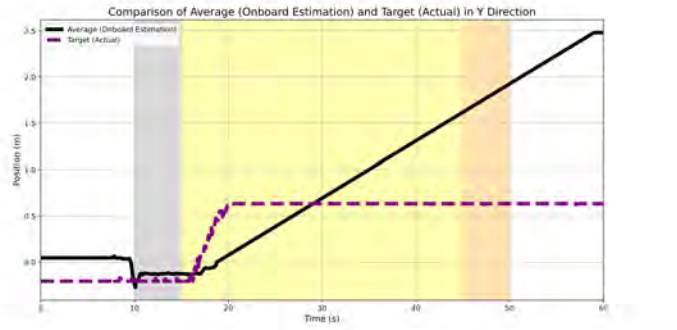
For case 1-1 $\sim$ 1-3 without obstacles, since the quadrotors is moving with velocity command in X_W and Y_W directions, the accuracy of the target position should be less than case 0-1 $\sim$ 0-3. The result of case 1-1 $\sim$ 1-3 is shown in Figure 5.32, Figure 5.33 and Figure 5.34.

For case 1-1 referring to Figure 5.32, in this case, the quadrotors are moving but the target is nearly stationary. We can observe that in Figure 5.32(a) and Figure 5.32(b), the actual target position by RGB camera is nearly stationary in X_W and Y_W directions and the position by the algorithm for extracting the target reference is not such stable than case 0-1 $\sim$ 0-3. The error is about -10% to 20% in X_W direction and -5% to 15% in Y_W

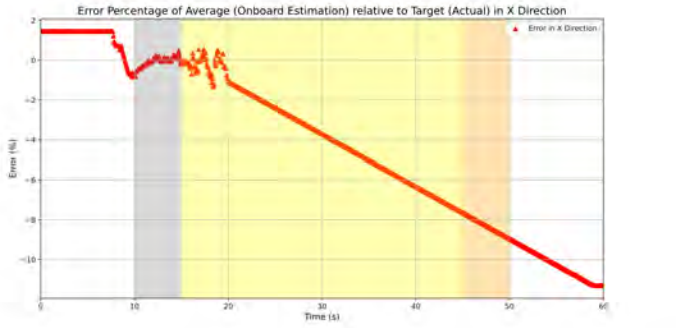
doi:10.6342/NTU202404031



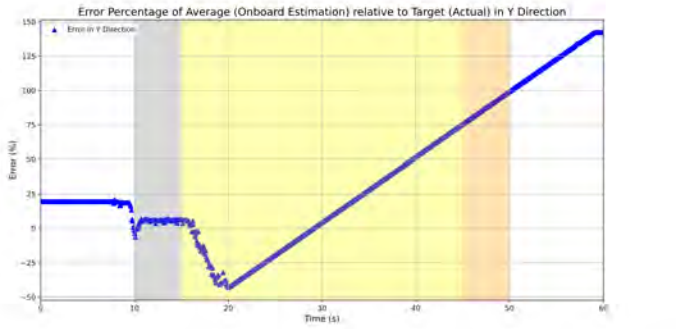
(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



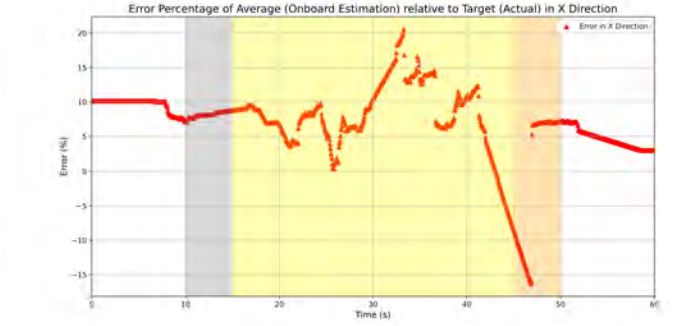
(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.31: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 0-3.

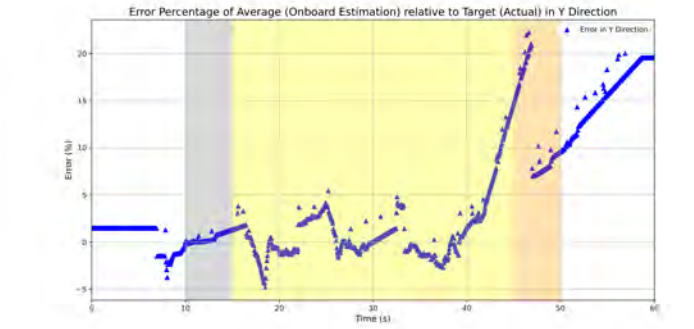
doi:10.6342/NTU202404031

(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.

(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.32: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 1-1.

doi:10.6342/NTU202404031

direction , as shown in Figure 5.32(c) and Figure 5.32(d).

For case 1-2 referring to Figure 5.33, in this case, the quadrotors are moving and the target is moving slowly. We can observe that in Figure 5.33(a) and Figure 5.33(b), since the actual target position by RGB camera is nearly stationary in X_W direction and slowly moving Y_W direction, the position by the algorithm for extracting the target reference is not such stable than case 1-1. The error is about -2% to 10% in X_W direction and -30% to 10% in Y_W direction , as shown in Figure 5.33(c) and Figure 5.33(d).

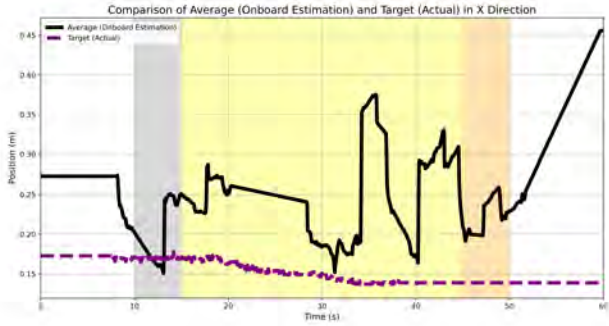
For case 1-3 referring to Figure 5.34, in this case, the quadrotors are moving and the target is moving fast. We can observe that in Figure 5.34(a) and Figure 5.34(b), since the actual target position by RGB camera is nearly stationary in X_W direction and fast moving Y_W direction, the position by the algorithm for extracting the target reference is not such stable than case 1-1 and 1-2. The error is about -2% to 10% in X_W direction and -5% to 40% in Y_W direction , as shown in Figure 5.34(c) and Figure 5.34(d).

5.5.3 Results of 2-1 $\sim$ 2-3, Control Cases, with an Obstacle

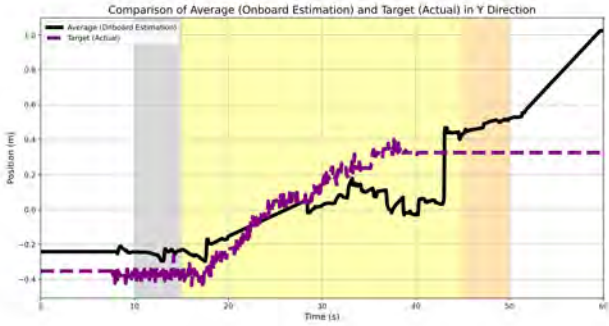
For case 2-1 $\sim$ 2-3 with an obstacle , since the quadrotors is moving with velocity command in X_W and Y_W directions and occlusion by an obstacle, the accuracy of the target position should be less than case 0-1 $\sim$ 0-3 and 1-1 $\sim$ 1-3. The result of case 2-1 $\sim$ 2-3 is shown in Figure 5.35, Figure 5.36 and Figure 5.37.

For case 2-1 referring to Figure 5.35, in this case, the quadrotors are moving but the target is stationary. We can observe that in Figure 5.35(a) and Figure 5.35(b), the actual target position by RGB camera is nearly stationary in X_W and Y_W directions. Nonetheless, since there is occlusion by an obstacle, the position by the algorithm for extracting the target reference is not such stable than 1-1 for the stationary case of the target. The error

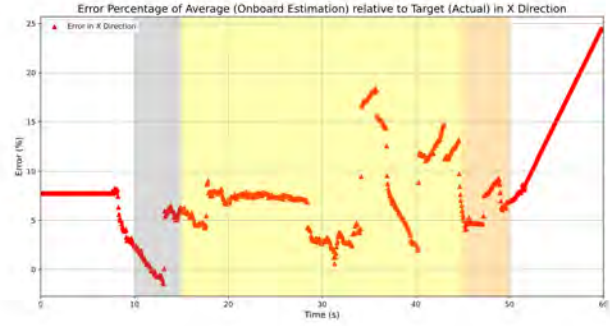
doi:10.6342/NTU202404031



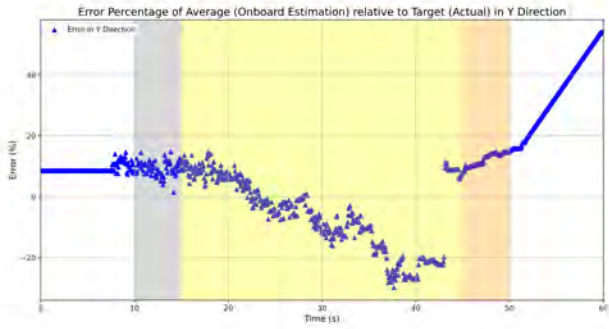
(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



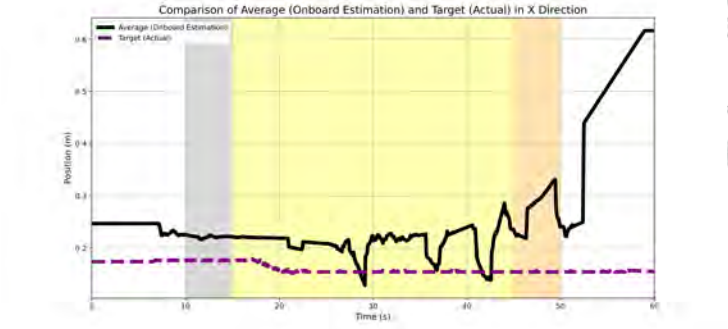
(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



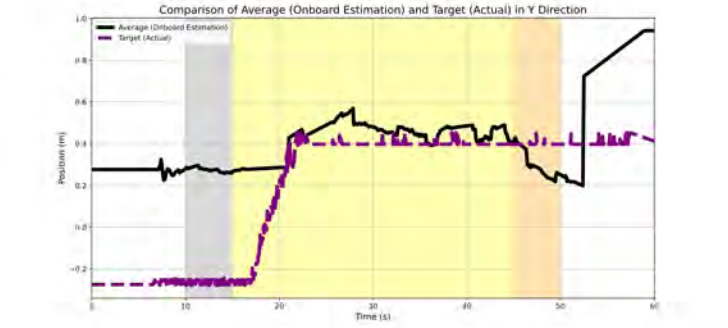
(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.33: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 1-2.

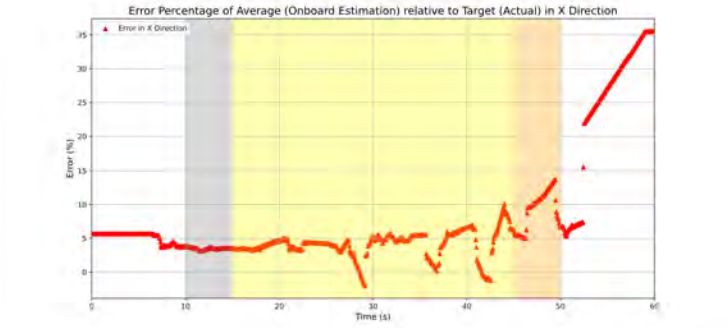
doi:10.6342/NTU202404031



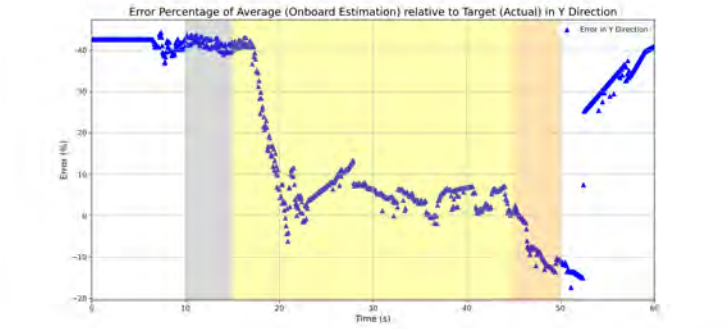
(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



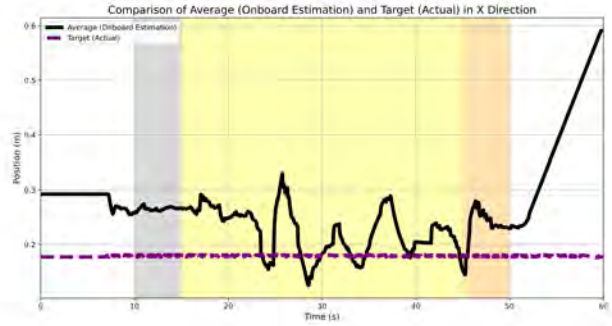
(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

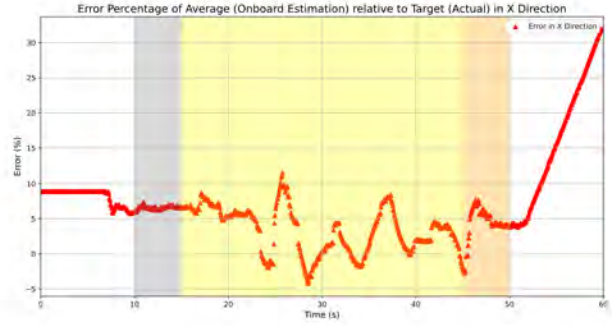
Figure 5.34: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 1-3.

doi:10.6342/NTU202404031

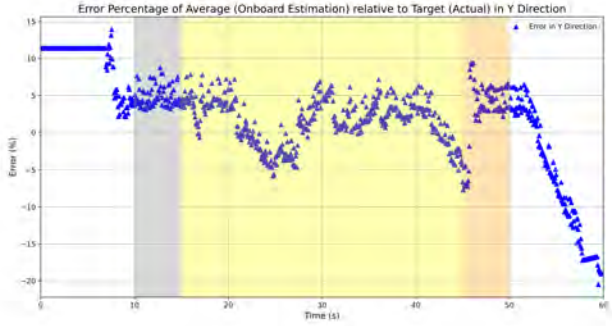


(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.

(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.35: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 2-1.

doi:10.6342/NTU202404031

is about -5% to 10% in X_W direction and $\pm$10 % in Y_W direction , as shown in Figure 5.35(c) and Figure 5.35(d).

For case 2-2 referring to Figure 5.36, in this case, the quadrotors are moving and the target is moving slowly. We can observe that in Figure 5.36(a) and Figure 5.36(b), since the actual target position by RGB camera is nearly stationary in X_W direction and moving slowly in Y_W direction and there is occlusion in the scenario, the position by the algorithm for extracting the target reference is not such stable than case 1-2 and 2-1. The error is about -15% to 10% in X_W direction and -20% to 10% in Y_W direction , as shown in Figure 5.36(c) and Figure 5.36(d).

For case 2-3 referring to Figure 5.37, in this case, the quadrotors are moving and the target is moving fast. We can observe that in Figure 5.37(a) and Figure 5.37(b), since the actual target position by RGB camera is nearly stationary in X_W direction and moving fast in Y_W direction and there is occlusion in the scenario, the position by the algorithm for extracting the target reference is not such stable than case 1-3, 2-1 and 2-2. The error is about -3% to 15% in X_W direction and -15% to 30% in Y_W direction , as shown in Figure 5.37(c) and Figure 5.37(d).

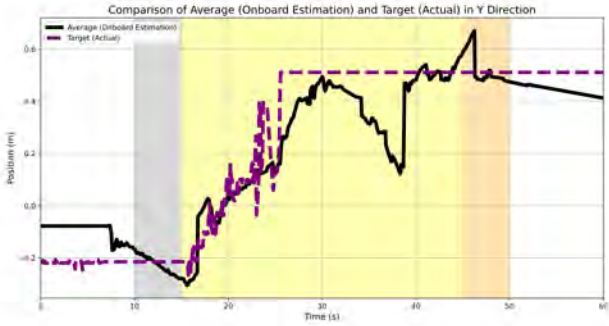
5.5.4 Results of 3-1 $\sim$ 3-3, Control Cases, with three Obstacles

For case 3-1 $\sim$ 3-3, since there are more occlusions by multiple obstacles, it can be thought of that the error should be larger than case 1 and 2 for the similar moving speeds. The result of case 3-1 $\sim$ 3-3 is shown in Figure 5.38, Figure 5.39 and Figure 5.40.

For case 3-1 referring to Figure 5.38, in this case, the quadrotors are moving but the target is stationary. We can observe that in Figure 5.38(a) and Figure 5.38(b), the actual target position by RGB camera is nearly stationary in X_W and Y_W directions. Nonetheless,

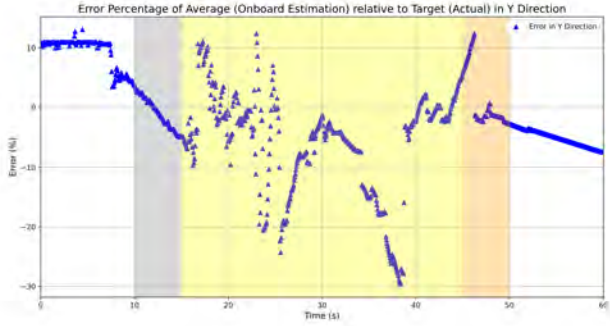
doi:10.6342/NTU202404031

(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

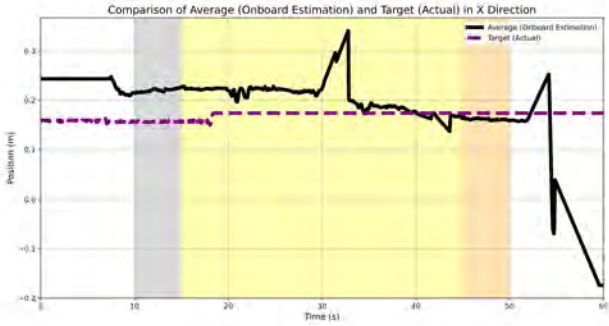
(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



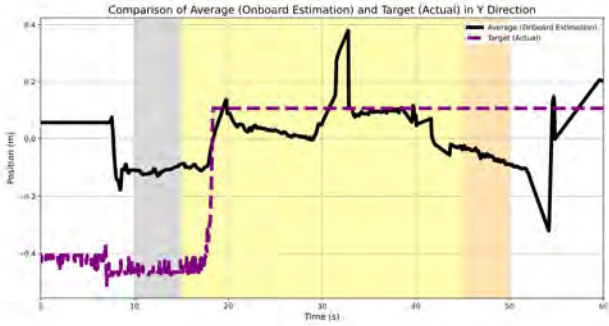
(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.36: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 2-2.

doi:10.6342/NTU202404031

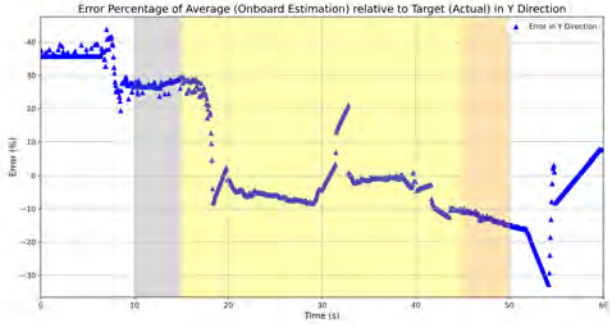


(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

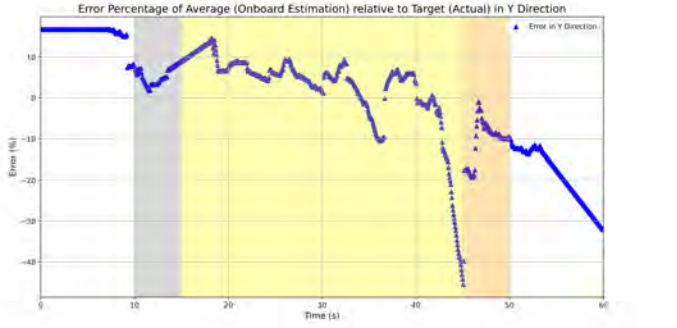
Figure 5.37: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 2-3.

doi:10.6342/NTU202404031

(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.

(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.38: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 3-1.

doi:10.6342/NTU202404031

since the flight is not such stable due to the lack of target information, the position by the algorithm for extracting the target reference is not such stable than case 1-1 and 2-1. The error is about ±15 % in X_W direction and -30% to 15% in Y_W direction , as shown in Figure 5.38(c) and Figure 5.38(d).

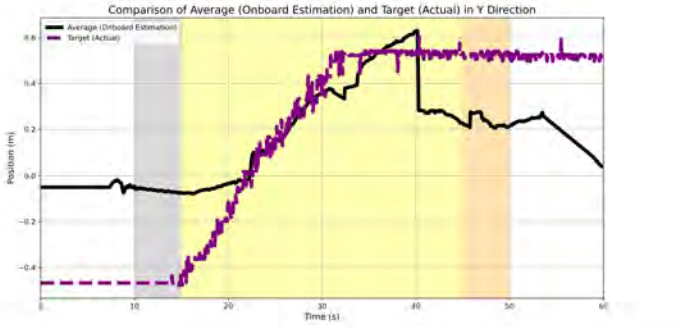
For case 3-2 referring to Figure 5.39, in this case, the quadrotors are moving and the target is moving slowly. We can observe that in Figure 5.39(a) and Figure 5.39(b), the actual target position by RGB camera is nearly stationary in X_W direction and moving slow in Y_W direction. Nonetheless, since the flight is not such stable due to the lack of target information and slowly movement of the target and quadrotors, the position by the algorithm for extracting the target reference is not such stable than case 1-2, 2-2 and 3-1. The error is about 0% to 10% in X_W direction and -25% to 30% in Y_W direction , as shown in Figure 5.39(c) and Figure 5.39(d).

For case 3-3 referring to Figure 5.40, in this case, the quadrotors are moving and the target is moving fast. We can observe that in Figure 5.40(a) and Figure 5.40(b), the actual target position by RGB camera is nearly stationary in X_W direction and moving fast in Y_W direction. Nonetheless, since the flight is not such stable due to the lack of target information and fast movement of the target and quadrotors, the position by the algorithm for extracting the target reference is not such stable than case 1-3, 2-3, 3-1 and 3-2. The error is about -10% to 25% in X_W direction and ±20 % in Y_W direction , as shown in Figure 5.40(c) and Figure 5.40(d).

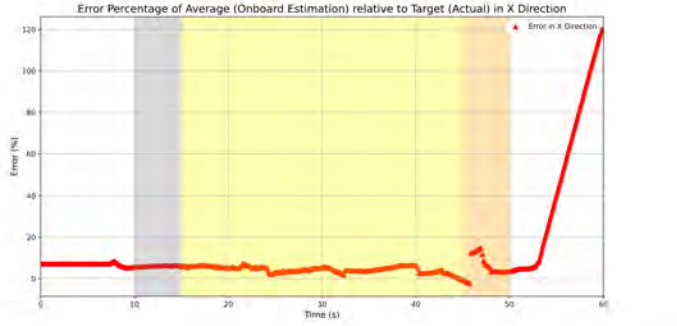
The comparison of the range of error for each case is shown in Figure 5.41. We can observe that from Figure 5.41(a) in X_W direction, there is no obvious tendency of the error among control cases but less error for non-control cases. Furthermore, from Figure 5.41(b)

doi:10.6342/NTU202404031

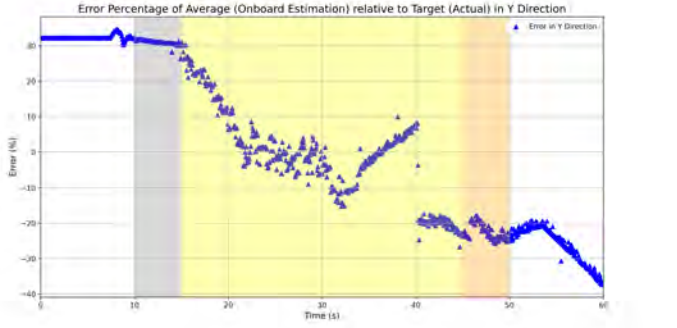
(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.



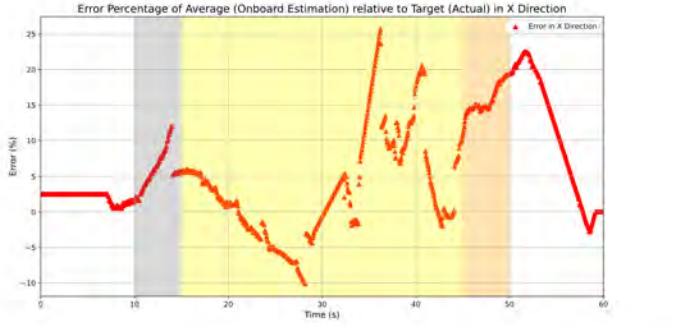
(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

Figure 5.39: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 3-2.

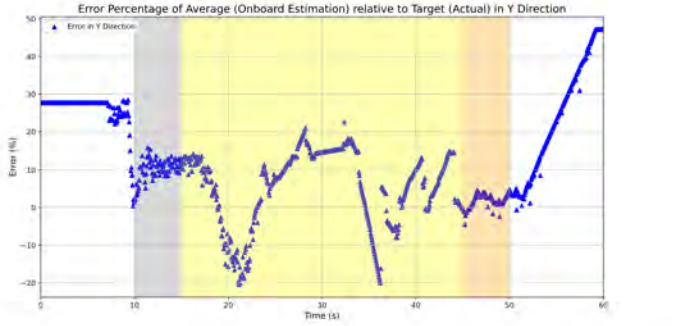
doi:10.6342/NTU202404031

(a) Comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.

(b) Comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.



(c) Error comparison of onboard prediction by averaging and estimation by RGB camera in X_W direction.

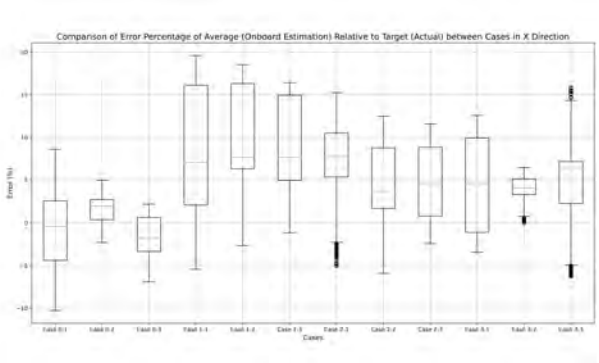


(d) Error comparison of onboard prediction by averaging and estimation by RGB camera in Y_W direction.

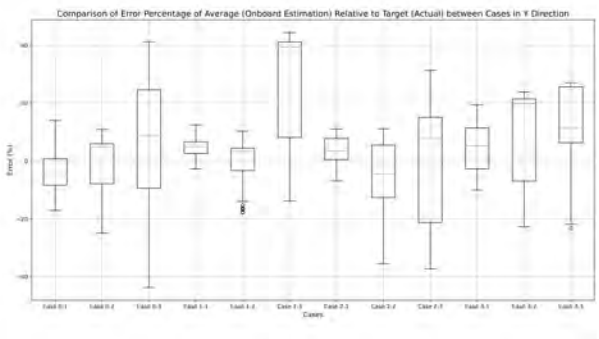
Figure 5.40: The comparison of onboard prediction by averaging and estimation by RGB camera as well as error within time of case 3-3.

doi:10.6342/NTU202404031

in Y_W direction, more occlusion will cause larger cooperative prediction error since there is less information of the target, but it is not such obvious. Simultaneously, larger speed of the target or drones and case without control will cause larger cooperative prediction error.



(a) Box plot of the onboard target prediction errors for every case in X_W direction.



(b) Box plot of the onboard target prediction errors for every case in Y_W direction.

Figure 5.41: The box plot comparison of the onboard target prediction errors for every case during control process.

5.6 Actual Positions by RGB Camera within Time

Multi-agent formation tracking control is the main essence of the system to observe the target we want to track. In this section, we will discuss the capability of formation tracking algorithms in the system. The comparison of all the positions captured by RGB camera is shown in this section. Furthermore, we will compare the position errors relative

doi:10.6342/NTU202404031

to the desired positions of the agents. The error is defined as

$$error(\%) = \frac{current\ position - desired\ position}{desired\ distance} \times 100\%, \tag{5.3}$$

where the $desired\ distance = 0.2\sqrt{3}(m)$, which is also the distance between the center and vertices of the desired triangle formation. Note that in the first two graphs for each case, the curves formed by (red) circles, (blue) crosses, (green) triangles and (purple) squares are the ground truth of tello_1, tello_2, tello_3 and the target respectively, and the (black) solid curve is the center of the triangular formation. Furthermore, for the last two graphs of the error in each case, the curves formed by (red) circles, (blue) crosses, (green) triangles and (purple) squares are the ground truth of tello_1, tello_2, tello_3 and the target respectively.

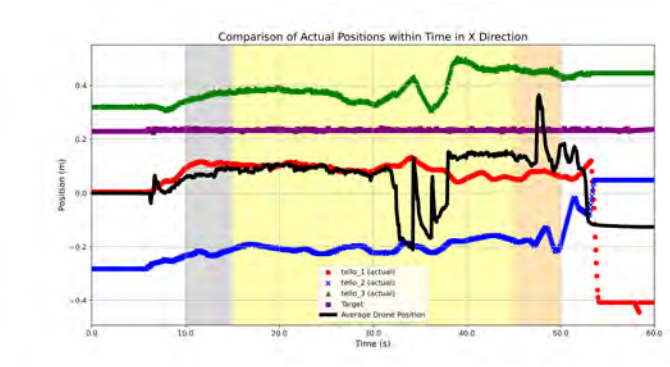
5.6.1 Results of 0-1 $\sim$ 0-3, Non-Control Cases, without Obstacles

We first examine that what will happen if there is no control command in X_W and Y_W directions in order to compare the cases with control. Figure 5.42, Figure 5.43 and Figure 5.44 denoted as the non-control cases show the positions of the drones and the target as well as the errors with respect to the desired positions.

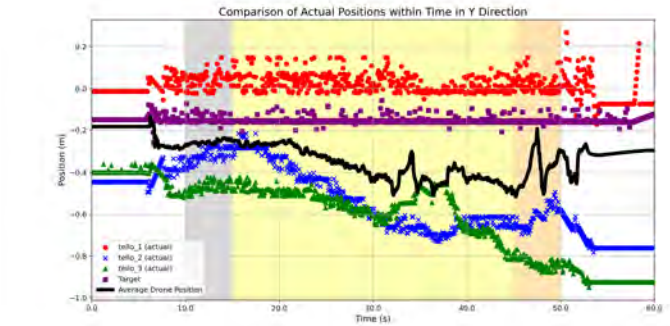
In Figure 5.42, since the target and quadrotors are not moving, the error will remains nearly constant without convergence at $\pm$20 % range in X_W and Y_W directions. Nonetheless, in case 0-2 and 0-3, since the target is moving, the absolute value of error will increase.

In Figure 5.43, the target is moving slowly along Y_W direction with nearly stationary in X_W direction, so the error in Y_W direction will slowly decrease -60% without any feedback control command while in X_W direction it has not much difference than case

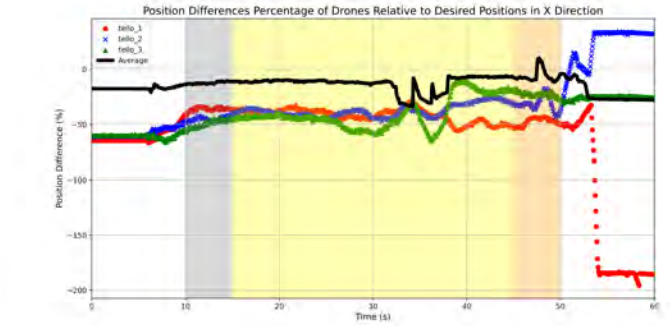
doi:10.6342/NTU202404031



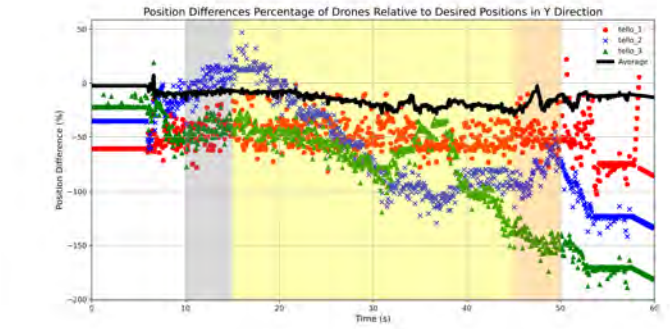
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



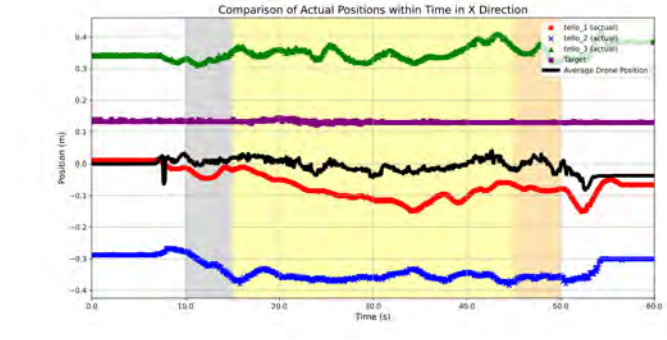
(c) Drone position errors by RGB camera estimation in X_W direction.



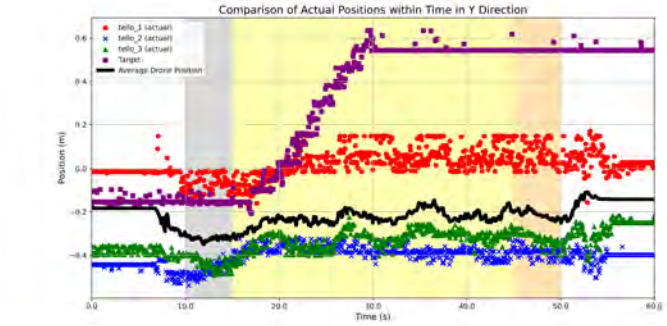
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.42: The comparison of actual positions and errors by RGB camera estimation within time of case 0-1.

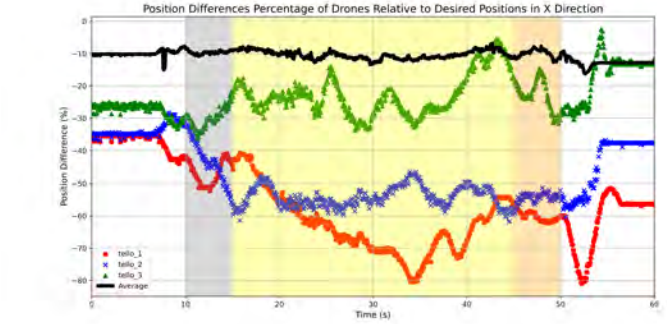
doi:10.6342/NTU202404031



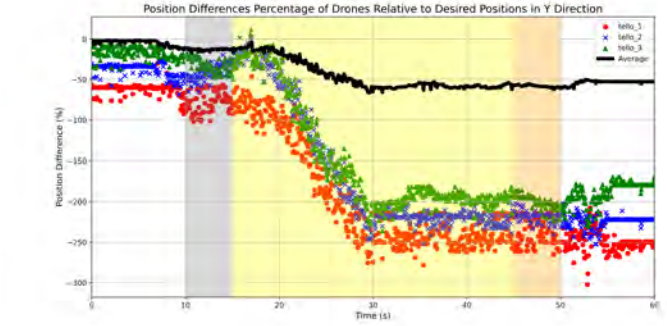
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



(c) Drone position errors by RGB camera estimation in X_W direction.



(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.43: The comparison of actual positions and errors by RGB camera estimation within time of case 0-2.

doi:10.6342/NTU202404031

0-1, as we can observe in Figure 5.43(c) and Figure 5.43(d).

In Figure 5.44, the target is moving fast along Y_W direction with nearly stationary in X_W direction, so the error in Y_W direction will drastically decrease to -80% without any feedback control command while in X_W direction it has not much difference than case 0-1 and 0-2, as we can observe in Figure 5.44(c) and Figure 5.44(d).

5.6.2 Results of 1-1 $\sim$ 1-3, Control Cases, without Obstacles

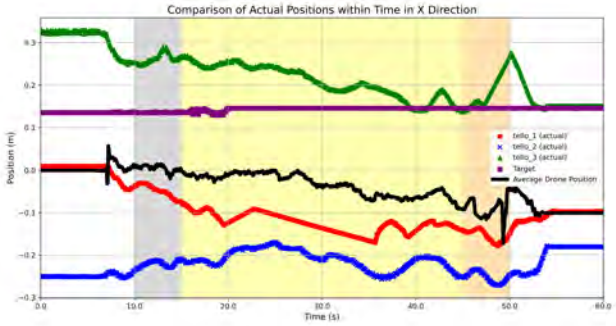
The purpose of this section is to demonstrate the scenario when there are no obstacles such as buildings or other architectures. Even if there are no obstacles, some quadrotors might not see the target. Nonetheless, we use the algorithms of the system to let the target be in FOV of the quadrotors again. We then compare the case with no obstacles (1-1$\sim$ 1-3). The results can be shown in Figure 5.45, Figure 5.46 and Figure 5.47.

We observe that for case 1-1 referring to Figure 5.45, the position of all the drones will not converge to the desired position. Instead, the positions of the drones will swing around the desired positions. It is because that the control strategy aims to make the drones converge to a desired positions as well as formation maintenance for group robustness [69: Ren and Beard 2007]. However, we can also observe the center of the triangular formation to verify the convergence of the entire formation. Other control cases including 1-2$\sim$ 1-3, 2-1$\sim$ 2-3 and 3-1$\sim$ 3-3 will obey this rule as well.

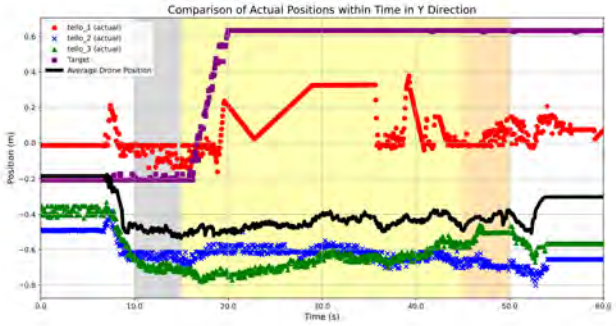
For case 1-1, we can observe that in X_W direction, the error of center of the formation will converge into $\pm$10% interval within 3s control at 18s and in Y_W direction, the error of center of the formation will converge into $\pm$10% interval within 4s control at 19s, as shown in Figure 5.45(c) and Figure 5.45(c).

For case 1-2 referring to Figure 5.46, we can observe that in X_W direction, the error

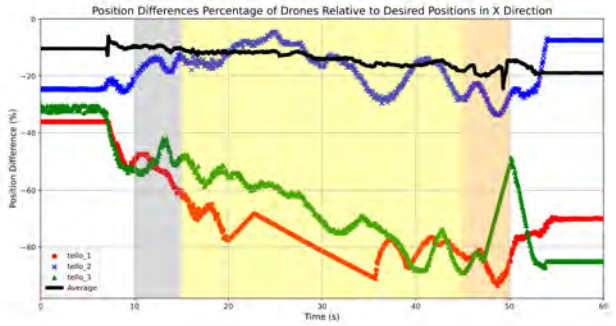
doi:10.6342/NTU202404031



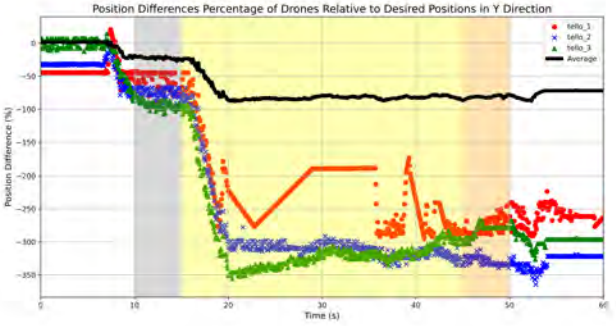
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



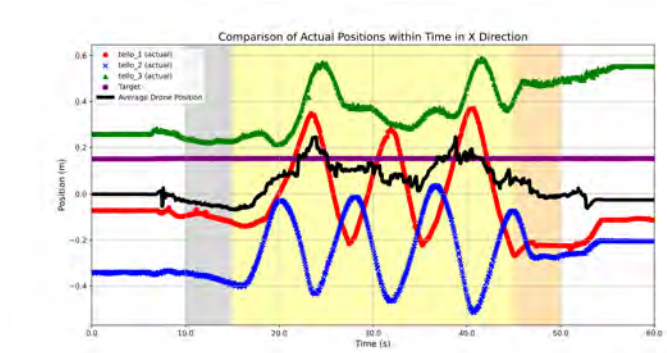
(c) Drone position errors by RGB camera estimation in X_W direction.



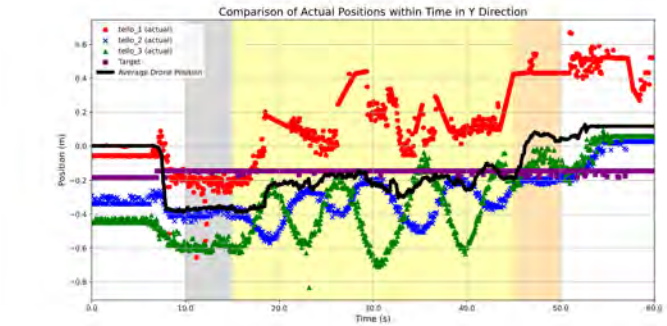
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.44: The comparison of actual positions and errors by RGB camera estimation within time of case 0-3.

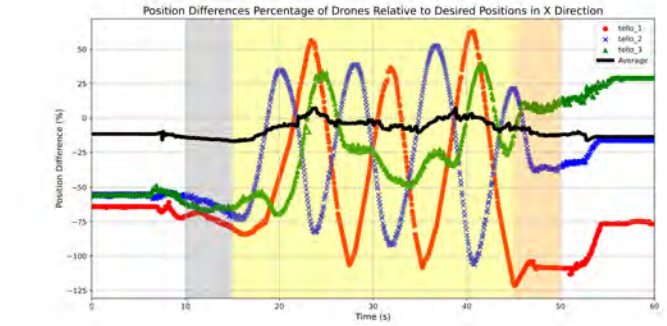
doi:10.6342/NTU202404031



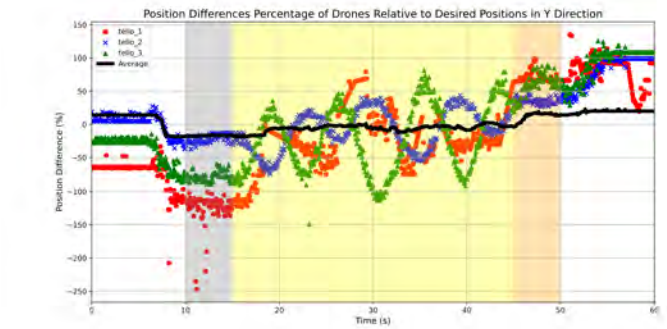
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



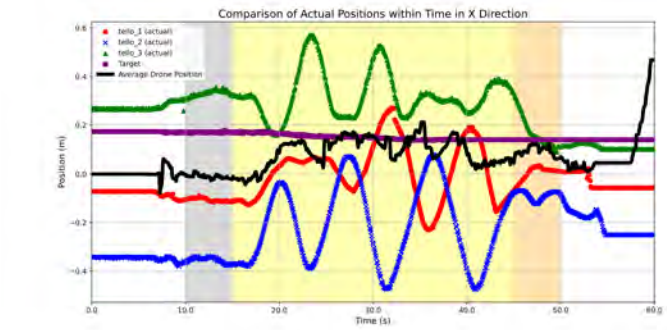
(c) Drone position errors by RGB camera estimation in X_W direction.



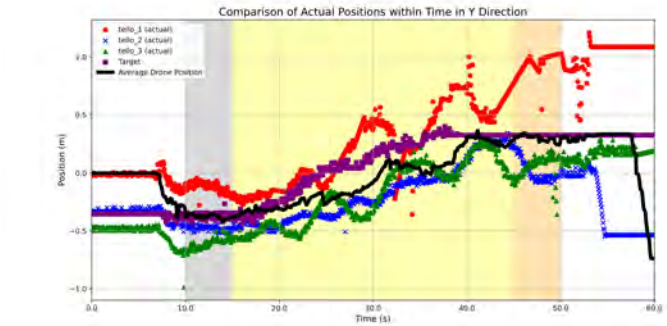
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.45: The comparison of actual positions and errors by RGB camera estimation within time of case 1-1.

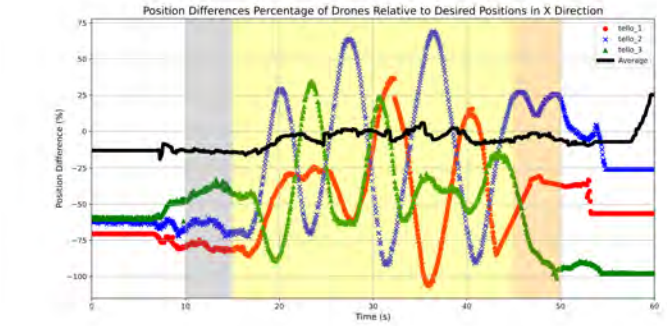
doi:10.6342/NTU202404031



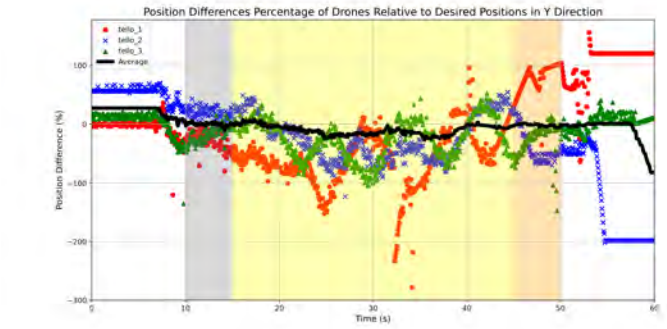
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



(c) Drone position errors by RGB camera estimation in X_W direction.



(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.46: The comparison of actual positions and errors by RGB camera estimation within time of case 1-2.

doi:10.6342/NTU202404031

of center of the formation will converge into $\pm 10\%$ interval within 5s control at 20s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 8s control at 23s, as shown in Figure 5.46(c) and Figure 5.46(c).

For case 1-3 referring to Figure 5.47, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 5s control at 20s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 12s control at 27s, as shown in Figure 5.47(c) and Figure 5.47(c).

5.6.3 Results of 2-1 $\sim$ 2-3, Control Cases, with an Obstacle

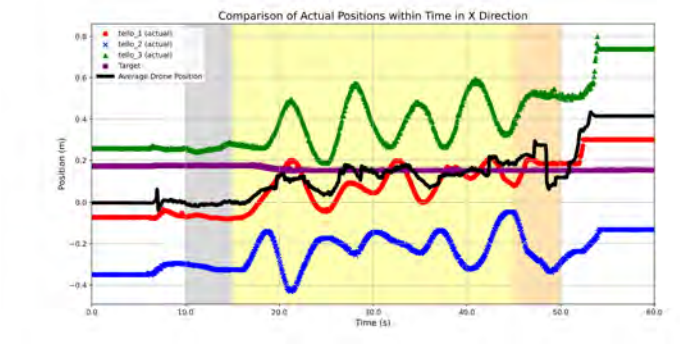
For the case of 2-1 $\sim$ 2-3 with an obstacle in the semi-complex environment, the results can be shown in Figure 5.48, Figure 5.49 and Figure 5.50.

For case 2-1 referring to Figure 5.48, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 3s control at 18s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 4s control at 19s, as shown in Figure 5.48(c) and Figure 5.48(c).

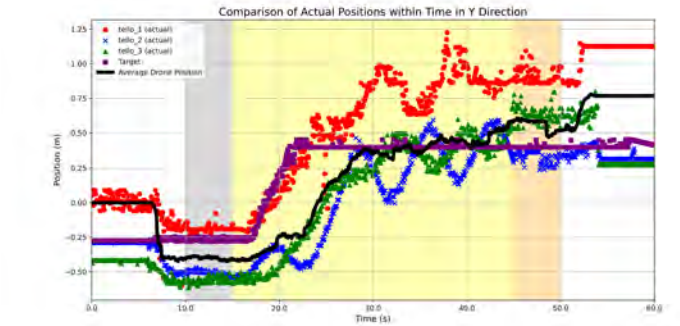
For case 2-2 referring to Figure 5.49, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 6s control at 21s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 7s control at 22s, as shown in Figure 5.49(c) and Figure 5.49(c).

For case 2-3 referring to Figure 5.50, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 8s control at 23s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within

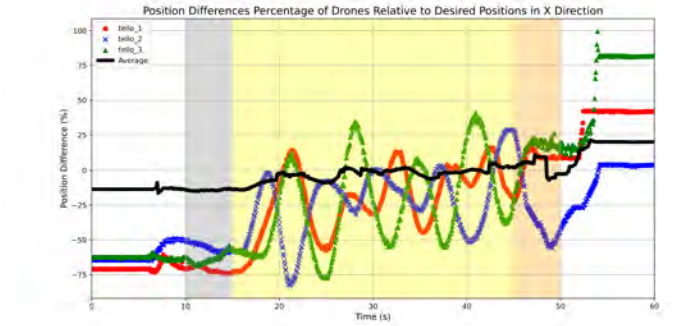
doi:10.6342/NTU202404031



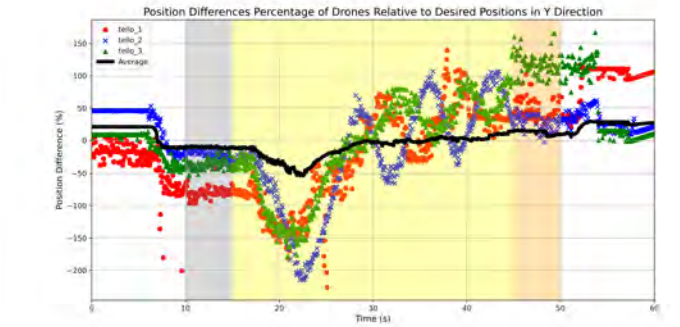
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



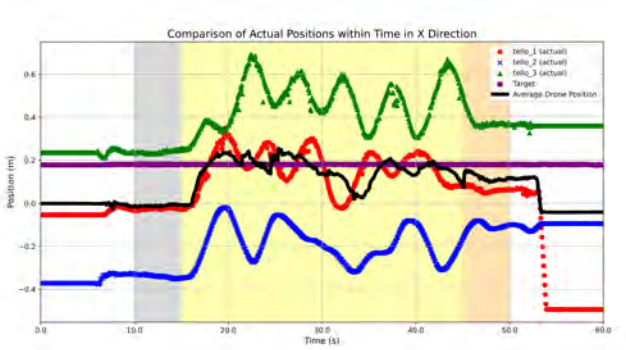
(c) Drone position errors by RGB camera estimation in X_W direction.



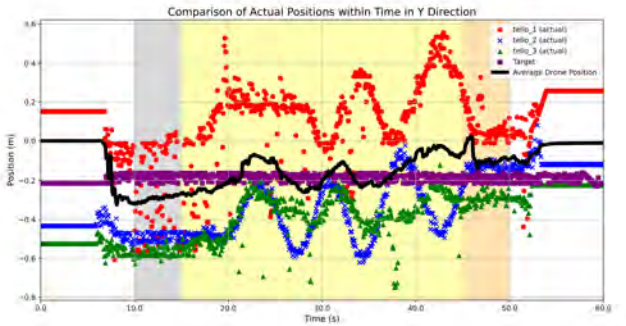
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.47: The comparison of actual positions and errors by RGB camera estimation within time of case 1-3.

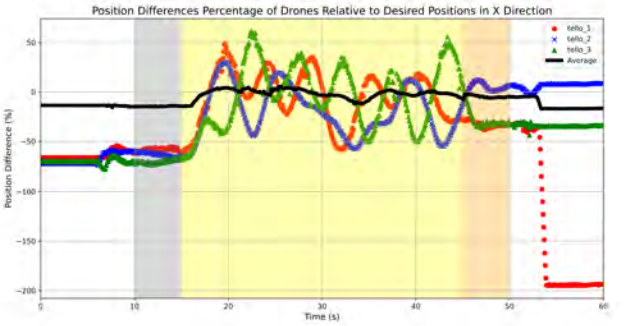
doi:10.6342/NTU202404031



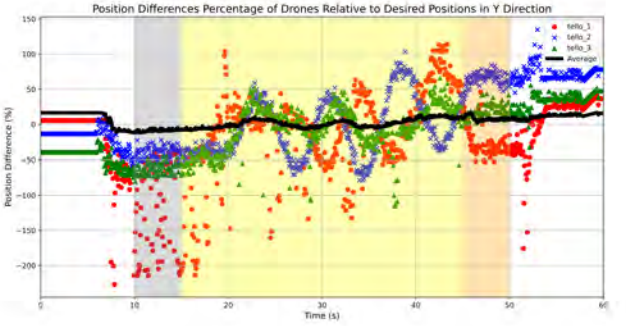
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



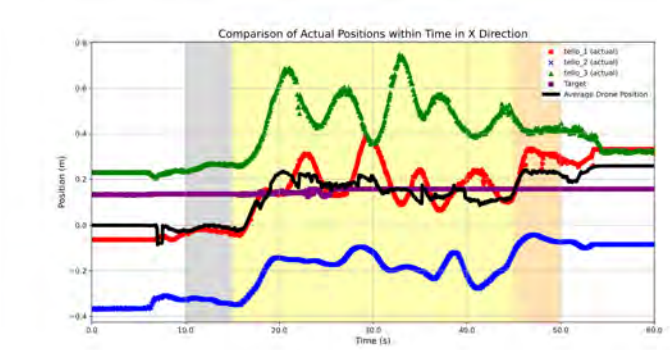
(c) Drone position errors by RGB camera estimation in X_W direction.



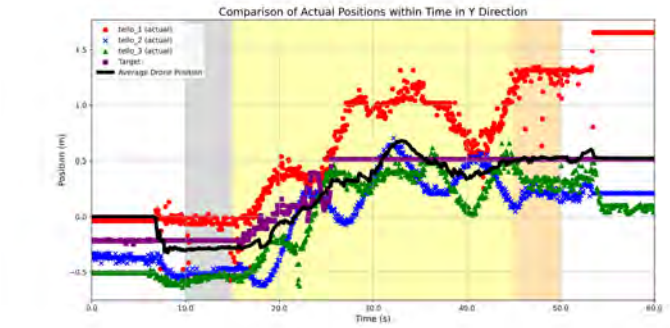
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.48: The comparison of actual positions and errors by RGB camera estimation within time of case 2-1.

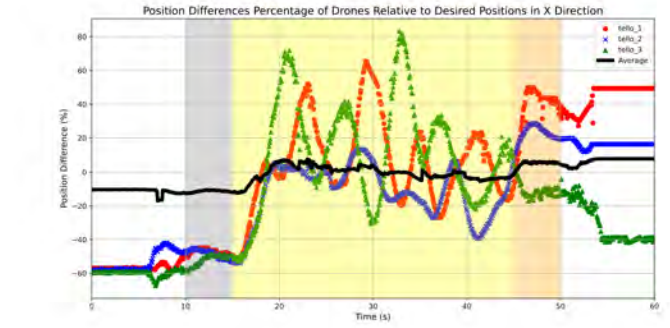
doi:10.6342/NTU202404031



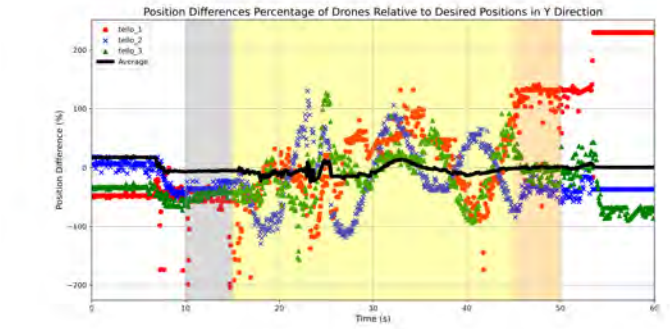
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



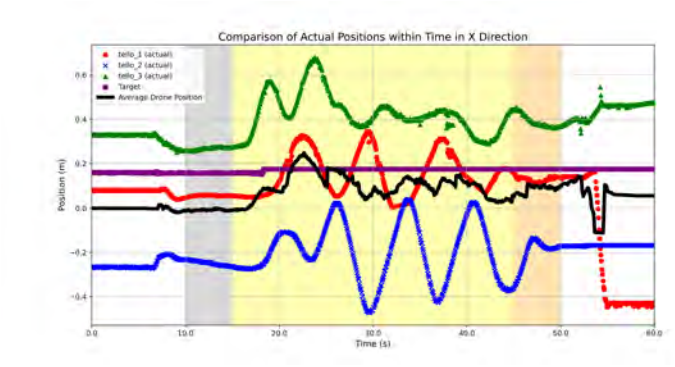
(c) Drone position errors by RGB camera estimation in X_W direction.



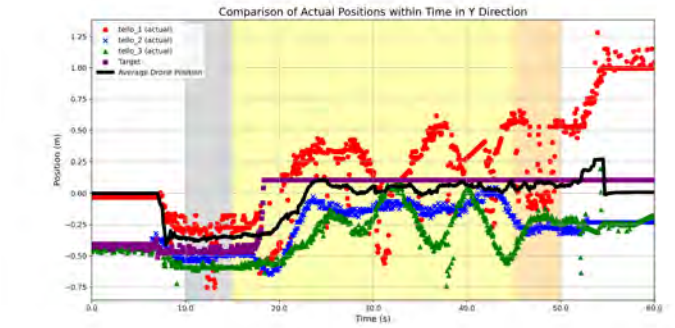
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.49: The comparison of actual positions and errors by RGB camera estimation within time of case 2-2.

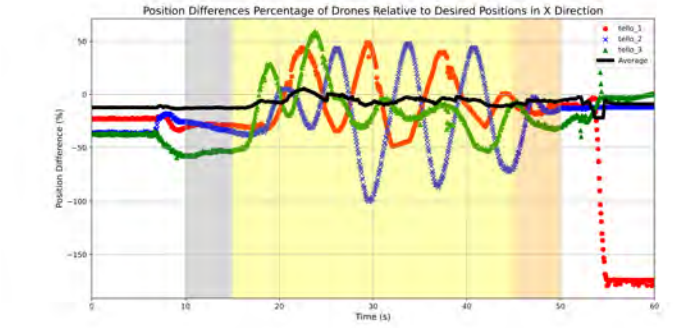
doi:10.6342/NTU202404031



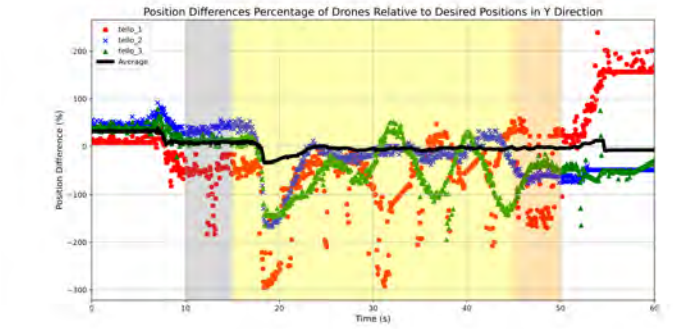
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



(c) Drone position errors by RGB camera estimation in X_W direction.



(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.50: The comparison of actual positions and errors by RGB camera estimation within time of case 2-3.

doi:10.6342/NTU202404031

8s control at 23s, as shown in Figure 5.50(c) and Figure 5.50(c).

5.6.4 Results of 3-1 $\sim$ 3-3, Control Cases, with three Obstacles

For the case of 3-1 $\sim$ 3-3 with multiple obstacles in the most complex environment, the results can be shown in Figure 5.51, Figure 5.52 and Figure 5.53.

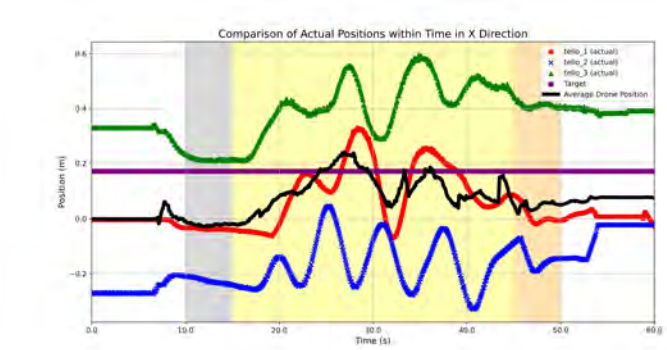
For case 3-1 referring to Figure 5.51, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 4s control at 19s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 3s control at 18s, as shown in Figure 5.51(c) and Figure 5.51(c).

For case 3-2 referring to Figure 5.52, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 4s control at 19s and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 4s control at 19s but with out of range at 15s$\sim$20s control at 30s$\sim$35s, as shown in Figure 5.52(c) and Figure 5.52(c). Nonetheless, the data will return into $\pm 10\%$ interval at the final of control process.

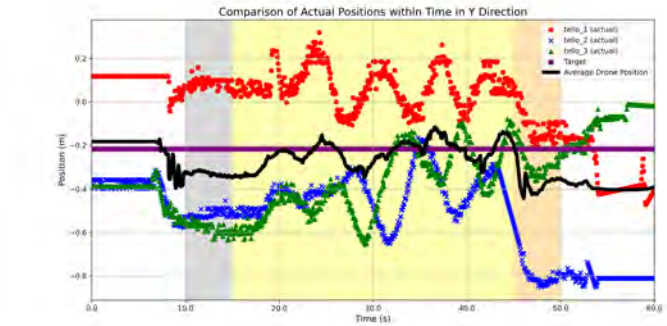
For case 3-3 referring to Figure 5.53, we can observe that in X_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 4s control at 19s but with out of range at 19s$\sim$24s control at 34s$\sim$39s, and in Y_W direction, the error of center of the formation will converge into $\pm 10\%$ interval within 18s control at 33s, as shown in Figure 5.53(c) and Figure 5.53(c). Note that the data will also return into $\pm 10\%$ interval at the final of control process.

In these cases from 3-1 to 3-3, there might be sometimes out of $\pm 10\%$ interval after converging such as Y_W direction in case 3-2 Figure 5.52(d) and X_W direction in case 3-3 Figure 5.53(c). We can compare the results in Figure 5.39(d) and Figure 5.40(c). There is

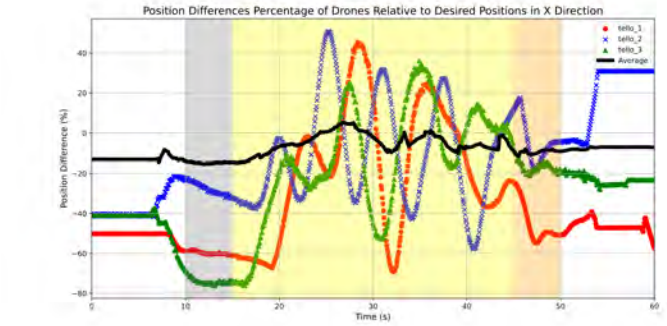
doi:10.6342/NTU202404031



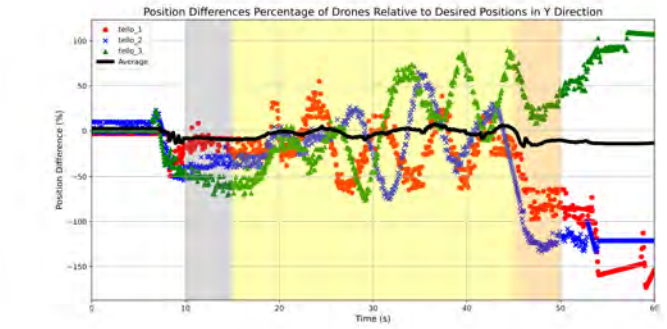
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



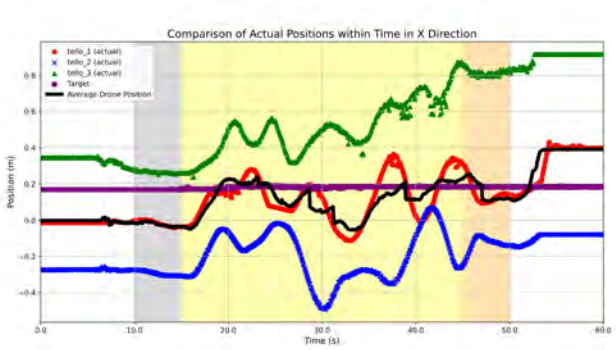
(c) Drone position errors by RGB camera estimation in X_W direction.



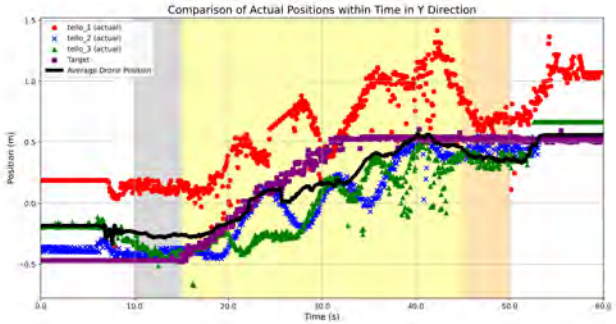
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.51: The comparison of actual positions and errors by RGB camera estimation within time of case 3-1.

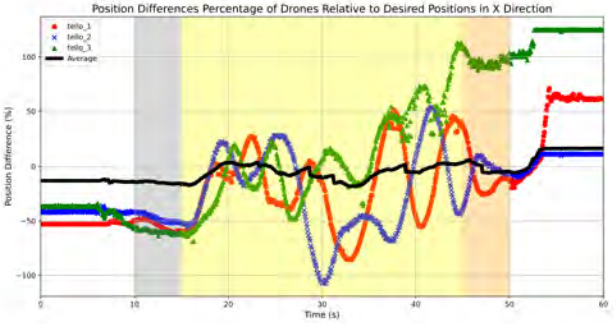
doi:10.6342/NTU202404031



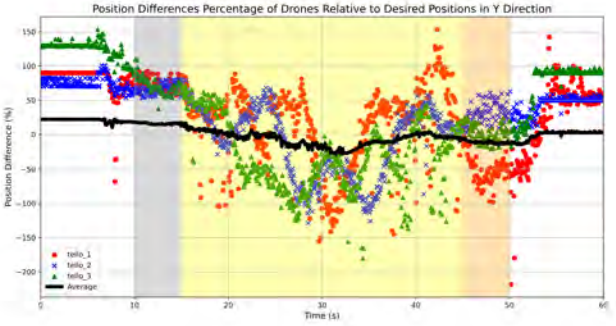
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



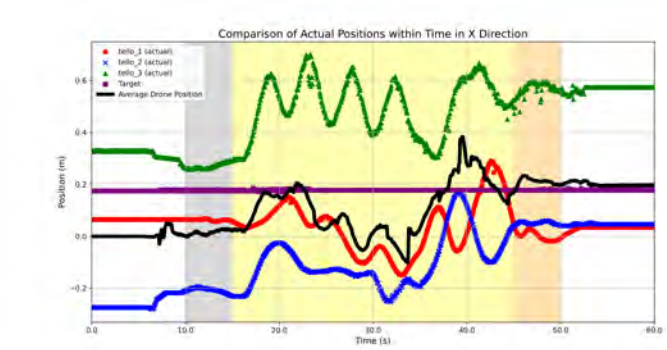
(c) Drone position errors by RGB camera estimation in X_W direction.



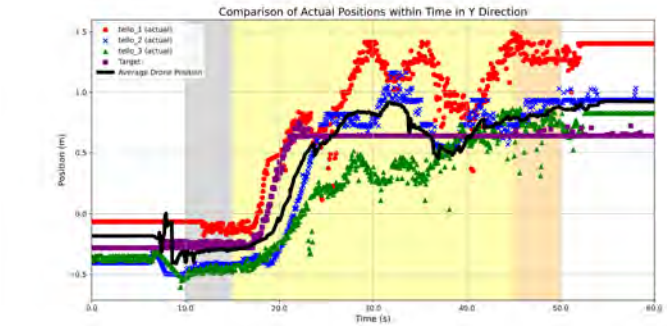
(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.52: The comparison of actual positions and errors by RGB camera estimation within time of case 3-2.

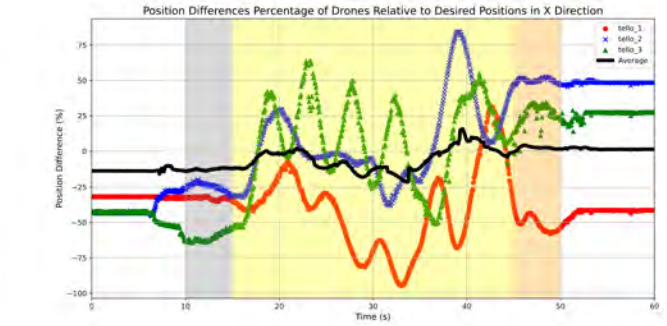
doi:10.6342/NTU202404031



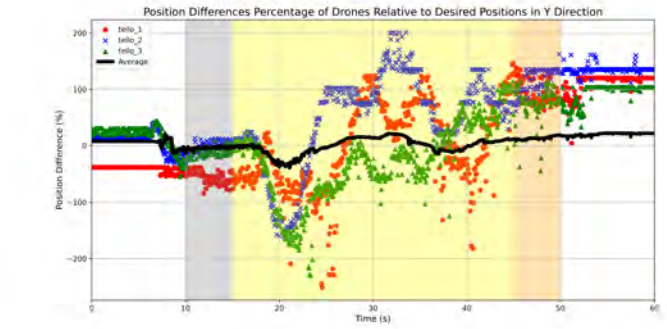
(a) Drone and target positions by RGB camera estimation in X_W direction.



(b) Drone and target positions by RGB camera estimation in Y_W direction.



(c) Drone position errors by RGB camera estimation in X_W direction.



(d) Drone position errors by RGB camera estimation in Y_W direction.

Figure 5.53: The comparison of actual positions and errors by RGB camera estimation within time of case 3-3.

doi:10.6342/NTU202404031

larger variation of estimation error than most of the cases during the certain time interval, which will make the positions out of range during control process. That is, for some cases with more occlusions, larger target estimation error will cause convergence error and make the formation center out of desired convergence range.

As shown in Table 5.2, in these cases from 1-1 to 3-3 for all control cases, we can observe that in X_W direction, since the target is nearly stationary, the time of convergence has not much difference, while in Y_W direction, larger target/quadrotors speed will cause larger control time of convergence obviously.

Table 5.2: Convergence time for all cases.

	No control		Control, no obs.		Control, 1 obs.		Control, 3 obs.	
Direction	X_W	Y_W	X_W	Y_W	X_W	Y_W	X_W	Y_W
Stationary	-	-	3s	4s	3s	4s	4s	3s
Slow	-	-	5s	8s	6s	7s	4s	4s
Fast	-	-	5s	12s	8s	8s	4s	18s

5.7 Command Velocities within Time

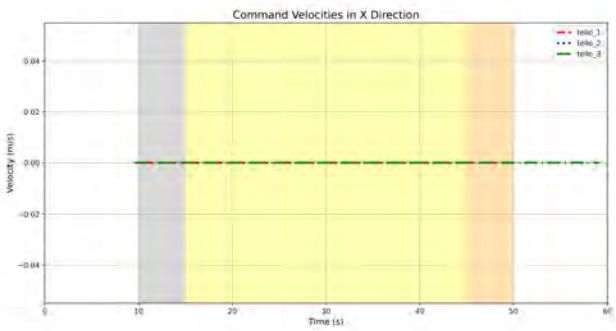
In this section, we demonstrate the command velocity u_{if} during the time interval of control process.

5.7.1 Results of 0-1 $\sim$ 0-3, Non-Control Cases, without Obstacles

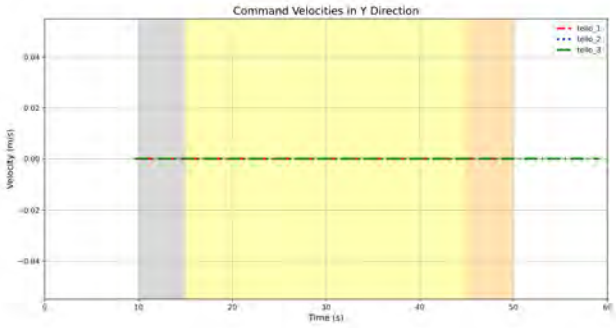
The results of 0-1 $\sim$ 0-3 are shown in Figure 5.54, Figure 5.55 and Figure 5.56 where the (red) dashed curves, (blue) dotted curves and (green) dashed-dotted curves are the command velocities (m/s) of tello_1, tello_2 and tello_3 respectively.

In Figure 5.54, Figure 5.55 and Figure 5.56, the result shows the command velocities in X_W and Y_W directions. Compared with Figure 5.42,Figure 5.43 and Figure 5.44, we can observe that there is not any nonzero command velocity during control interval. As a

doi:10.6342/NTU202404031

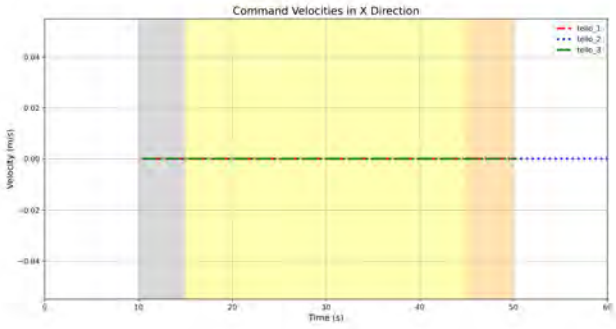


(a) Command Velocity in X_W direction.

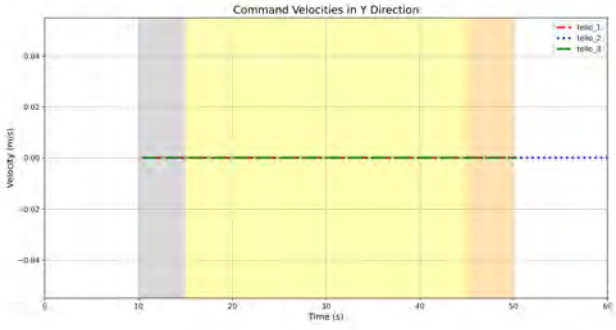


(b) Command Velocity in Y_W direction.

Figure 5.54: The comparison of command velocity within time of case 0-1.



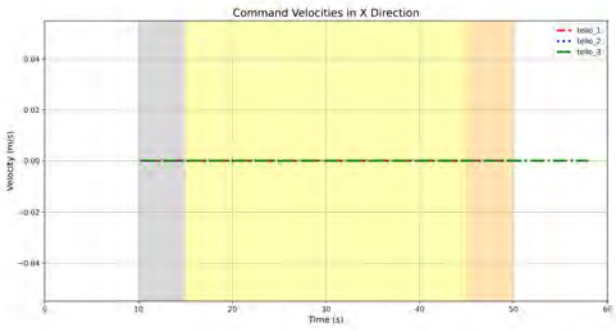
(a) Command Velocity in X_W direction.



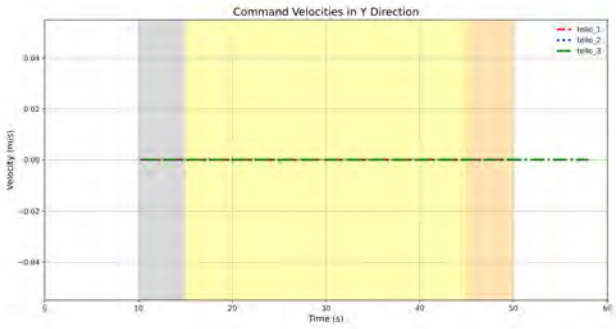
(b) Command Velocity in Y_W direction.

Figure 5.55: The comparison of command velocity within time of case 0-2.

doi:10.6342/NTU202404031



(a) Command Velocity in X_W direction.



(b) Command Velocity in Y_W direction.

Figure 5.56: The comparison of command velocity within time of case 0-3.

result, the drones will not move actively.

5.7.2 Results of 1-1 ∼ 1-3, Control Cases, without Obstacles

The results of 1-1 ∼ 1-3 for the control cases are shown in Figure 5.57, Figure 5.58 and Figure 5.59.

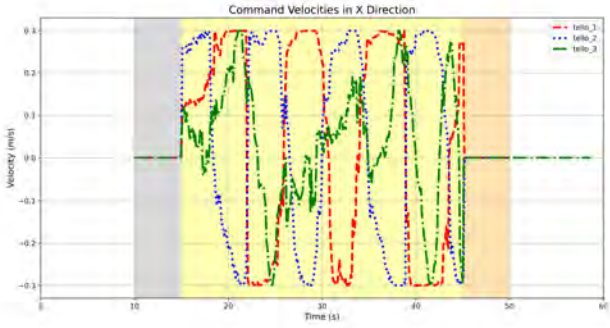
In Figure 5.57, Figure 5.58 and Figure 5.59, the result shows the command velocities in X_W and Y_W directions. Compared with Figure 5.45,Figure 5.46 and Figure 5.47, we can observe that the nonzero command only executes during control interval.

5.7.3 Results of 2-1 ∼ 2-3, Control Cases, with an Obstacle

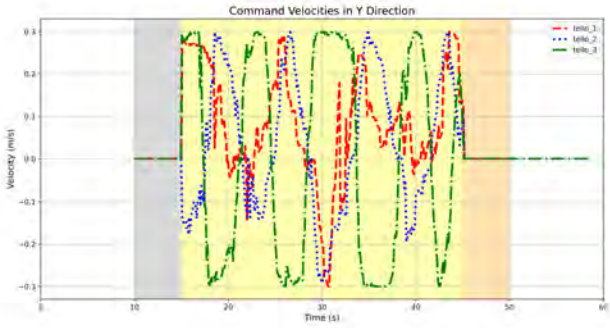
The results of 2-1 ∼ 2-3 for the control cases are shown in Figure 5.60, Figure 5.61 and Figure 5.62.

In Figure 5.60, Figure 5.61 and Figure 5.62, the result shows the command velocities

doi:10.6342/NTU202404031

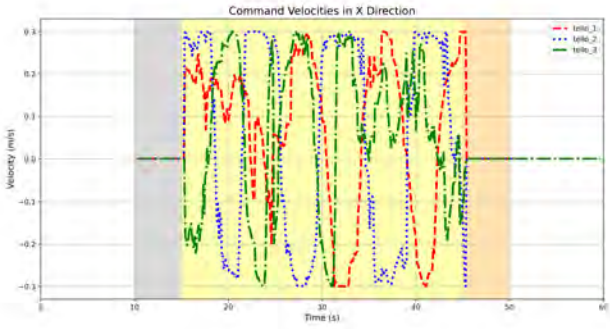


(a) Command Velocity in X_W direction.

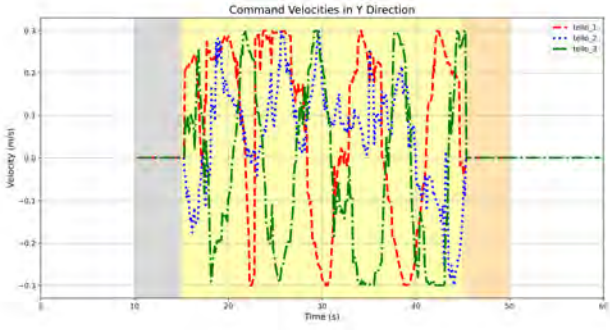


(b) Command Velocity in Y_W direction.

Figure 5.57: The comparison of command velocity within time of case 1-1.



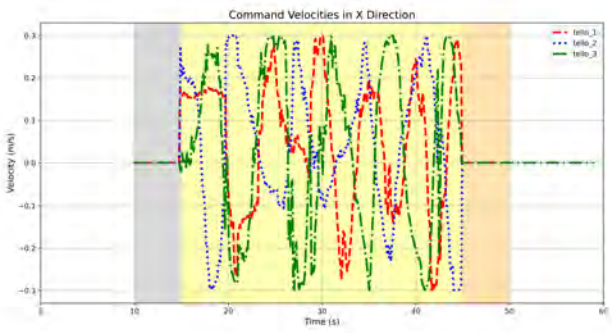
(a) Command Velocity in X_W direction.



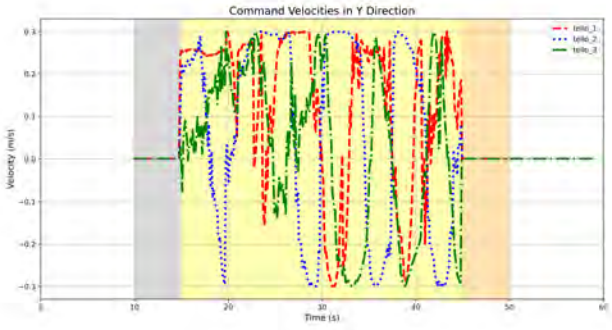
(b) Command Velocity in Y_W direction.

Figure 5.58: The comparison of command velocity within time of case 1-2.

doi:10.6342/NTU202404031

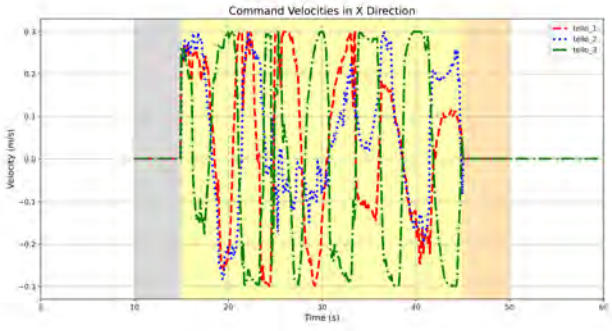


(a) Command Velocity in X_W direction.

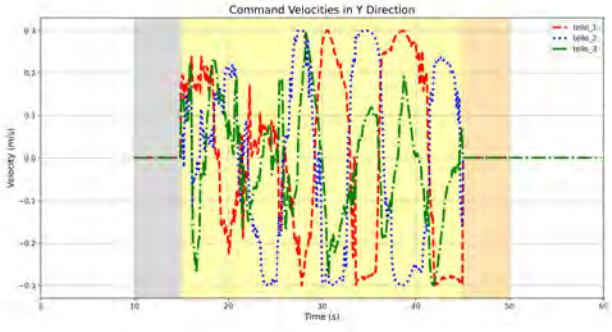


(b) Command Velocity in Y_W direction.

Figure 5.59: The comparison of command velocity within time of case 1-3.



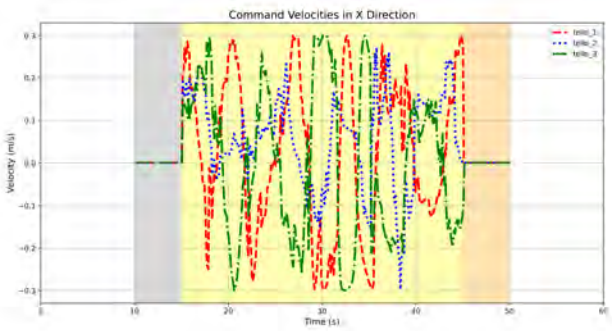
(a) Command Velocity in X_W direction.



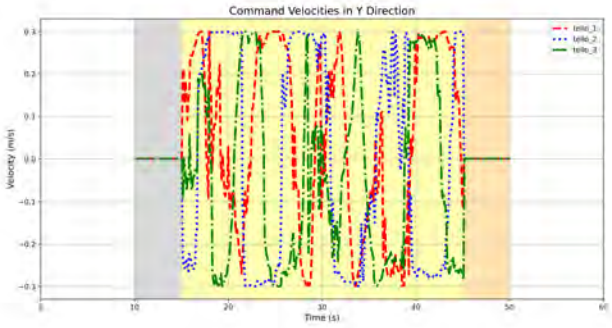
(b) Command Velocity in Y_W direction.

Figure 5.60: The comparison of command velocity within time of case 2-1.

doi:10.6342/NTU202404031

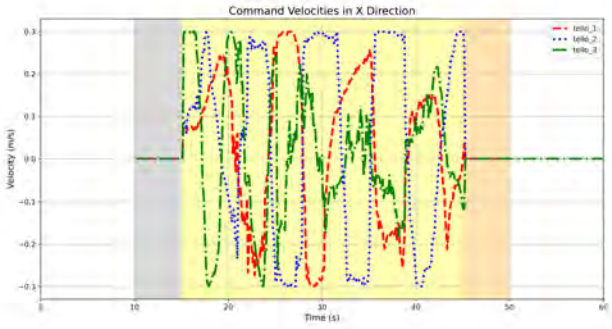


(a) Command Velocity in X_W direction.

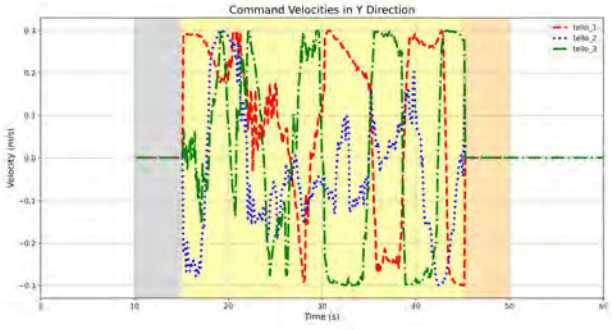


(b) Command Velocity in Y_W direction.

Figure 5.61: The comparison of command velocity within time of case 2-2.



(a) Command Velocity in X_W direction.



(b) Command Velocity in Y_W direction.

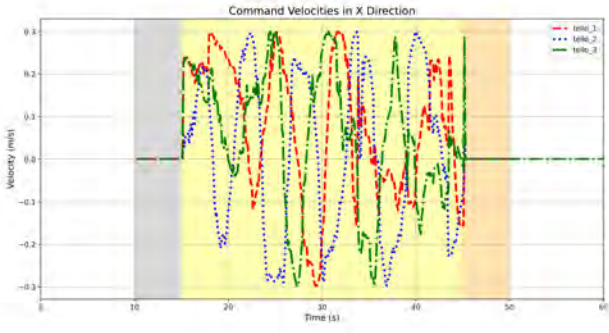
Figure 5.62: The comparison of command velocity within time of case 2-3.

doi:10.6342/NTU202404031

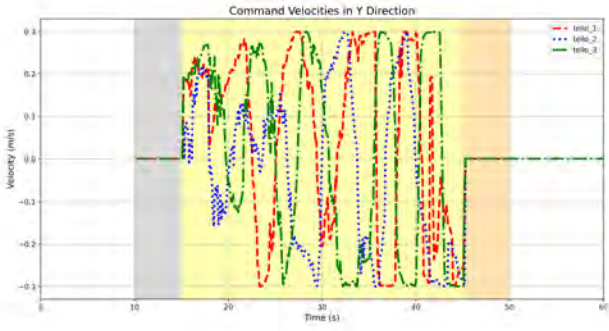
in X_W and Y_W directions. Similarly, compared with Figure 5.48,Figure 5.49 and Figure 5.50, we can observe that the nonzero command only executes during control interval.

5.7.4 Results of 3-1 ∼ 3-3, Control Cases, with three Obstacles

The results of 3-1 ∼ 3-3 for the control cases are shown in Figure 5.63, Figure 5.64 and Figure 5.65.



(a) Command Velocity in X_W direction.

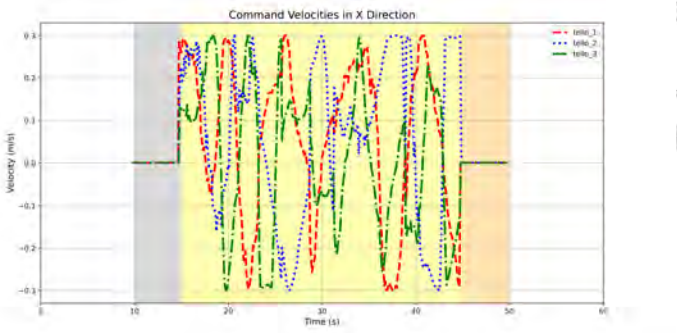


(b) Command Velocity in Y_W direction.

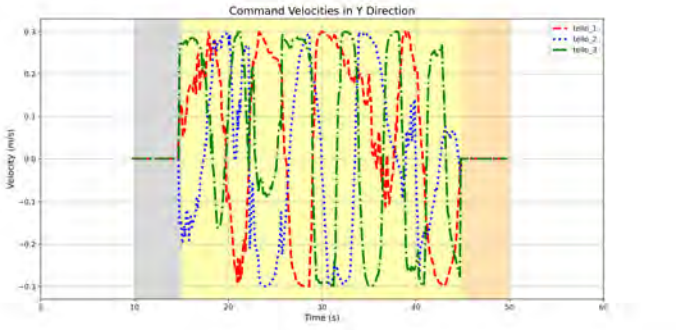
Figure 5.63: The comparison of command velocity within time of case 3-1.

In Figure 5.63, Figure 5.64 and Figure 5.65, the result shows the command velocities in X_W and Y_W directions. Similar to case 1-1 ∼ 1-3 and 2-1 ∼ 2-3, in 3-1 ∼ 3-3, compared with Figure 5.51,Figure 5.52 and Figure 5.53, we can observe that the nonzero command only executes during control interval.

doi:10.6342/NTU202404031

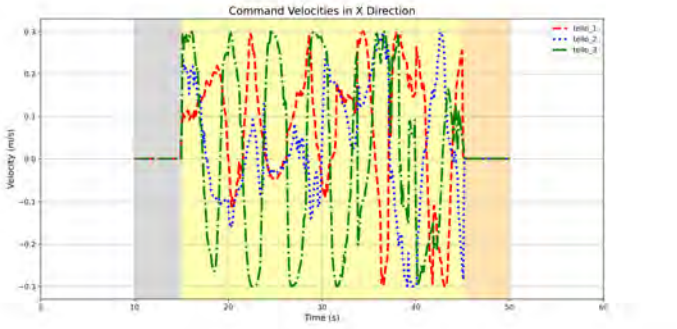


(a) Command Velocity in X_W direction.

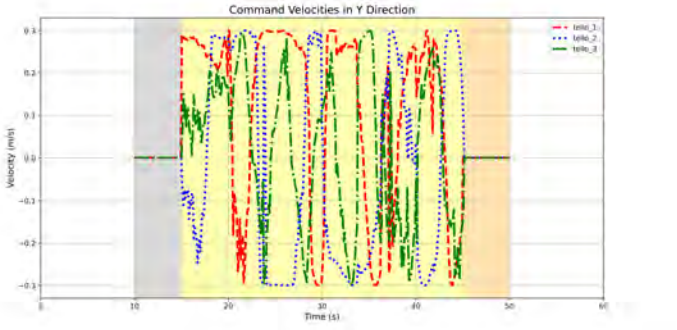


(b) Command Velocity in Y_W direction.

Figure 5.64: The comparison of command velocity within time of case 3-2.



(a) Command Velocity in X_W direction.



(b) Command Velocity in Y_W direction.

Figure 5.65: The comparison of command velocity within time of case 3-3.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this thesis, we perform the consensus-based multi-agent formation trajectory tracking toward a ground target with occlusion by obstacles or signal transmission failures.

We use the downward vision with landmarks as well as IMU velocity measurement to design an KF-based localization algorithm. Next, we estimate the target position according to the relative positions of the quadrotors and the target.

In order to solve the unpredicted estimation failures, we perform an motion prediction algorithm using sliding-window-based points selection for the last 10 or 9 estimation points as well as curve fitting in the world frame and obtain the trajectory of the target for each agent, which is a function of time. After obtaining the trajectory, we can obtain the predicted position of the target for each agent.

Nonetheless, each agent has its predicted target position, so we use the algorithm to select the reasonable states of the target and perform the averaging method to obtain the unique target reference state including the position and velocity.

Finally, according to the unique target reference state, we apply the variables into the consensus-based formation tracking controller to perform the objective of multi-agent control.

In order to demonstrate the capability of the system, we perform 1×3 non-control cases and 3×3 control cases from stationary to fast and from no obstacle to multiple obsta-

doi:10.6342/NTU202404031

cles for comparison which include onboard localization accuracy, estimation or prediction of the target without/with sliding-window-based curve fitting, predicted target position accuracy with range selection and averaging method, actual positions of the agents and the target using RGB camera from the third perspective and command velocities of the agents for validation of control command in the assigned time interval.

For onboard localization accuracy, because of the flight stability, non-control cases have the most accurate performance. Speed of the target has not obvious effect on the onboard localization accuracy for the contorl cases. Nonetheless, the distribution of error is in $\pm 20\%$ for most cases. As a result, this localization algorithm is suitable for all cases with and without control.

For estimation or prediction of the target without/with sliding-window-based curve fitting with predicted target position accuracy with averaging method to extract the unique target reference position, the predicted target position accuracy has not much difference in X_W direction (nearly stationary of the target) but increases from -15% $\sim$ 10% to -45% $\sim$ 75% as target speed increases in Y_W direction (from nearly stationary to fast speed of the target) for non-control cases of 0-1 to 0-3. In X_W direction of the control cases, occlusions by obstacles dominate the condition of accuracy since the speed is nearly zero in this direction. Nonetheless, it is not such obvious and has several exceptions. In Y_W direction of the control cases, target speed dominates the condition of accuracy obviously since the target and the formation of quadrotors is mainly moving along this direction. For the comparison of occlusions, the error increases from -5% $\sim$ 15% to -30% $\sim$ 15% for case 1-1 to 3-1 (stationary target) and -30% $\sim$ 10% to -25% $\sim$ 30% for case 1-2 to 3-2 (slow target). Nonetheless, for case 1-3 to 3-3, there is not much difference of the range of error. We can infer that as speed of the target increases to a certain value, the range of

doi:10.6342/NTU202404031

error will be not such obvious.

For the actual position error of the quadrotor formation center, we assign the position interval of convergence at $\pm 10\%$. For non-control cases of 0-1 to 0-3, the target is moving along Y_W direction, so the error has not much difference in X_W direction but increases according to the target speed from 0% $\sim$ -20% to 0% $\sim$ -80%. For the control cases in X_W direction, since the target is nearly stationary in this direction, the control time to converge into the $\pm 10\%$ interval is not such obvious. For the control cases in Y_W direction, the target speed dominates the condition of convergence time, which makes the control time of convergence from 4s to 12s for case 1-1 to 1-3, from 4s to 8s for case 2-1 to 2-3 and from 3s to 88s for case 3-1 to 3-3. Furthermore, more occlusions will also affect the error of convergence. Because of larger onboard prediction of the target at some time intervals, the actual position error might be out of $\pm 10\%$ range sometimes such as at 15s-20s control in case 3-2 in Y_W direction and at 19s-24s control in case 3-3 in X_W direction.

6.2 Future Works

In the future, there might be some issues we can focus on. For example, due to the hardware limitation, the motion of the quadrotors cannot be such prompt or it could cause the inter-collision of the agents because of the capability of calculation, transmission or velocity saturation of the command. Perhaps we can also apply the trajectory planning algorithm as mentioned in Appendix E on the better hardware one day.

On the other hand, there are a lot of researches that focus on the collision avoidance of the external obstacles, which can be one of our further research objectives.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

References

[1: Goldman Sachs 2016]

Goldman Sachs. "Drones: Flying into the Mainstream." (May 2016), [Online]. Available: https://www.goldmansachs.com/intelligence/pages/drones-flying-into-the-mainstream.html.

[2: Droneblog 2021]

Droneblog. "Which Drones Do the Police Use?" (Aug. 2021), [Online]. Available: https://www.droneblog.com/drones-police-use/.

[3: 華視新聞 CH52]

華視新聞 CH52, "北市無人機隊成立! 助"交通疏導.刑案偵查" | 華視新聞 20221003," [Online]. Available: %5Curl%7Bhttps://www.youtube.com/watch?v=lcBkOMOzKu0&t=16s%7D.

[4: Police 1 2018]

Police 1. "How to develop a police UAS training program." (May 2018), [Online]. Available: https://www.police1.com/police-training/articles/how-to-develop-a-police-uas-training-program-WxaxWcLRSz4buYp5/.

[5: 洪哲政 2023]

洪哲政, "無人機警監系統民間都說能作結果出人意料," *UDN*, 2023.

[6: Zhang 2019]

J. Zhang, "Occlusion-aware UAV Path Planning for Reconnaissance and Surveillance in Complex Environments," in *2019 IEEE International Conference on Robotics and Biomimetics*, Dali, China, 2019.

[7: Yoshida 2021]

Junko Yoshida, "自動駕駛技術的迫切挑戰：贏得大眾信任," *EE Times Taiwan*, 2021.

[8: Portugal 2020]

Steven J. Portugal, "Bird flocks," in *Current Biology Magazine*, 2020.

doi:10.6342/NTU202404031

[9: Tylus 2020]

Paulina Tylus, "2020: Fires, floods, pandemics, and now···locust plagues. WTF?!" In *Green is the New Black*, 2020.

[10: Logan Berry Heritage Farm 2018]

Logan Berry Heritage Farm, "Spring Is Honey Bee Swarm Season!," 2018.

[11: Young 2023]

Jerry Young, "All About the Hyena: Misunderstood Predators," 2023.

[12: Liu *et al.* 2022]

Daqian Liu, Xiaomin Zhu, Weidong Bao, Bowen Fei, and Jianhong Wu, "SMART: Vision-Based Method of Cooperative Surveillance and Tracking by Multiple UAVs in the Urban Environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 941–24 956, 2022.

[13: Wu, Wang, and Ying 2022]

Guohui Wu, Ning Wang, and Jin Ying, "Research on Distributed Real-Time Formation Tracking Control of High-Order Multi-UAV System," *IEEE Access*, vol. 10, pp. 36 286–36 298, 2022.

[14: Chen *et al.* 2022]

B. S. Chen, Y. C. Liu, M. Y. Lee, and C. L. Hwang, "Decentralized H PID Team Formation Tracking Control of Large-Scale Quadrotor UAVs Under External Disturbance and Vortex Coupling," *IEEE Access*, vol. 10, pp. 108 169–108 184, 2022.

[15: Xie and Lynch 2017]

Hui Xie and Alan F. Lynch, "Input Saturated Visual Servoing for Unmanned Aerial Vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 952–960, 2017.

[16: Murtaza's Workshop - Robotics and AI]

Murtaza's Workshop - Robotics and AI, "Drone Programming With Python Course | 3 Hours | Including x4 Projects | Computer Vision," [Online]. Available: https://www.youtube.com/watch?v=LmEcyQnfpDA&t=1326s&ab_channel=Murtaza%27sWorkshop-RoboticsandAI.

doi:10.6342/NTU202404031

[17: Qu and Zhang 2011]

Yaohong Qu and Youmin Zhang, "Cooperative localization against GPS signal loss in multiple UAVs flight," *Journal of Systems Engineering and Electronics*, vol. 22, no. 1, pp. 103–112, 2011.

[18: Ma *et al.* 2023]

Liqun Ma, Dongyuan Meng, Xu Huang, and Shuaihe Zhao, "Vision-Based Formation Control for an Outdoor UAV Swarm With Hierarchical Architecture," *IEEE Access*, vol. 11, pp. 75 134–75 151, 2023.

[19: Wang *et al.* 2022]

Dongjia Wang, Baowang Lian, Yangyang Liu, and Bo Gao, "A Cooperative UAV Swarm Localization Algorithm Based on Probabilistic Data Association for Visual Measurement," *IEEE Sensors Journal*, vol. 22, no. 20, pp. 19 635–19 644, 2022.

[20: Zheng, Xie, and Li 2023]

Yi Zheng, Yaqin Xie, and Jiamin Li, "Multi-UAV Collaboration and IMU Fusion Localization Method in Partial GNSS-Denied Scenarios," *IEEE Access*, vol. 11, pp. 105 499–105 512, 2023.

[21: Thomas *et al.* 2017]

J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous Flight for Detection, Localization, and Tracking of Moving Targets With a Small Quadrotor," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1762–1769, 2017.

[22: Li *et al.* 2019]

Yibing Li, Mingyang Jiu, Qian Sun, and Qianhui Dong, "An Adaptive Distributed Consensus Control Algorithm Based on Continuous Terminal Sliding Model for Multiple Quad Rotors' Formation Tracking," *IEEE Access*, vol. 7, pp. 173 955–173 967, 2019.

[23: NightjarOne]

NightjarOne, "Tello Localization: Odometry," [Online]. Available: https://youtu.be/WLbI0QtSdA0.

doi:10.6342/NTU202404031

[24: Nister, Naroditsky, and Bergen 2004]

D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2004.

[25: Tang, Chen, and Lin 2020]

S. Tang, Z. Chen, and W. Lin, "Visual inertial odometry," *Journal of Industrial Mechatronics*, p. 30, 2020.

[26: Liu and Shen 2017]

T. Liu and S. Shen, "High altitude monocular visual-inertial state estimation: Initialization and sensor fusion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, 2017.

[27: Weiss *et al.* 2013]

Stephan Weiss, Markus W. Achtelik, Simon Lynen, Michael C. Achtelik, Laurent Kneip, Margarita Chli, and Roland Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *J. Field Robot.*, vol. 30, no. 5, pp. 803–831, 2013.

[28: Weiss and Siegwart 2011]

S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011.

[29: Forster *et al.* 2017]

C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, 2017.

[30: Caruso *et al.* 2017]

D. Caruso, A. Eudes, M. Sanfourche, D. Vissière, and G. Le Besnerais, "A robust indoor/outdoor navigation filter fusing data from vision and magneto-inertial measurement unit," *Sensors*, vol. 17, no. 12, p. 2795, 2017.

[31: Mourikis and Roumeliotis 2007]

A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for

doi:10.6342/NTU202404031

vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2007.

[32: Palezieux, Nageli, and Hilliges 2016]

N. de Palezieux, T. Nageli, and O. Hilliges, "Duo-VIO: Fast, lightweight, stereo inertial odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Daejeon, Korea (South), 2016.

[33: Han, Kim, and Myung 2013]

S. B. Han, J. H. Kim, and H. Myung, "Landmark-Based particle localization algorithm for mobile robots with a fish-eye vision system," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1745–1756, 2013.

[34: Zhang, Zhang, and Dai 2012]

H. Zhang, L. Zhang, and J. Dai, "Landmark-Based Localization for Indoor Mobile Robots with Stereo Vision," in *2012 Second International Conference on Intelligent System Design and Engineering Application (ISDEA)*, Sanya, China, 2012.

[35: Salahuddin *et al.* 2011]

M.A. Salahuddin, A. Al-Fuqaha, V.B. Gavirangaswamy, and M. Anan M. Ljucovic, "An efficient artificial landmark-based system for indoor and outdoor identification and localization," in *2011 7th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Istanbul, Turkey, 2011.

[36: Popova and Liu 2016]

Marineda G. Popova and Hugh H. T. Liu, "Position-Based Visual Servoing for Target Tracking by a Quadrotor UAV," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, San Diego, USA, 2016, pp. 2092–2103.

[37: Zhao *et al.* 2020]

Wanbing Zhao, Hao Liu, Frank L. Lewis, Kimon P. Valavanis, and Xinlong Wang, "Robust Visual Servoing Control for Ground Target Tracking of Quadrotors," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1980–1987, 2020.

[38: Liu *et al.* 2021]

Yuzhen Liu, Ziyang Meng, Yao Zou, and Ming Cao, "Visual Object Tracking and

doi:10.6342/NTU202404031

Servoing Control of a Nano-Scale Quadrotor: System, Algorithms, and Experiments," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 344–360, 2021.

[39: Chen, Liu, and Shen 2016]

Jing Chen, Tianbo Liu, and Shaojie Shen, "Tracking a Moving Target in Cluttered Environments Using a Quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, Oct. 2016.

[40: Serra *et al.* 2015]

P. Serra, R. Cunha, T. Hamel, C. Silvestre, and F. Le Bras, "Nonlinear Image-Based Visual Servo Controller for the Flare Maneuver of Fixed-Wing Aircraft Using Optical Flow," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 570–583, 2015.

[41: Xie and Lynch 2017]

Hui Xie and Alan F. Lynch, "Input Saturated Visual Servoing for Unmanned Aerial Vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 952–960, 2017.

[42: Chesi and Shen 2012]

G. Chesi and T. Shen, "Conferring Robustness to Path-Planning for Image-Based Control," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 4, pp. 950–959, 2012.

[43: Malis, Chaumette, and Boudet 1999]

E. Malis, F. Chaumette, and S. Boudet, "2 1/2 D visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, 1999.

[44: Corke and Hutchinson 2001]

P. I. Corke and S. A. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, 2001.

[45: Chang 2020] Li-Yang Chang, "Formation Control for Multiple Unmanned Aerial Vehicles System with Fiducial Marker- Based Pose Estimation System," M.S. thesis, National Taiwan University, Taipei, Taiwan, 2020.

doi:10.6342/NTU202404031

[46: Wang 1991]

P. K. C. Wang, "Navigation strategies for multiple autonomous mobile robots moving in formation," *J. Robot. Syst.*, vol. 8, no. 2, pp. 177–195, 1991.

[47: Desai, Ostrowski, and Kumar 1998]

J.P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, Leuven, Belgium, 1998.

[48: Do and Pan 2007]

K. D. Do and J. Pan, "Nonlinear formation control of unicycle-type mobile robots," *Robot. Auton. Syst.*, vol. 55, no. 3, pp. 191–204, 2007.

[49: Do 2008]

K. D. Do, "Formation tracking control of unicycle-type mobile robots with limited sensing ranges," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 3, pp. 527–538, 2008.

[50: Yu *et al.* 2019]

J. Yu, X. Dong, Q. Li, and Z. Ren, "Practical time-varying output formation tracking for high-order multi-agent systems with collision avoidance, obstacle dodging and connectivity maintenance," *J. Franklin Inst.*, vol. 356, no. 12, pp. 5898–5926, 2019.

[51: Pang *et al.* 2021]

Z.H. Pang, C. B. Zheng, J. Sun, Q. L. Han, and G. P. Liu, "Distanceand velocity-based collision avoidance for time-varying formation control of second-order multi-agent systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 4, pp. 1253–1257, 2021.

[52: Beard, Lawton, and Hadaegh 2000]

R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A feedback architecture for formation control," in *Proc. of the 2000 American Control Conference*, Chicago, IL, USA, 2000.

[53: Kang, Xi, and Sparks 2000]

W. Kang, N. Xi, and A. Sparks, "Formation control of autonomous agents in 3D

doi:10.6342/NTU202404031

workspace," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, USA, 2000.

[54: Leonard and Fiorelli 2001]

N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proc. IEEE Conf. Decision and Control*, Orlando, FL, USA, 2001.

[55: Chen *et al.* 2021]

Q. Chen, Y. Sun, M. Zhao, and M. Liu, "Consensus-based cooperative formation guidance strategy for multiparafoil airdrop systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 4, pp. 2175–2184, 2021.

[56: Bae, Lim, and Ahn 2021]

Y.B. Bae, Y.H. Lim, and H.S. Ahn, "Distributed robust adaptive gradient controller in distance-based formation control with exogenous disturbance," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2868–2874, 2021.

[57: Liu and Huang 2021]

T. Liu and J. Huang, "Discrete-time distributed observers over jointly connected switching networks and an application," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1918–1924, 2021.

[58: Balch and Arkin 1998]

T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, 1998.

[59: Schneider-Fontan and Mataric 1998]

M. Schneider-Fontan and M. J. Mataric, "Territorial multirobot task division," *IEEE Trans. Robot. Autom.*, vol. 14, no. 5, pp. 815–822, 1998.

[60: Lawton, Beard, and Young 2003]

Jonathan R. T. Lawton, Randal W. Beard, and Brett J. Young, "A Decentralized Approach to Formation Maneuvers," *IEEE Trans. Robot. Autom.*, vol. 19, no. 5, pp. 933–941, 2003.

doi:10.6342/NTU202404031

[61: Lu 2020]

J. Lu, "Multi-Intelligent Vehicle Cooperative Formation Control Method based on Visual Navigation," in *2020 IEEE International Conference on Industrial Application of Artificial Intelligence*, Harbin, China, 2020.

[62: Huang *et al.* 2022]

J. Huang, Z. Ji, S. Xiao, C. Jia, P. Wang, and X. Wang, "Research on formation control of multi intelligent driving vehicles based on swarm motion," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference*, Chongqing, China, 2022.

[63: Er *et al.* 2024]

M. J. Er, H. Gong, Y. Liu, and T. Liu, "Intelligent Trajectory Tracking and Formation Control of Underactuated Autonomous Underwater Vehicles: A Critical Review," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 1, pp. 543–555, 2024.

[64: Ortiz, Gonçalves, and Cabrera 2017]

Luis Ortiz, Luiz Gonçalves, and Elizabeth Cabrera, "A Generic Approach for Error Estimation of Depth Data from (Stereo and RGB-D) 3D Sensors," May 2017.

[65: Tsai 1987]

R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987. DOI: 10.1109/JRA.1987.1087109.

[66: Welch and Bishop 2006]

Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter," Jul. 2006.

[67: Wikipedia]

Wikipedia, "Convex function," [Online]. Available: %5Curl%7Bhttps://en.wikipedia.org/wiki/Convex_function%7D.

doi:10.6342/NTU202404031

[68: Wu 2019] Jui-Che Wu, "Person Surveillance System by Unmanned Aerial Vehicles Using Image-Based Visual Servoing Control," M.S. thesis, National Taiwan University, Taipei, Taiwan, 2019.

[69: Ren and Beard 2007]

Wei Ren and Randal W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*. Communications and Control Engineering, Jan. 2007.

[70: eBay]

eBay, "Bed Spectacles Horizontal Reading Lying Down Watching TV Lazy Prism Eye Glasses," [Online]. Available: %5Curl%7Bhttps://www.ebay.com/itm/Bed-Spectacles-Horizontal-Reading-Lying-Down-Watching-TV-Lazy-Prism-Eye-Glasses-/303389060572%7D.

[71: Works-Of-Claye]

Works-Of-Claye, "Tello Mirror Clip," [Online]. Available: %5Curl%7Bhttps://www.thingiverse.com/thing:2911427%7D.

[72: ArUco markers generator!]

ArUco markers generator!, [Online]. Available: %5Curl%7Bhttps://chev.me/arucogen/%7D.

[73: Chadha 2018]

Harveen Singh Chadha, "Extended Kalman Filter: Why do we need an Extended Version?" In *Towards Data Science*, 2018.

[74: Wikipedia]

Wikipedia, "Kalman filter," [Online]. Available: %5Curl%7Bhttps://en.wikipedia.org/wiki/Kalman_filter%7D.

[75: Chadha 2018]

Harveen Singh Chadha, "The Unscented Kalman Filter: Anything EKF can do I can do it better!" In *Towards Data Science*, 2018.

doi:10.6342/NTU202404031

[76: Du *et al.* 2023]

Y. Du, P. Huang, Y. Cheng, Y. Fan, and Y. Yuan, "Fault Tolerant Control of a Quadrotor Unmanned Aerial Vehicle Based on Active Disturbance Rejection Control and Two-Stage Kalman Filter," *IEEE Access*, vol. 11, pp. 67 556–67 566, 2023.

[77: Yang *et al.* 2023]

B. Yang, E. Yang, L. Yu, and C. Niu, "Adaptive Extended Kalman Filter-Based Fusion Approach for High-Precision UAV Positioning in Extremely Confined Environments," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 1, pp. 543–554, 2023.

[78: Guo *et al.* 2021]

P. Guo, J. Li, T. Chen, and Z. Wu, "Heave Motion Estimation Based on Cubature Kalman Filter," in *2021 International Conference on Cyber-Physical Social Intelligence (ICCSI)*, Beijing, China, 2021, pp. 1–5.

[79: Wikipedia]

Wikipedia, "Mathematical optimization," [Online]. Available: %5Curl%7Bhttps://en.wikipedia.org/wiki/Mathematical_optimization%7D.

[80: Wikipedia]

Wikipedia, "Sequential quadratic programmimg," [Online]. Available: %5Curl%7Bhttps://en.wikipedia.org/wiki/Sequential_quadratic_programming%7D.

[81: Johnson 2007]

Steven G. Johnson, "The NLopt nonlinear-optimization package," 2007.

[82: OpenCV]

OpenCV, "Perspective-n-Point (PnP) pose computation," [Online]. Available: %5Curl%7Bhttps://docs.opencv.org/3.4/d5/d1f/calib3d_solvePnP.html%7D.

[83: Wikipedia]

Wikipedia, "Proportional – integral – derivative controller," [Online]. Available: %5Curl%7Bhttps://en.wikipedia.org/wiki/Proportional%E2%80%93integral%E2%80%93derivative_controller%7D.

doi:10.6342/NTU202404031

[84: koide3]

koide3, "hdl_localization," [Online]. Available: %5Curl%7Bhttps://github.com/koide3/hdl_localization%7D.

[85: ROS.org]

ROS.org, "Getting Started with the Velodyne VLP16," [Online]. Available: %5Curl%7Bhttps://wiki.ros.org/velodyne/Tutorials/Getting%20Started%20with%20the%20Velodyne%20VLP16%7D.

[86: Hot Robotics]

Hot Robotics, "Velodyne VLP-16 LiDAR," [Online]. Available: %5Curl%7Bhttps://hotrobotics.co.uk/equipment/velodyne-vlp-16-lidar/%7D.

doi:10.6342/NTU202404031

Appendix A

Image Processing for Target and Landmark Detection with Calibrated Downward Vision

A.1 Downward Vision Processing

The quadrotor "Tello" has a front camera and a downward camera. In SDK, we can switch the camera we want to use. However, we cannot use the downward camera while using ROS which we use for communication. As a result, we need to deal with the field of view (FOV) problem. As shown in Figure A.1, we modify the vision angle using a mirror to reflect the image. The idea comes from the periscope which is common used in our life.

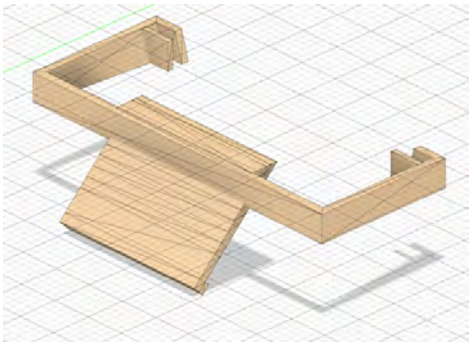
(a) The schematic concept for downward vision acquisition.

(b) Real product in the market eBay]. [70:

Figure A.1: The concept of downward vision acquisition.

We use a structure with a mirror to reflect the image, as shown in Figure A.2. Nonetheless, this method will make the image coordinate opposite. We flip the image by using OpenCV library as well, as shown in Figure A.3. We can use the processed image to go

doi:10.6342/NTU202404031



(a) The CAD of the structure [71: Works-Of-Claye].

(b) Installation on Tello.

Figure A.2: The appearance of the structure with Tello.

(a) Original image (before).

(b) Flipped image (after).

Figure A.3: Image before and after flipping by OpenCV.

on the further process.

A.2 Undistortion of the Image in Camera Calibration

In chapter 3, we introduce the pinhole model of a camera. There are some parameters we want to calibrate. As a result, in this section we will introduce the method for calibration of camera parameters.

We use OpenCV with a checkerboard to obtain the parameters. Checkerboard is a pattern with black and white grids. It is common in the applicaiton of camera calibration. As shown in Figure A.4, we take multiple (usually greater than 20) pictures of the checkerboard from different distances and angles using the camera we want to calibrate. By calculating the 2D distances of the corners of the grids, we can obtaint the parameters

doi:10.6342/NTU202404031

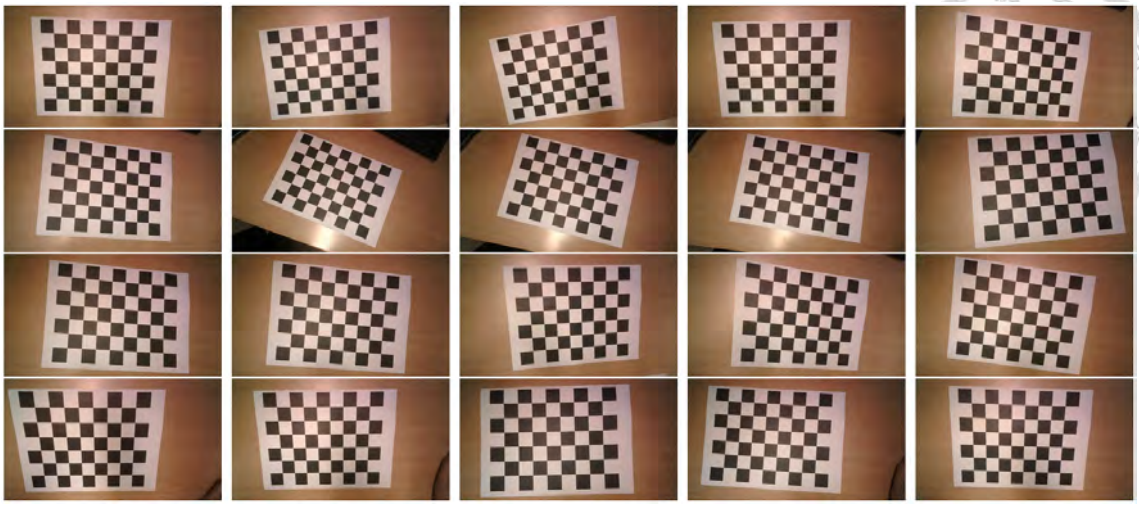


Figure A.4: The images of a checkboard from different distances angles.

of the specific camera.

In this framework, we assume we can take the pictures from all distances and angles. Nonetheless, by during this method, the result may not be such ideal. As a result, we only use the information of the parameters for solving the distortion problem. After undistortion, the comparison can be shown in Figure A.5. We can see after the adjustment, we

(a) Distorted image (before). **(b) Undistorted image (after).**

Figure A.5: The result before and after distortion adjustment.

can solve the distortion of the image, especially around the edges of the image. We can observe from the upper right of the images.

A.3 Target and Landmark Detection Using ArUco Markers

doi:10.6342/NTU202404031

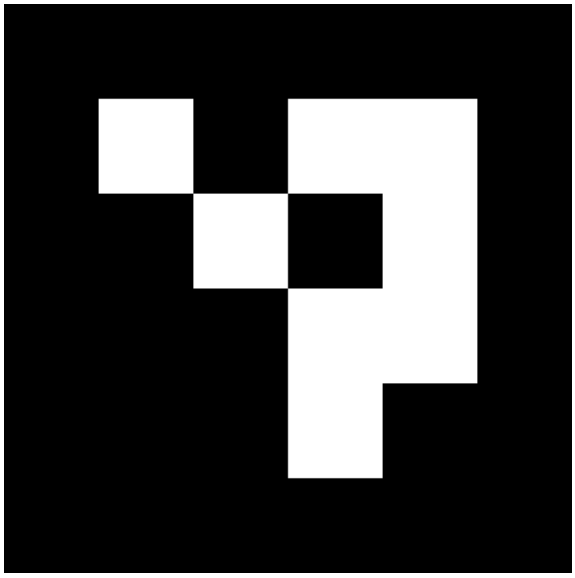


Figure A.6: ArUco marker of ID = 0.

We use ArUco markers as the object we want to detect. ArUco marker is a type of fudicial marker of two-dimensional barcode and usually can be used for camera pose estimation and localization in computer vision. They were developed by Rafael Muñoz and others in 2014 and are part of the OpenCV library, making them easy to generate and detect. As shown in Figure A.6 [72: ArUco markers generator!], it has four corners as feature points. We can obtain the center position by averaging the positions of four corners.

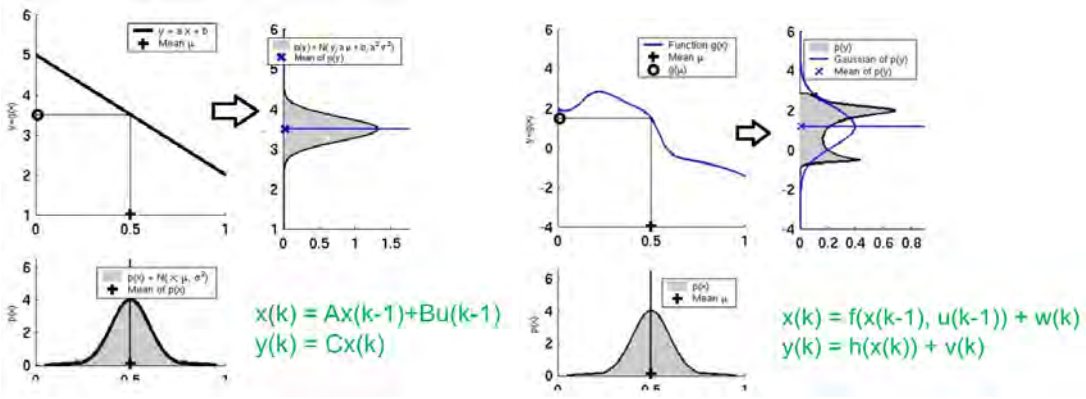
In the OpenCV library, we can easily detect the ArUco markers by using the dictionary inside it. By this framework, we can obtain the pixel values of the ArUco position in the 2D image.

doi:10.6342/NTU202404031

Appendix B

Introduction of Nonlinear Kalman Filters

For some cases, the traditional linear Kalman filter has the limitation that the algorithm needs to be under the assumption that the noise, state and measurement are all Gaussian distribution. As shown in Figure B.1 [73: Chadha 2018], in the linear system, a Gaussian distribution after the transformation is also a Gaussian distribution, but if the system is nonlinear, the transformation will not be a Gaussian distribution, which will make the algorithm invalid. As a result, there are some modified Kalman filter algorithms we can consider for nonlinear systems. These algorithms will be discuss in this chapter.



(a) Linear system case. **(b) Nonlinear system case.**

Figure B.1: The linear and nonlinear system transformation.

B.1 Extended Kalman Filter (EKF) Using Linearization

doi:10.6342/NTU202404031

In a nonlinear system, the motion model and the sensor model can be represented as the continuously differentiable nonlinear functions f and h. They can be defined as

$$x_k = f(x_{k-1}, u_k, w_k), \tag{B.1}$$

$$z_k = h(x_k, v_k), \tag{B.2}$$

where w_k and v_k are the term of the measurement and sensor noises respectively [74: Wikipedia].

Extended Kalman Filter (EKF) is one of the modified Kalman filter for nonlinear system by linearization of the nonlinear function. We can define the state transformation Jacobian matrix $A(k)$ and the Jacobian matrix of the observation model $H(k)$, which can be defines as

$$A(k) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}, \tag{B.3}$$

$$H(k) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \frac{\partial h_m}{\partial x_2} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}, \tag{B.4}$$

B.2 Unscented Kalman Filter (UKF) Using Points Fitting

Besides of extended Kalman filter (EKF), there are also several kinds of nonlinear

doi:10.6342/NTU202404031

Kalman filters. In this chapter, we introduce the Unscented Kalman Filter (UKF). It is also a modified Kalman filter for the nonlinear system. Unlike EKF, UKF does exact function transformation instead of linear approximation [75: Chadha 2018]. Similar to traditional Kalman filter, UKF has the prediction and correction process. In the prediction process, we first consider the set of sigma points for a more precise approximation.

$$\chi^{[0]} = \mu, \tag{B.5}$$

$$\chi^{[i]} = \mu + (\sqrt{(n+\lambda)\Sigma})_i \quad \text{for } i = 1, \ldots, n, \tag{B.6}$$

$$\chi^{[i]} = \mu - (\sqrt{(n+\lambda)\Sigma})_{i-n} \quad \text{for } i = n+1, \ldots, 2n, \tag{B.7}$$

where χ is the sigma points matrix, μ is the mean of the Gaussian, n is the dimensionality of the system, λ is the scaling factor, and Σ is the covariance matrix. Simultaneously, we compute the weights of the sigma points

$$\omega^{[0]} = \frac{\lambda}{n+\lambda}, \tag{B.8}$$

$$\omega^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ldots, 2n, \tag{B.9}$$

and predicted mean and covariance of the approximate Gaussian

$$\mu' = \sum_{i=0}^{2n} \omega^{[i]} g(\chi^{[i]}), \tag{B.10}$$

$$\Sigma' = \sum_{i=0}^{2n} \omega^{[i]} \left(g(\chi^{[i]}) - \mu'\right) \left(g(\chi^{[i]}) - \mu'\right)^T + R_t, \tag{B.11}$$

where R_t is the noise covariance matrix.

doi:10.6342/NTU202404031

In the correction step, we calculate the matrices in the measurement space.

$$Z = h(\chi), \tag{B.12}$$

$$\hat{z} = \sum_{i=0}^{2n} \omega^{[i]} Z^{[i]}, \tag{B.13}$$

$$S = \sum_{i=0}^{2n} \omega^{[i]} (Z^{[i]} - \hat{z})(Z^{[i]} - \hat{z})^T + Q, \tag{B.14}$$

where Z is the sigma points after transformation, $\hat{z}$ is the mean, S is the covariance in measurement. As a result, we can calculate the cross correlation matrix T and obtain the Kalman gain K.

$$T = \sum_{i=0}^{2n} \omega^{[i]} (\chi^{[i]} - \mu')(Z^{[i]} - \hat{z})^T, \tag{B.15}$$

$$K = TS^{-1}. \tag{B.16}$$

Finally we can calculate mean of the Gaussian μ and obtain the corrected covariance Σ.

$$\mu = \mu' + K(z - \hat{z}), \tag{B.17}$$

$$\Sigma = (I - KT)\Sigma'. \tag{B.18}$$

Consequently, UKF algorithms use multiple sigma points for transformation. Therefore, although it is more complex in the math interpretation, it is more precise while calculating the Kalman filter algorithms with similar computation.

B.3 Other Kinds of Kalman Filters

In this section, we introduce several the most common kinds of Kalman filters. As a matter of fact, there are many kinds of Kalman filters and their applications. For instance, in [76: Du *et al.* 2023], the authors proposed a Two-Stage Kalman Filter (TSKF) for

doi:10.6342/NTU202404031

state estimation, in [77: Yang *et al.* 2023], the authors proposed an Adaptive Extended Kalman Filter (AEKF) to fuse the UWB (ultrawideband) and IMU data for high-precision positioning of a UAV, and in [78: Guo *et al.* 2021], the authors use Cubature Kalman Filter (CKF) for heave motion estimation. As a result, Kalman filter is an vital algorithm we can consider in the state estimation or sensor fusion framework.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Appendix C

Introduction of Other Mathematical Optimization Problems

C.1 Convex Optimization Problems

C.1.1 Linear Programming (LP)

Linear programming (LP) is the problem when the objective function and inequality constraints are all linear, as shown below.

$$\begin{aligned} \min_{x} \quad & c_{lp}^T x + d_{lp} \\ \text{s.t.} \quad & G_{lp} x \preceq h_{lp} \\ & A_{lp} x = b_{lp}, \end{aligned}$$

where $G_{lp} \in R^{m \times n}$, $A_{lp} \in R^{p \times n}$, and and $b_{lp} \in R^p$.

C.1.2 Second-order Cone Programming (SOCP)

Second-order cone programming (SOCP) provides a formalism to describe optimization problems with second-order cone constraints , as the form below.

$$\begin{aligned} \min_{x} \quad & c_{socp}^T x \\ \text{s.t.} \quad & ||A_{i,socp} x + b_{i,socp}||_2 \leq c_{i,socp}^T + d_{i,socp} \quad i = 1, \ldots, m \\ & F_{socp} x = g_{socp}, \end{aligned}$$

doi:10.6342/NTU202404031

where $A_{i,socp} \in R^{n_i \times n}$, $b_{i,socp} \in R^{n_i}$, $c_{i,socp}, c_{socp} \in R^n$, $d_{i,socp} \in R$, $F_{socp} \in R^{p \times n}$ and $g_{socp} \in R^p$.

C.1.3 Semidefinite Programming (SDP)

In semidefinite programming (SDP), semidefinite constraints on matrix variables are applied, which has the form as

$$\begin{aligned} \min_{X \in S^n} \quad & \langle C_{sdp}, X \rangle \\ \text{s.t.} \quad & X \succeq 0 \\ & \langle A_{i,sdp}, X \rangle = b_{i,sdp} \quad i = 1, \ldots, p, \end{aligned}$$

where $\langle \bullet \rangle$ is the inner product, $C_{sdp}, A_{i,sdp} \in S^n$, and $b_{i,sdp} \in R$.

C.1.4 Conic Programming (CP)

Conic programming (CP) is optimize problem as the form

$$\begin{aligned} \min_{x \in D} \quad & c_{cp}^T x \\ \text{s.t.} \quad & D_{cp}(x) + d_{cp} \in K_{cp} \\ & A_{cp} x = b_{cp}, \end{aligned}$$

where $D_{cp}(x) : R^n \to R^n$ is a linear mapping of x, $K_{cp} \in S_+^n$ is a closed convex cone, $A_{cp} \in R^{p \times n}$, and $b_{cp} \in R^p$.

C.2 Other Research Fields of Mathematical Optimization

doi:10.6342/NTU202404031

There are many research fields of mathematical optimization [79: Wikipedia]. Integer programming aims to the linear optimization in which some or all variables are constrained to take on integer values. Fractional programming studies the problem of the ratio of two nonlinear functions. Nonlinear programming has the condition that the objective function or the constraints or both contain nonlinear parts. These research fields are also being studied by people recently.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Appendix D

Other Formation Tracking Control Algorithms for Multi-Agent Systems

In Chapter 2, we briefly introduce some literature survey about the formation tracking methods. In this section, we will introduce these multi-agent control strategies mentioned before except the consensus-based algorithm we implement in the system.

D.1 Leader-Follower

Leader-follower is an algorithm to control with leaders and followers. The followers need to track the leaders according to the motion of the leaders. This methodology isn't restricted to robotics; it finds parallels in nature, such as migratory birds flying in a leader-follower formation. While relatively straightforward to implement, it's susceptible to error propagation, particularly when leaders fail, leading to deviations in the followers' trajectories.

D.2 Potential Function

Potential function is a method by designing potential field in the functions according to the states of the target and plants which offers a flexible approach to guiding agents towards desired states. By designing potential functions for the system dynamics and objectives, virtual forces are generated to direct agents accordingly. Gradients of these potential functions plays a key role in determining the direction and magnitude of these forces. This method allows for the creation of control methods that are adaptable to various

doi:10.6342/NTU202404031

environmental conditions and system constraints, making it a powerful tool for designing autonomous systems capable of navigating complex environments and achieving desired objectives.

D.3 Virtual Structure

Virtual structure is a method that each agent has its own trajectory. It aims at controlling a group of robots to maintain a rigid formation, including defining the desired dynamics of the virtual structure, translating this desired motion of the virtual structure into individual motions for each robot, and deriving individual tracking controllers for each robot to ensure they follow their assigned trajectories and coordinate with others to maintain the overall formation stability to achieve the maintenance of the formation. It solves the error propagation but cannot avoid collisions because there is no interaction with each other. That is, an open-loop control is applied in the algorithm.

D.4 Behavior-Based

The behavior-based method is an approach to agent control that relies on defining specific behaviors for agents to exhibit based on their environment. For instance, in the context of a school of fish, individual fish follow simple rules governing their movement, resulting in collective behavior such as clustering. It is usually combined with the potential function method. One notable advantage of this method is its scalability, allowing for the seamless addition of more agents by incorporating additional behaviors. However, a limitation arises in the inability to guarantee a fixed pattern of behavior that ensures the convergence of the formation to a desired configuration. This inherent uncertainty underscores the need for careful design and monitoring of the behavioral rules to achieve

doi:10.6342/NTU202404031

desired outcomes in complex systems.

D.5 Intelligence

Intelligence method use artificial Intelligence method to control the agents. Different with the previous algorithms, the intelligence-based algorithm applies data-driven methods on the formation. As a result, it can implement more kinds of actions, which will make the diversity of the movements. However, it is hard to analyze with mathematical derivations, so it is not such suitable on the vehicles with highly rigorous design such as rockets nowadays.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Appendix E

Two-Step Error-Based Trajectories Planning of Quadrotors

In this section, we will introduce the strategies including how to construct the polynomial trajectory of the target and do trajectory planning according to the known trajectories of the target in x and y direction.

E.1 Sequential Quadratic Programming (SQP) for Solving Mathematical Optimization Problem

Sequential Quadratic Programming (SQP) is an iterative method for constrained nonlinear optimization which may be considered a quasi-Newton method. It is used on mathematical problems for which the objective function and the constraints are twice continuously differentiable, but not necessarily convex [80: Wikipedia].

We consider a nonlinear programming problem of the form

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g(x) \geq 0 \\ & h(x) = 0. \end{aligned}$$

The objective is to solve this kind of problem. In the traditional method, we first obtain

doi:10.6342/NTU202404031

the Lagrangian for this problem

$$\mathfrak{L}(x, \lambda, \sigma) = f(x) - \lambda g(x) - \sigma h(x),$$

where λ and σ are Lagrange multipliers. It has the characteristic that when the gradient of Lagrangian $\bigtriangledown\mathfrak{L}(x, \lambda, \sigma)$ approaches zero, we can find the solution. To solve the problem, the standard Newton's method can be applied to search for the solution by iterating the equations below.

$$\begin{aligned}\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \\ \sigma_{k+1} \end{bmatrix} &= \begin{bmatrix} x_k \\ \lambda_k \\ \sigma_k \end{bmatrix} - \begin{bmatrix} \bigtriangledown_{xx}{}^2\mathfrak{L} & \bigtriangledown g & \bigtriangledown h \\ \bigtriangledown g^T & 0 & 0 \\ \bigtriangledown h^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \bigtriangledown f + \lambda_k \bigtriangledown g + \sigma_k \bigtriangledown h \\ g \\ h \end{bmatrix} \\ &= \begin{bmatrix} x_k \\ \lambda_k \\ \sigma_k \end{bmatrix} - \bigtriangledown^2\mathfrak{L} \, \bigtriangledown\mathfrak{L},\end{aligned} \tag{E.1}$$

where $\bigtriangledown^2\mathfrak{L}$ is the Hessian matrix of $\mathfrak{L}$. However, because $\bigtriangledown^2\mathfrak{L}$ is generally singular, it is noninvertible. As a result, the Newton step $d_k = (\bigtriangledown^2\mathfrak{L})^{-1}\bigtriangledown\mathfrak{L}$ cannot be solved by the calculation of the matrices.

In order to solve the problem, the optimization problem can be modified as

$$\begin{aligned} \min_d \quad & f(x_k) + \bigtriangledown f(x_k)^T d + \frac{1}{2} d^T \bigtriangledown^2 f(x_k) d \\ \text{s.t.} \quad & g(x_k) + \bigtriangledown g(x_k)^T d \geq 0 \\ & h(x_k) + \bigtriangledown h(x_k)^T d = 0. \end{aligned}$$

It is because when we find the solution by using Newton's method, the gradients $\bigtriangledown f(x_k)$,

doi:10.6342/NTU202404031

$\nabla g(x_k)$ and $\nabla h(x_k)$ and $\nabla^2 f(x_k)$ will all approach zero. Thus, the optimization problem will be the same as the initial one.

The workflow is shown in Figure E.1. We first choose the initial value of $(x_0, \lambda_0, \sigma_0)$ to calculate the gradient $\nabla \mathfrak{L}$ and Hessian $\nabla^2 \mathfrak{L}$ of Lagrangian $\mathfrak{L}$. We then iterate $(x_k, \lambda_k, \sigma_k)$ by calculating the Newton step d_k until satisfying the convergence requirement. That is, we can set an error tolerance. If error is smaller than the tolerance, the iteration will stop.

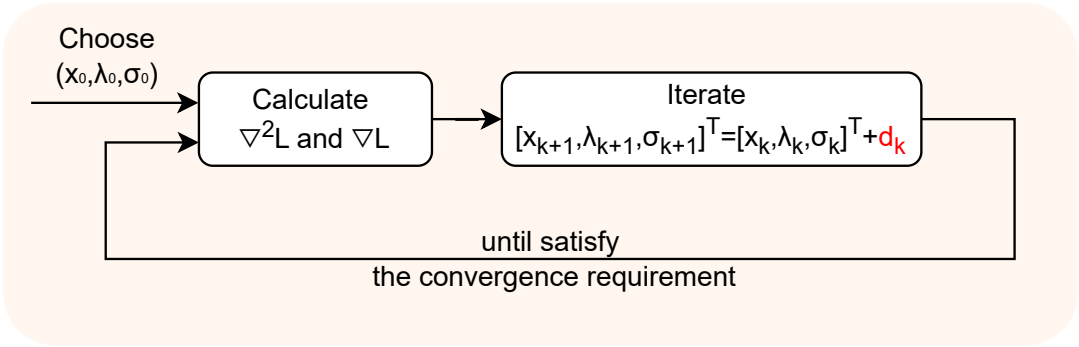


Figure E.1: The workflow for solving SQP by Newton's method.

We use the library "NLopt" to compute the optimization problem [81: Johnson 2007] Figure E.2. It is an open-source library for nonlinear optimization, which is suitable to solve optimization using C, C++, Fortran, Matlab or GNU Octave, Python, GNU Guile, Julia, GNU R, Lua, OCaml, Rust and Crystal. We only need to type the objective function, constraints, upper and lower bounds and tolerances and we can solve the optimization problem, which is convenience for us.

E.2 Cost Function Based on Error

In [21: Thomas *et al.* 2017], the authors use multiple cost functions to construct the algorithm. We will use this concept in the planning algorithm as well. The error of the

doi:10.6342/NTU202404031

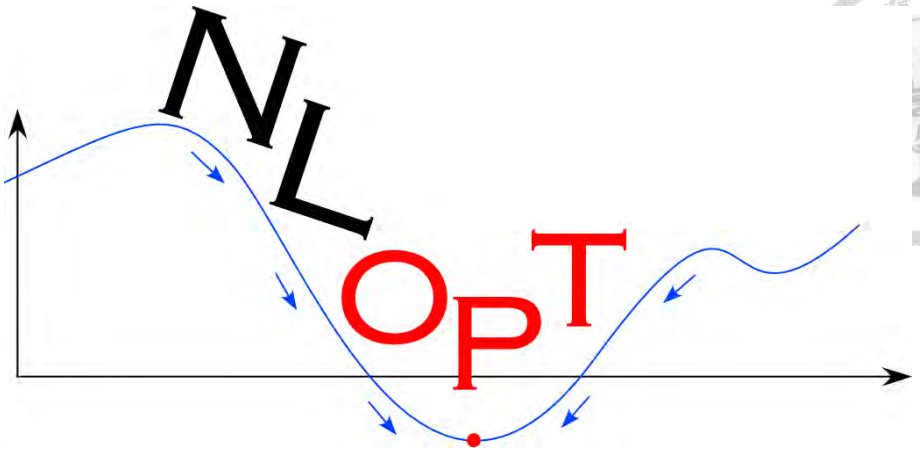


Figure E.2: Logo of NLopt.

trajectory can be defined as

$$e(t) = g(t) - p(t), \tag{E.2}$$

which means it is the difference between the trajectories of the target $(g(t))$ and quadrotor $(p(t))$. The cost functions can be defined as

$$\mathcal{J}_r = \int_{t_0}^{t_f} \left\| e^{(r)}(t) \right\|^2 dt, \quad r = 0, 1, 2, 3, \ldots, \tag{E.3}$$

which is the integration of the square of the Euclidean norm of the r^{th} derivative of the error.

E.3 Two-Step Error-Based Planning

The whole planning algorithm can be formulated in Algorithm 1. We first define the initial cost function and update the trajectories of the target, and then we can solve the SQP problem to update the planning trajectory of the drone. After optimization, we update the trajectory error, and select whether to update the cost function according to the error.

This algorithm use a switching method based on error. The reason is that when the direction of error and difference of error are the same or when the error is small the position

doi:10.6342/NTU202404031

Algorithm 1 The Planning Algorithm

$J \leftarrow \lambda_1 J_1 + \lambda_3 J_3$
for Each Horizon **do**
    update($g(t)$)
    repeat
        $p(t) \leftarrow$ iterateSQP(J, $g(t)$, $p(t)$)
    until Out of Time
    $e(t) \leftarrow g(t) - p(t)$
    if ($e \cdot \dot{e} \geq 0$ or $\|\dot{e}\| \leq$ thresh) **then**
        $J \leftarrow \lambda_0 J_0 + \lambda_1 J_1 + \lambda_3 J_3$
    end if
end for

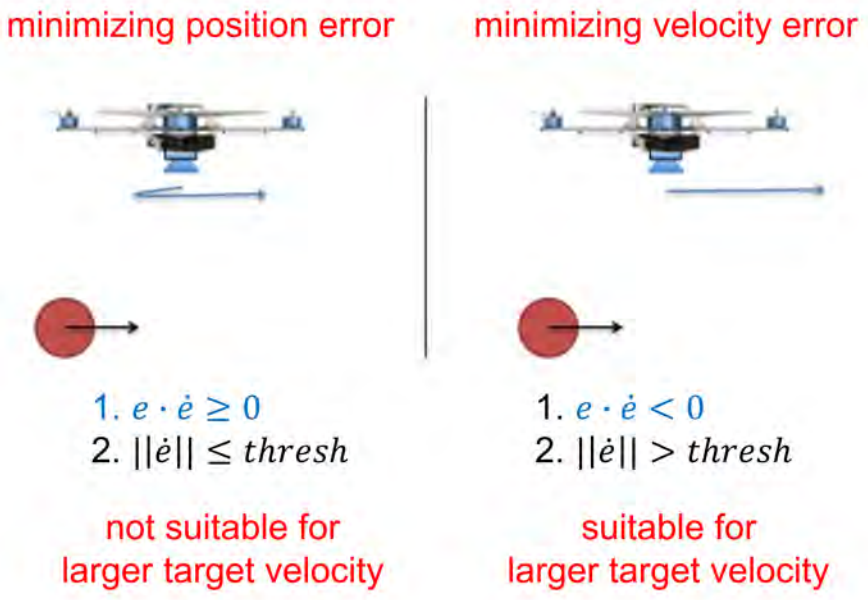


Figure E.3: Different situations according to the error condition.

is easier to track. However, when the the direction of error and difference of error are not the same or when the error is large, the quadrotor will go through a process to fly back and forth severely, which is not an ideal sutuation to track the position error, as shown in Figure E.3. As a result, this algorithm provides an efficient tracking method to track the ground target with onboard downward camera. However, due to computation limitation of the hardware, we won't use this method for trajectory planning of the quadrotors in the implementation of this thesis.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

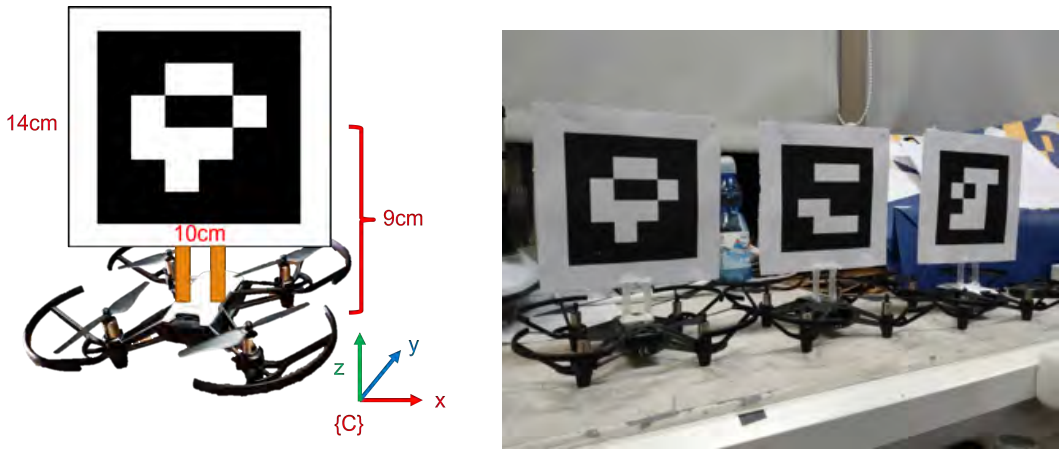
Appendix F

Motion Capture System from the Third Perspective for Validation

In order to validate the strategies we applied, we need to obtain the information of the objects from a third perspective. As a result, a motion capture system is what we can do. In this section, we will introduce how to implement a motion capture system from the third perspective.

F.1 Hardware Design on the Quadrotors, an ArUco Marker Based State Estimation

We design a structure using ArUco markers, as shown in Figure F.1. An 4x4 ArUco



(a) The schematic design with an ArUco marker. **(b) The real appearance.**

Figure F.1: The structure design of ArUco markers and Tellos.

marker is installed on a Tello with two pieces of straws with marker ID of 98 on tello_1, marker ID of 99 on tello_2, and marker ID of 100 on tello_3. The side lengths of ArUco markers are 10 centimeters and the side lengths of the styrene papers on the back are 14

doi:10.6342/NTU202404031

centimeters. The heights between the geometric centers of Tellos and ArUco markers are 9 centimeters. We use this structure as the motion capture target.

In order to validate whether it will affect much about the flight performance, we use an electronic scale to measure the weights before and after installing the structure individually. As shown in Table F.1, each initial weight including the drone itself and battery is around 86 grams, and the whole system is around 94 grams. Each designed structure has the weight less than 10 grams.

Table F.1: The weight of each drone (before and after).

Drone	Initial Weight (Drone+Battery)	Final Weight (Drone+Battery+Marker)
tello_1	86.3g	94.2g
tello_2	86.2g	94.2g
tello_3	85.4g	93.8g

F.2 Feature Extraction Using RGB Image and Markers

We use an RGB camera as our sensor to detect the ArUco markers. As shown in Figure F.2 [82: OpenCV], we use an RGB camera to observe the motion of the ArUco marker(s) on the Tello(s). According to the function of OpenCV package, we can extract the corners of the ArUco marker(s).

F.3 Perspective-n-Point (PnP) Algorithm for Motion Capturing from 2D RGB Image

We use Perspective-n-Point (PnP) method to obtain the relative 3D position using several feature points. According to the coordinate transformation from Section 3.1, it

doi:10.6342/NTU202404031

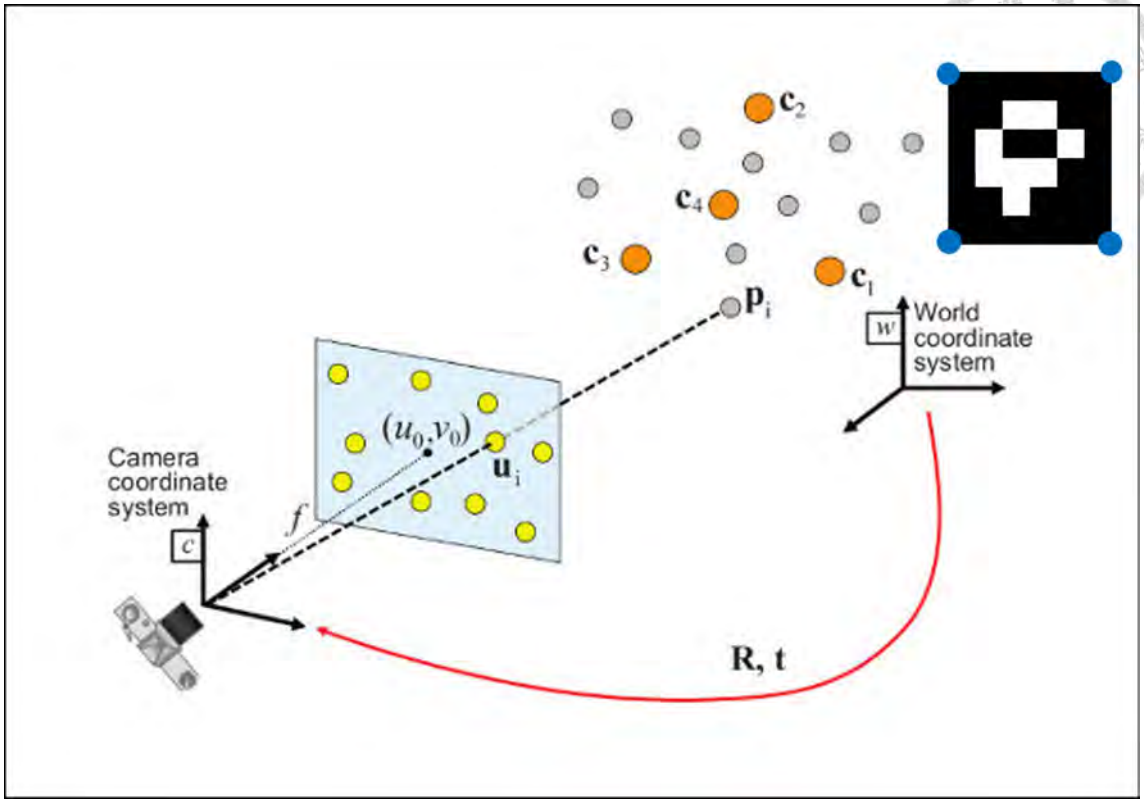


Figure F.2: The schematic figure of PnP.

has the form as

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}, \quad \text{(F.1)}$$

where c_x and c_y are the principle axes in the camera image in pixel. Now we know $f_x = 380.0$, $f_y = 381.7$, $c_x = 323.4$ and $c_y = 249.5$ in pixel, so we can obtain the camera intrinsic matrix. Furthermore, we can extract four points of features in an ArUco marker by using OpenCV package such that we can obtain the positions in the RGB image. Finally we define these four points in world frame. Since we know that the side length of the marker is 0.1 meter as shown in Figure F.2, we can use the `"cv.SOLVEPNP_IPPE_SQUARE"` algorithm with the defined points below.

doi:10.6342/NTU202404031

- $Point\ 0 : \left[-\frac{\text{side length}}{2}, \frac{\text{side length}}{2}, 0\right]$
- $Point\ 1 : \left[\frac{\text{side length}}{2}, \frac{\text{side length}}{2}, 0\right]$
- $Point\ 2 : \left[\frac{\text{side length}}{2}, -\frac{\text{side length}}{2}, 0\right]$
- $Point\ 3 : \left[-\frac{\text{side length}}{2}, -\frac{\text{side length}}{2}, 0\right]$

This algorithm assumes that the object we want to detect is a square with the number of feature points ≥ 4. We can use these points to obtain the PnP solution by calculating the matrix equations. Thus, we can obtain the rotation and translation.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{F.2}$$

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}. \tag{F.3}$$

After obtaining the rotation and translation, we can solve the motion capture problem using the 2D RGB image.

doi:10.6342/NTU202404031

Appendix G

Error Accumulation Validation

Figure 2.2 shows the effect of error accumulation of onboard IMU. In this chapter, we will validate the problem to explain the necessity of KF-based state estimation.

G.1 Experiment Setup

We use an RGB camera to observe a Tello with an ArUco marker at MD 5F, as shown in Figure G.1. We use PnP method to calculate the relative 3D position in world frame

(a) The overall setup. **(b) Image snapshot of the RGB camera.**

Figure G.1: Scenario to validate error accumulation of IMU.

using 2D RGB image. The process include

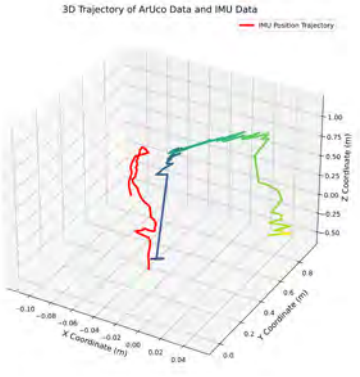
1. Tello and camera being launched at the same time
2. Tello motor on to wait for the camera turning on (10 sec)
3. Takeoff and Hovering (5 sec)
4. Moving with 0.3 m/sec open-loop command in Y_W direction (5 sec)
5. Landing.

We conducted 3 experiments in the same scenario.

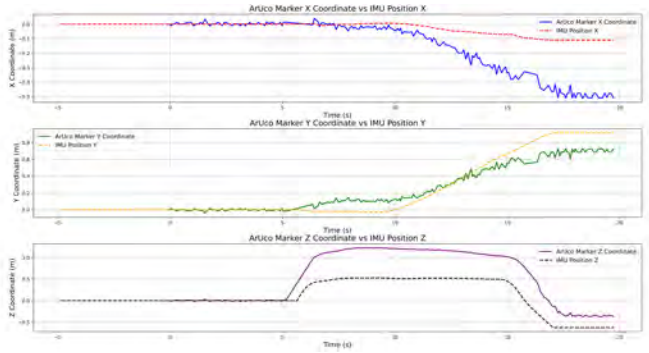
doi:10.6342/NTU202404031

G.2 3D Motion and Position within Time

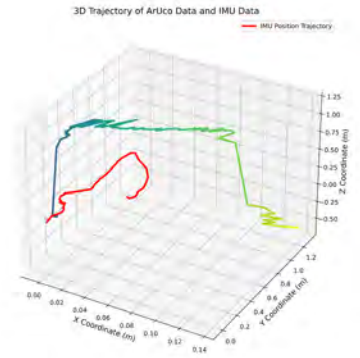
The comparison of 3D trajectory and measured by IMU and RGB camera is shown in Figure G.2.



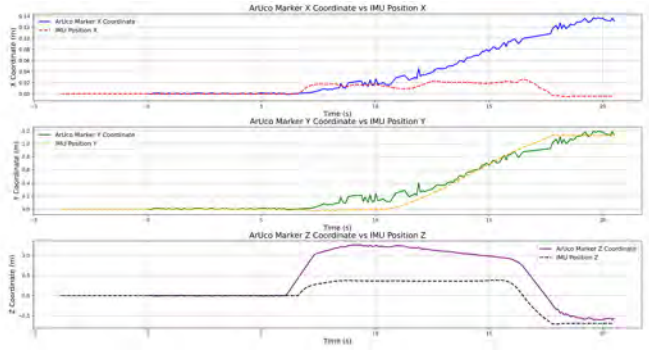
(a) 3D trajectories in Exp. 1.



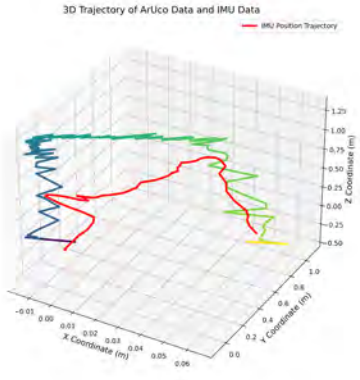
(b) Position v.s. time in Exp. 1.



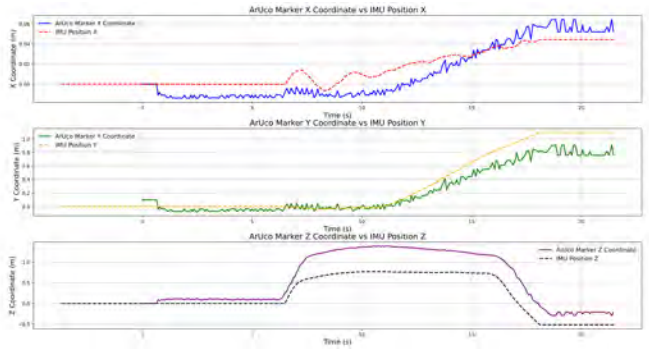
(c) 3D trajectories in Exp. 2.



(d) Position v.s. time in Exp. 2.



(e) 3D trajectories in Exp. 3.



(f) Position v.s. time in Exp. 3.

Figure G.2: Comparison of 3D trajectory and position within time.

In Figure G.2(a), Figure G.2(c) and Figure G.2(e), the monochrome curves are the measurement of IMU and the curves with color gradient from blue to yellow (from dark to

doi:10.6342/NTU202404031

light) are the measurement of RGB camera from the third perspective. In Figure G.2(b), Figure G.2(d) and Figure G.2(f), the data from X_W, Y_W and Z_W is plotted in order. The dashed curves are the measurement from IMU data and the solid curves are the measurement by RGB camera using PnP, which is seen as the ground truth.

As we can see, when the scale of distance gets smaller, the error percentage gets higher, as the X_W data in the figure. Nonetheless, as it gets larger, we can easily observe the tendency of the data. We can observe that as time goes, the error of IMU measurement gets higher.

Additionally, in the data of Z_W direction, the height before and after the flight is not the same. It is because the IMU will turn on after taking off for a few time, which is one of the main problem of the hardware. However, due the the difference of the IMU velocities measurement between different agents. The calculation of the height often fails to be accurate, so we won't use this algorithm for the height control. Instead, we know that the height while taking off is 1.3m for the inner loop controller, so we can use this characteristic to design the estimation framework.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Appendix H

Height and Yaw Control with PID Controllers

In this section, we will introduce the control strategies to control the height and yaw.

H.1 Introduction of PID Controllers

Proportional-Integral-Derivative (PID) control is a common technique in control systems for achieving desired performance in various systems. It is particularly common in industries like manufacturing, robotics, automotive, aerospace, etc. In PID control strategy, we can divide the control input into three parts, including P, I and D, as shown in the equation [83: Wikipedia].

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \tag{H.1}$$

where the proportional, integral and derivative gain $K_p, K_i, K_d \geq 0$.

In Figure H.1, according to the sensor information or observation, we can obtain the current state $y(t)$. After obtain the state, we can compare it with the reference $r(t)$ such as the desired position and obtain the error $e(t)$. Consequently, we can design a controller to formulate the control input $u(t)$ to control the plant. However, because our system is a discrete-time system, we can modify the control input as

$$u(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i)T + K_d \frac{e(k) - e(k-1)}{T}, \tag{H.2}$$

doi:10.6342/NTU202404031

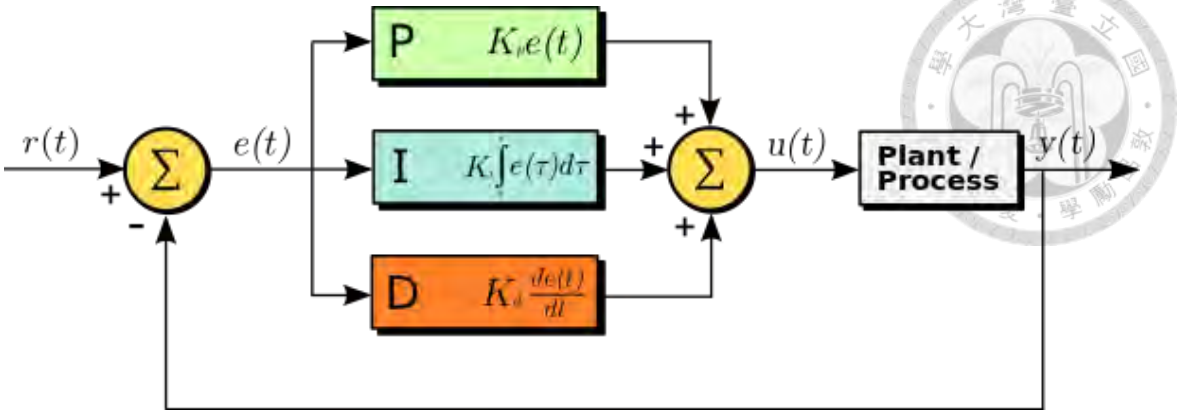


Figure H.1: The block diagram of PID controller.

where T is the sampling time.

H.2 Lowpass Filter for Highly-Jittering Signals

Lowpass filter is a filter to let the signal with low frequency pass and reject the high-frequency term. It is widely used in signal processing such as image, audio or even integrated circuit (IC) design. Figure H.2 shows the frequency response of a first-order lowpass filter, where x axis is the angular frequency (rad/s) and y axis is the gain (dB). We can see as frequency get higher, the gain pass through the filter will decrease. As a result, we can design it to reject the high-frequency noise.

H.2.1 Height Control with IMU Information

As for the height control, we use the IMU data directly to obtain the position in z direction of the quadrotor. However, because the IMU velocity data is jittering, as shown in Figure H.4 The integrated position data in Z_W direction is not such accurate. We induce the first-order lowpass filter to reduce the problem, which can be defined as

$$height_f(k) = \alpha_h \times height(k) + (1.0 - \alpha_h) \times height_f(k-1), \tag{H.3}$$

where α_h is a constant we need to design, $height_f(k)$ is the filtered height and $height(k)$

doi:10.6342/NTU202404031

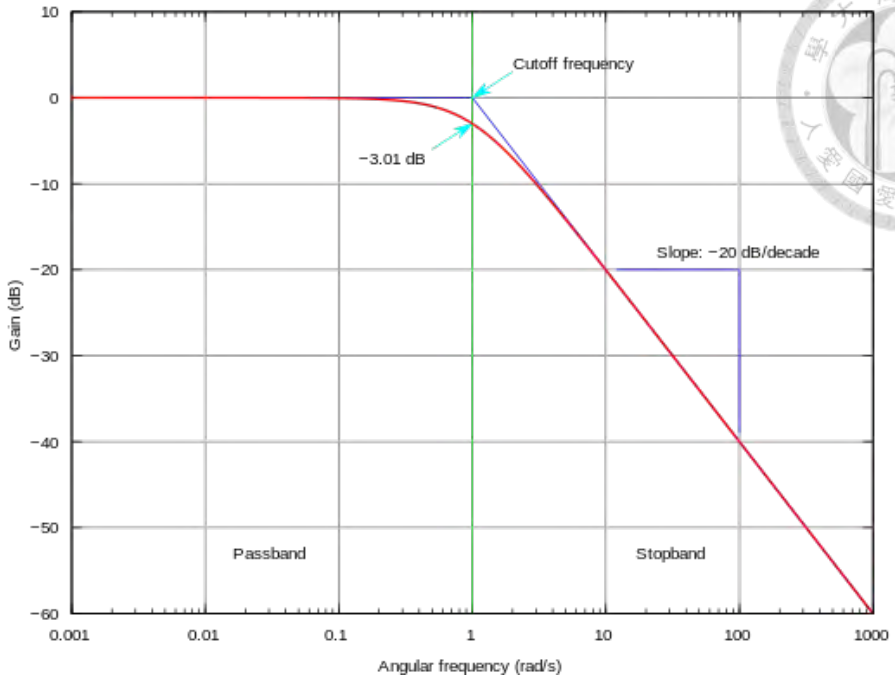


Figure H.2: Frequency response of a first-order lowpass filter.

is the raw data measured by IMU at time k. We use the filtered data as the current state $y(k)$ to form a PID controller on the quadrotor.

The other problem we need to solve is that the height while taking off is a constant and the IMU will function at the time during taking off, where we can see the initial height before taking off and the final height after landing is not the same, as shown in A.2. However, we can observe the takeoff height is about 1.3 meter. This information can be a vital essence we can go further. We set the initial height at 1.3 meter and the desired height at 1.5 meter and we can apply the PID controller on the height control.

H.2.2 Yaw Control with IMU Information

Similarly, yaw is also the data we want to improve. As shown in Figure H.5, it jitters while fiying. The first-order lowpass filter on the information of yaw can be defined as

$$yaw_f(k) = \alpha_{yaw} \times yaw(k) + (1.0 - \alpha_{yaw}) \times yaw_f(k-1), \tag{H.4}$$

doi:10.6342/NTU202404031

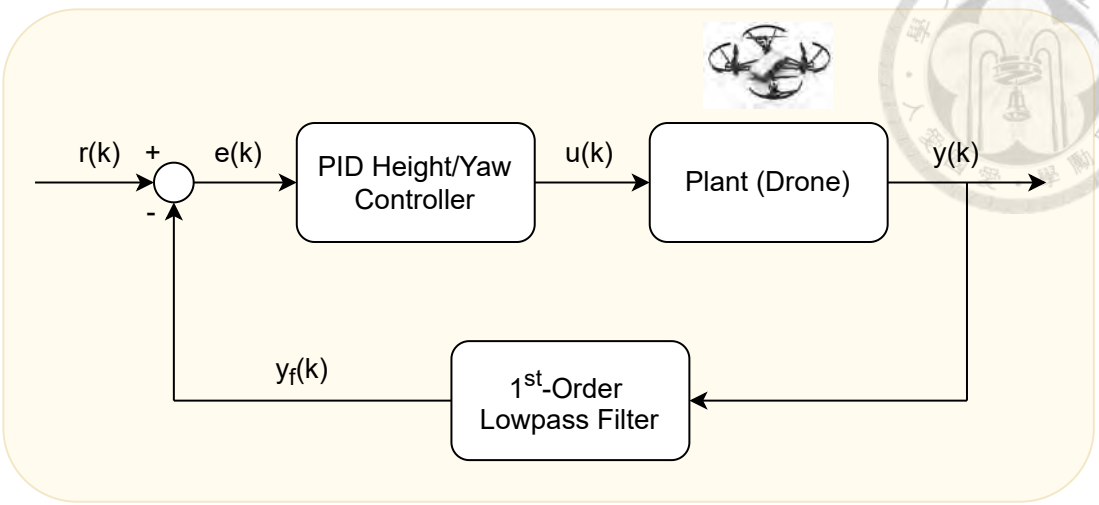


Figure H.3: The workflow of height and yaw control using lowpass filter and PID controller.

we also design α_{yaw} to make the data smoother. Now we want the yaw angle keep at 0 degree with initial yaw at 0 degree as well. We can also control it according to the algorithm of PID controller.

Consequently, we design a lowpass filter to make the IMU data smoother, including height and yaw. Furthermore, as shown in Figure H.3 we use the filtered data as the current state $y_f(k)$ to compare with the desired state $r(k)$ and design a PID controller to give the linear/angular velocity command $u(k)$ on the quadrotors.

H.3 Experiment Results for Height and Yaw Control with IMU Information

In this section, we will demonstrate the capability to control the height and yaw with IMU data. These states has the characteristic that the traveling distance is smaller. Thus, relatively mild error accumulation occurs. In 4.4, we design a PID controller with a first-order lowpass filter. As a result, this section demonstrate the capability of the control

doi:10.6342/NTU202404031

method for yaw and vertical direction.

H.3.1 Experiment Setup

We use an RGB camera to observe a Tello with an ArUco marker at MD 5F. Furthermore, we use PnP method to calculate the relative 3D position in world frame using 2D RGB image as well. The process include

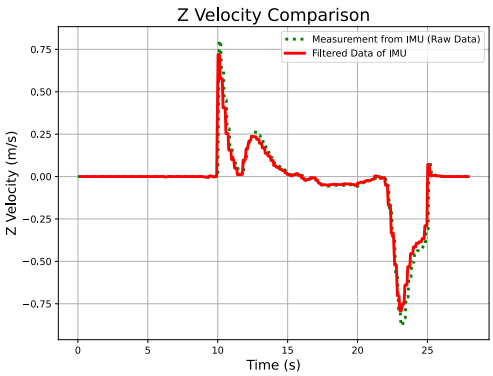
1. Tello and camera being launched at the same time
2. Tello motor on to wait for the camera turning on (10 sec)
3. Takeoff and controlling the height and yaw (10 sec)
4. Landing.

We conducted 3 experiments in the same scenario.

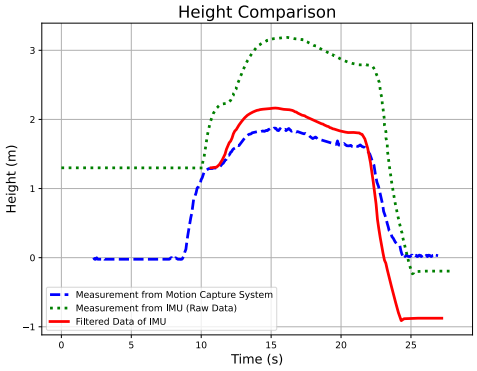
H.3.2 Comparison of Raw/Filtered Data of Height

The comparison of velocity and height in Z_w direction is shown in Figure H.4 In Figure H.4(a), Figure H.4(c) and Figure H.4(e), the dotted curves are the raw velocity data from IMU measurement. We can observe that according to the filtering process, the velocity data will be smoothed as the solid curves. Figure H.4(b), Figure H.4(d) and Figure H.4(f) shows the result of height control framework, where the dotted curves are the raw data of height from IMU velocity measurement, the solid curves are the filtered height data from integration of filtered velocity data, and the dashed curves are the height measurement from the motion capture system using RGB image from the third perspective, which is also the ground truth. We can observe that after filtering, the height data will be

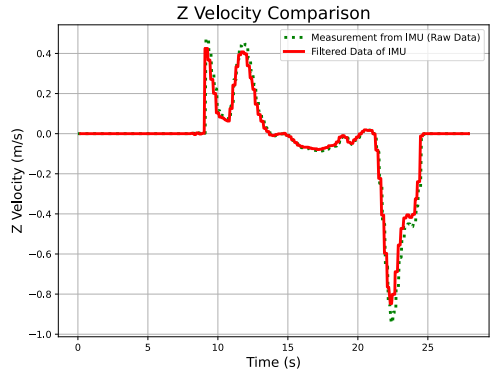
doi:10.6342/NTU202404031



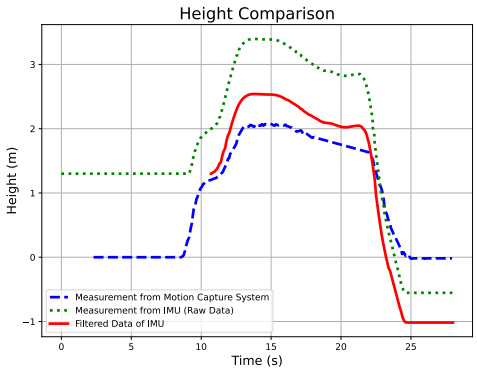
(a) Z velocity v.s. time (test 1).



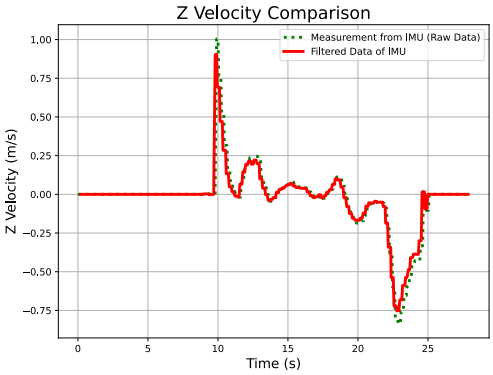
(b) Height v.s. time (test 1).



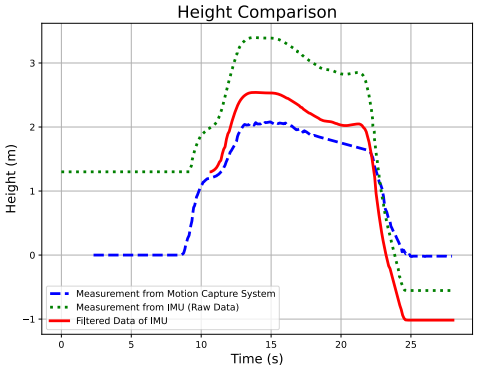
(c) Z velocity v.s. time (test 2).



(d) Height v.s. time (test 2).



(e) Z velocity v.s. time (test 3).



(f) Height v.s. time (test 3).

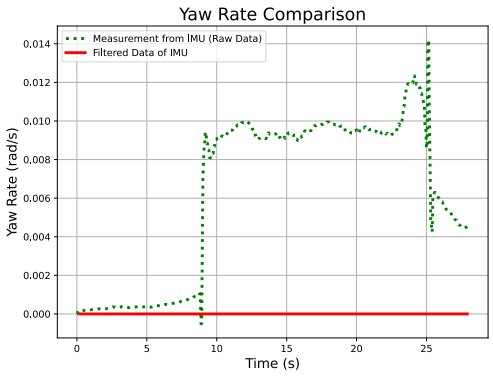
Figure H.4: Comparison of Z velocity and height within time.

doi:10.6342/NTU202404031

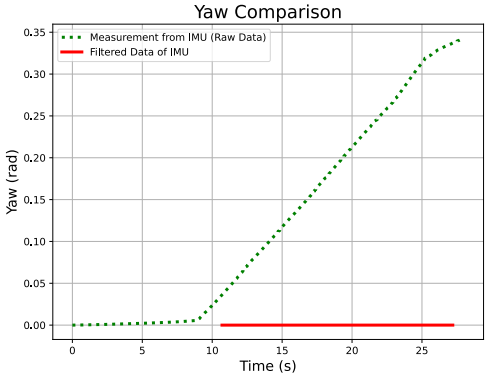
more accurate than the original one, which is our objective.

H.3.3 Comparison of Raw/Filtered Data of Yaw

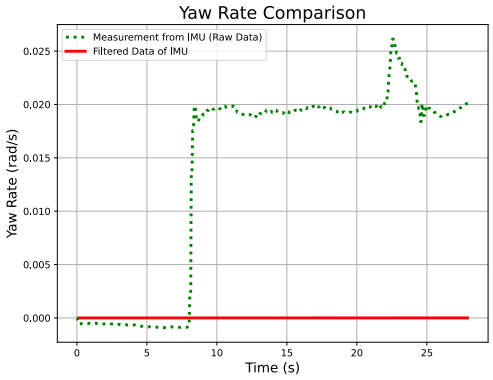
The comparison of yaw rate and yaw is shown in Figure H.5 In Figure H.5(a), Figure



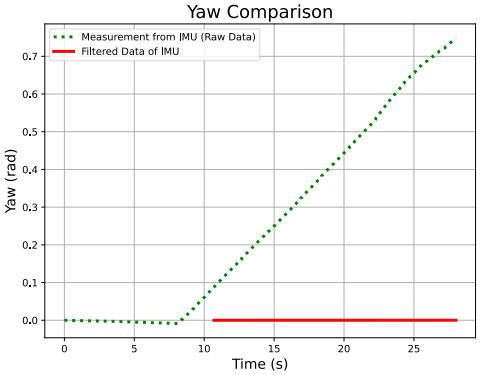
(a) Yaw rate v.s. time (test 1).



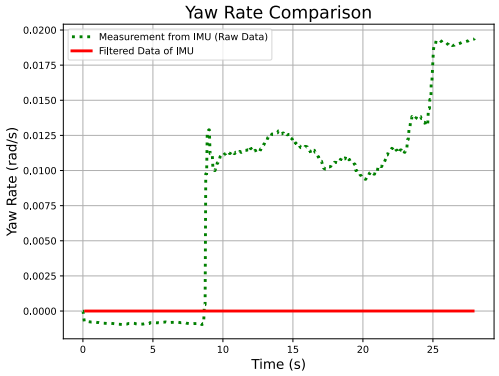
(b) Yaw v.s. time (test 1).



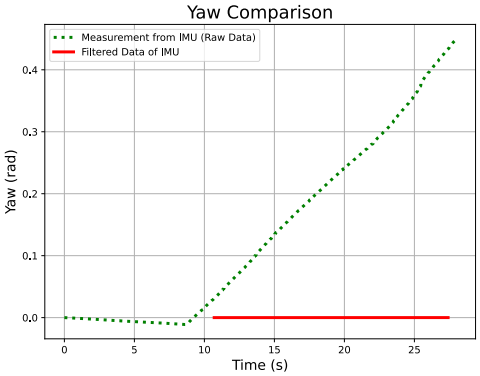
(c) Yaw rate v.s. time (test 2).



(d) Yaw v.s. time (test 2).



(e) Yaw rate v.s. time (test 3).



(f) Yaw v.s. time (test 3).

Figure H.5: Comparison of yaw rate and yaw within time.

H.5(c) and Figure H.5(e), the dotted curves are the raw yaw rate data from IMU measurement. We can observe that according to the filtering process, the velocity data will

doi:10.6342/NTU202404031

approach zero as the solid curves. Figure H.5(b), Figure H.5(d) and Figure H.5(f) shows the result of yaw control framework, where the dotted curves are the raw data of yaw from yaw rate measurement and the solid curves are the filtered yaw data from integration of filtered yaw rate data, We can observe that after filtering, the yaw rate data will remain near zero, which is our objective. Otherwise, the slight yaw rate error will accumulate, which will result in the yaw offset after some certain time.

doi:10.6342/NTU202404031

Appendix I

KF-Based State Estimation

In this chapter, we will conduct some experiments about the capability of KF-based state estimation according to the landmarks.

I.1 Simulation: Vision Unavailable Case

First we do some simulation in Matlab. Recalling from the workflow of Kalman filter in Figure 3.4, the state space model to estimate the state can be defined as

$$\begin{cases} x(k+1) = Ax(k) \\ y(k) = \begin{bmatrix} \dot{X_W} \\ \dot{Y_W} \end{bmatrix} = Hx(k), \end{cases}$$

doi:10.6342/NTU202404031

where

$$x = \begin{bmatrix} X_W \\ Y_W \\ \dot{X_W} \\ \dot{Y_W} \end{bmatrix}, \tag{I.1}$$

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{I.2}$$

$$H = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{I.3}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{I.4}$$

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix}, \tag{I.5}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{I.6}$$

doi:10.6342/NTU202404031

The notation $dt = 1$ is the calculation step and $(q_1, q_2, q_3, q_4) = (1, 10, 1, 0.1)$. The formulation of H is defined above because when vision unavailable, we can only obtain the data of IMU as the measurement. That is, we can only obtain the velocity measured by IMU.

For the actual model, the state space can represented as

$$\begin{cases} x(k+1) = Ax(k) + w \\ y(k) = \begin{bmatrix} \dot{X_W} \\ \dot{Y_W} \end{bmatrix} = Hx(k) + v, \end{cases}$$

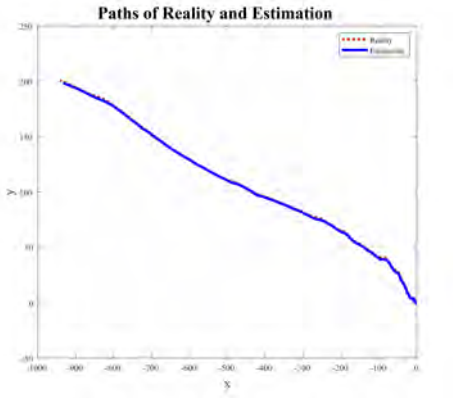
where w is the state noise and v is the sensor noise. Both of them are random Gaussian noises whose average are zero.

The simulation result of KF estimation is shown in Figure I.1. The dashed lines are the real signals from the actual model and the solid lines are the values by KF estimation. We can see in Figure I.1(a), Figure I.1(c), Figure I.1(e), Figure I.1(g) are the paths of the cases with w and v, without w, without v and without w and v. Furthermore, in Figure I.1(b), Figure I.1(f), Figure I.1(f), Figure I.1(h) the first and second columns are the term in X_W and Y_W directions, and the first and second rows are the term of velocity and position estimation within time respectively.

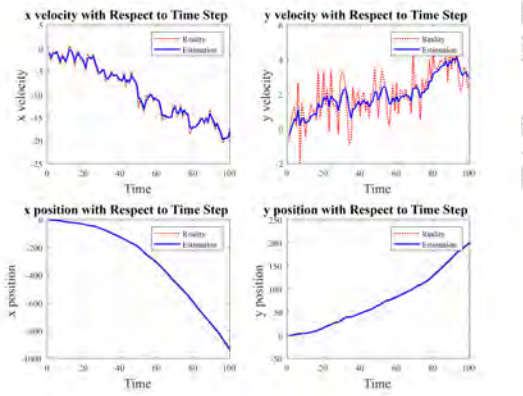
The motions are all from the random noises w and v. If there is no motion as the case in Figure I.1(g) and Figure I.1(h), the data will all be zero. We can observe that the KF algorithm can estimate the state in the cases with and without noises.

I.2 Experiment Results

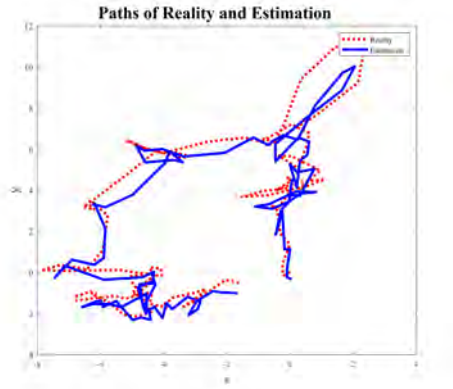
doi:10.6342/NTU202404031



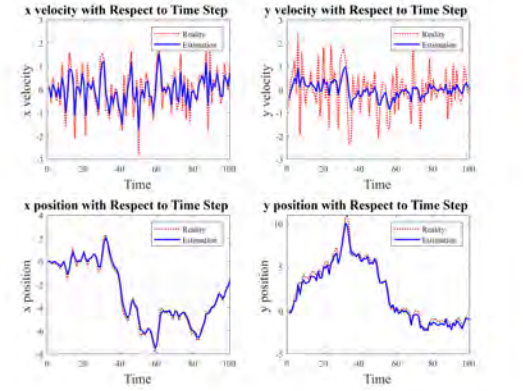
(a) Path of normal case with w, v.



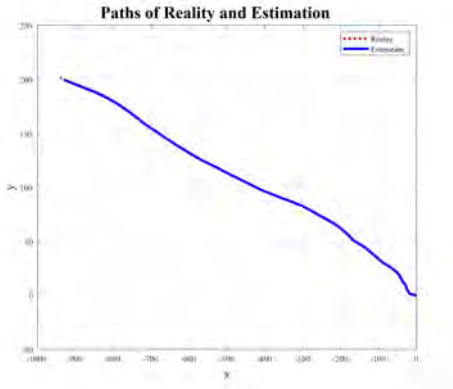
(b) Velocity and position with time with w, v.



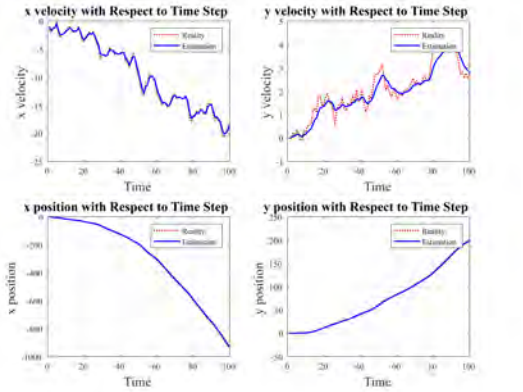
(c) Path of normal case without w.



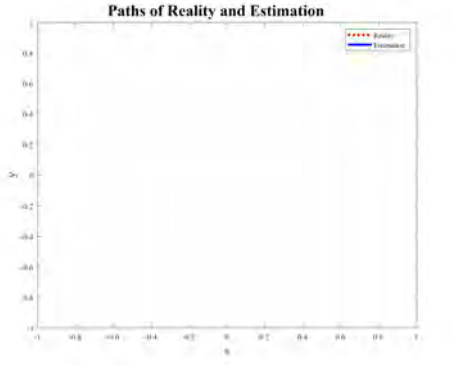
(d) Velocity and position with time without w.



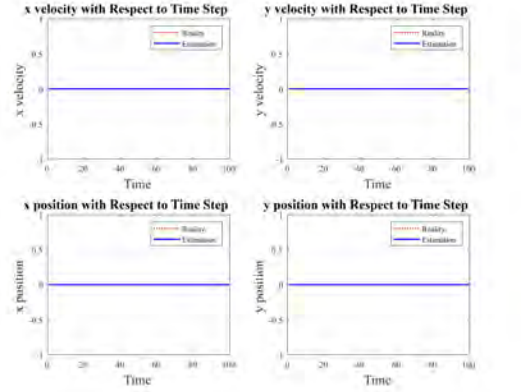
(e) Path of normal case without v.



(f) Velocity and position with time without v.



(g) Path of normal case without w, v.



(h) Velocity and position with time without w, v.

Figure I.1: Simulation results of KF-based state estimation.

doi:10.6342/NTU202404031

In this section, a KF-based localization framework is used in the valization by experiments. A drone will go through several points with open-loop command, where the landmarks will be in the FOV of the drone. In this case, we can use the landmarks for localization. Otherwise, we use the IMU velocity data for localization. For 1D case experiment, a drone will fly along Y_W direction and the landmark (ID: 5, 6) at $(0, 0, 0)$ and $(0, 1, 0.353)$ (m) in frame $\{W\}$ will be in the FOV after certain time of flight and out of FOV finally. For 2D case experiment, a drone will fly along a square clockwise each landmark (ID: 5, 6, 7, 8) is at $(0, 0, 0)$, $(0, 2, 0.353)$, $(2, 2, 0.355)$ and $(2, 0, 0.415)$ (m) respectively. We compare the localization result with the result from RGB camera-based motion capture system from the third perspective.

I.2.1 Experiment: 1D Case, Linear Motion

In this experiment, as shown in Figure I.2, the process will go through 4 steps, including

1. Tello motor on (5 sec)

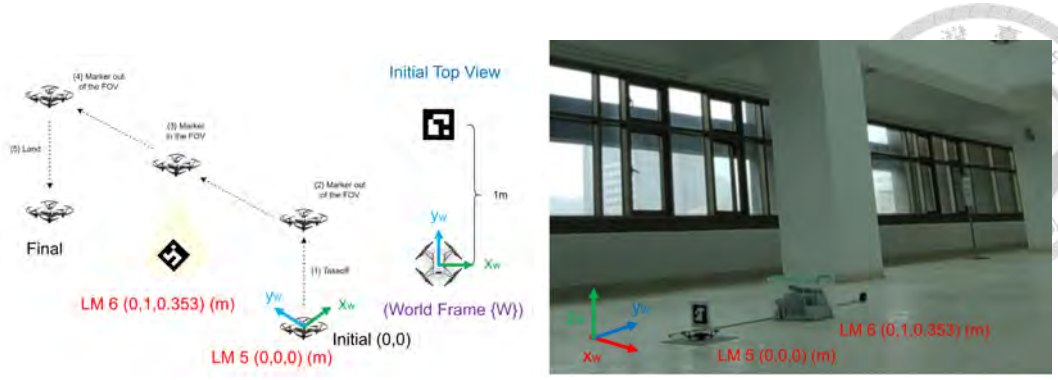
2. Takeoff and hovering (5 sec)

3. 0.5 m/s command along Y_W direction (5 sec)

4. Hovering (5 sec) and landing.

We conduct the experiment three times for this case and label them by case 1, 2 and 3. The result of the position of the drone by direct integration of IMU velocity, KF-based onboard localization and RGB camera-based motion capture system for 1D motion case

doi:10.6342/NTU202404031



(a) Schematic scenario of KF experiemnt, 1D motion case.

(b) Real scenario of KF experiemnt, 1D motion case.

Figure I.2: Scenario of KF experiemnt, 1D motion case.

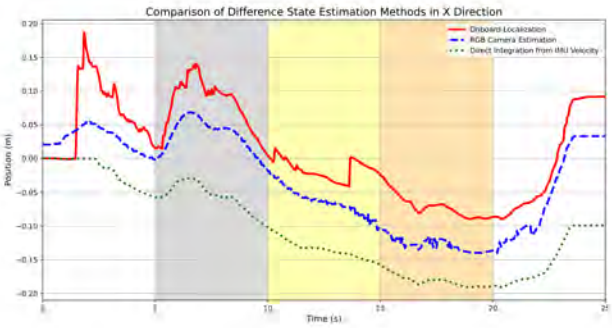
is shown in Figure I.3, Figure I.4 and Figure I.5, where the error is defined as

$$error(\%) = \frac{position(onboard\ or\ IMU) - position(RGB)}{height} \times 100\%. \tag{I.7}$$

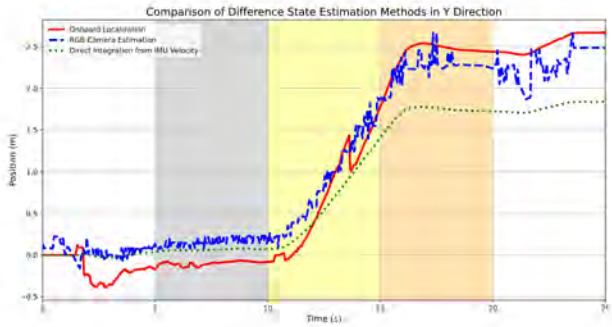
In the first two subfigures in each figure for each experiment, the (red) solid curves are the position by onboard KF-based localization, the (green) dotted curves are the position by directly integration from IMU velocity , and (blue) dashed curves are the position by RGB camera estimation, which is also seen as the ground truth. Furthermore, in the last two subfigures in each figure, the (red) solid curves are the position error by onboard KF-based localization and the (green) dotted curves are the position error by directly integration from IMU velocity Same definitions are used in the next section of 2D motion cases.

Simultaneously, the error comparison of all 1D cases is shown in Figure I.6. We examine the data from 9s to 16s because drone will see the landmark with high probability during this time interval and use the box plot to compare these experiments. We can observe that the position information from IMU velocity integration has less standard deviation for each case but larger range between different cases. In summary, though KF-based localization method has larger standard deviation in the same case, we can guarantee that the position data is in the certain range.

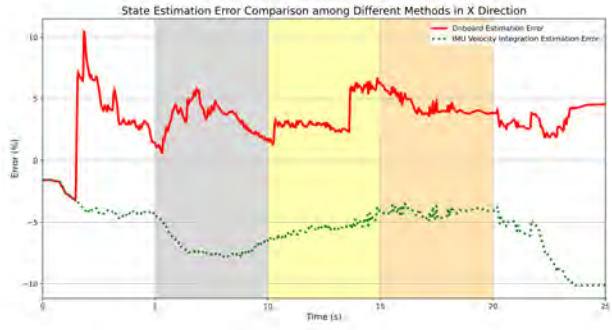
doi:10.6342/NTU202404031



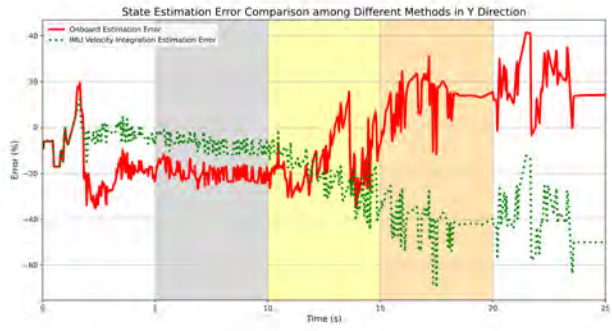
(a) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in X_W direction.



(b) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in Y_W direction.



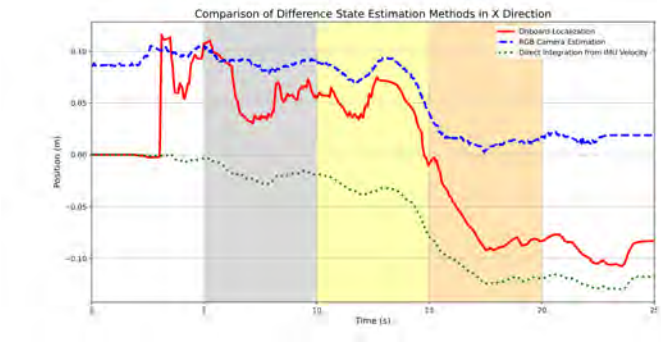
(c) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in X_W direction.



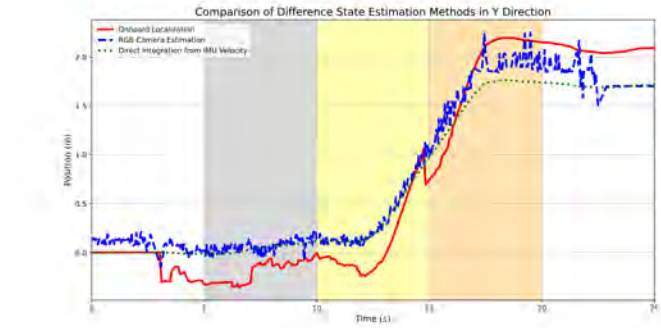
(d) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in Y_W direction.

Figure I.3: The comparison of drone positions by onboard/direct integration of IMU velocity/RGB camera estimation within time of case 1, 1D motion.

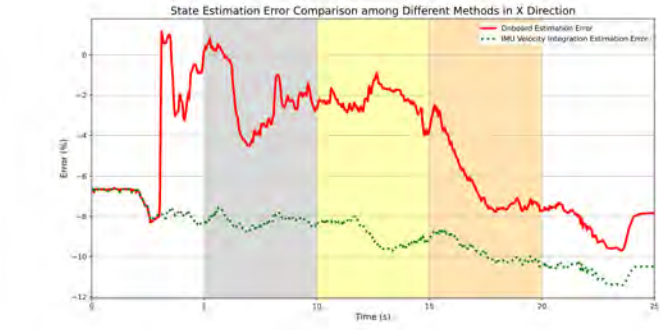
doi:10.6342/NTU202404031



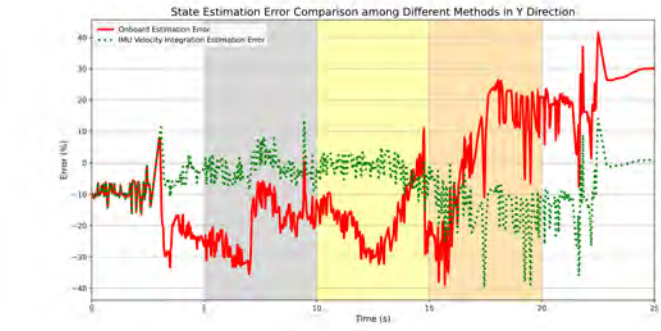
(a) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in X_W direction.



(b) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in Y_W direction.



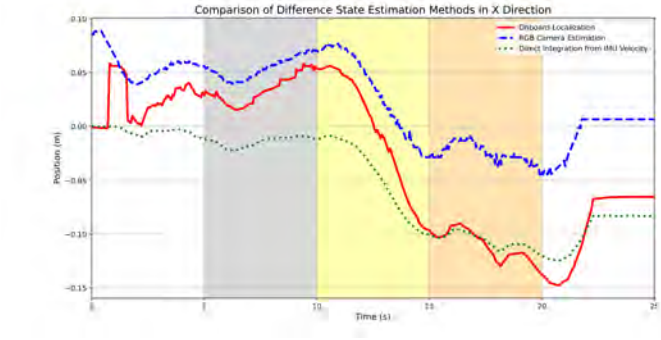
(c) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in X_W direction.



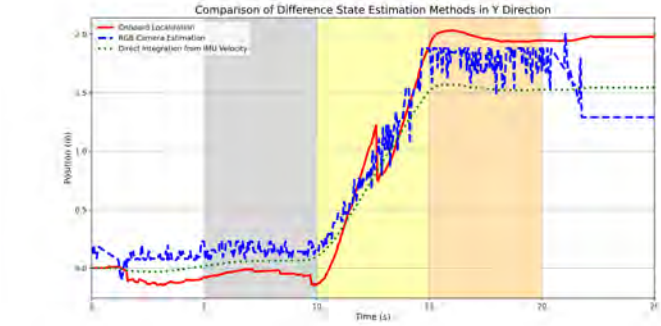
(d) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in Y_W direction.

Figure I.4: The comparison of drone positions by onboard/direct integration of IMU velocity/RGB camera estimation within time of case 2, 1D motion.

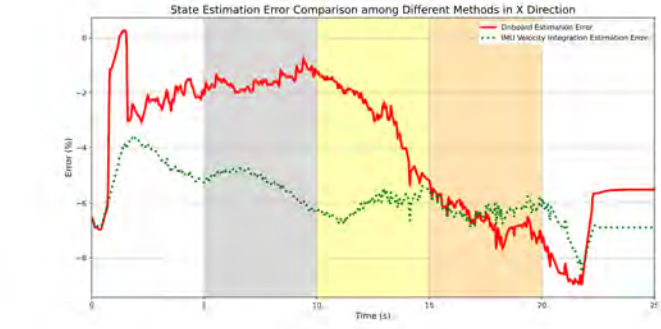
doi:10.6342/NTU202404031



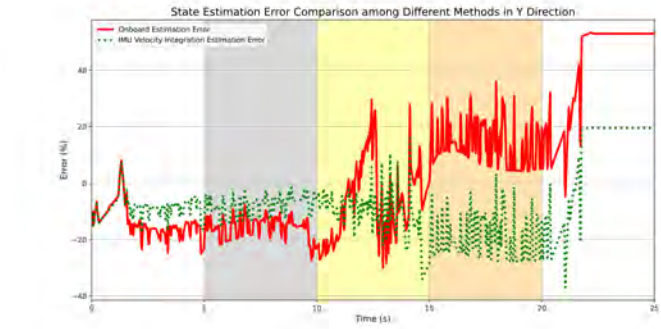
(a) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in X_W direction.



(b) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in Y_W direction.



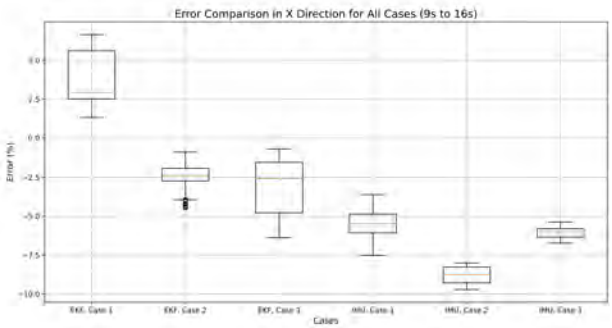
(c) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in X_W direction.



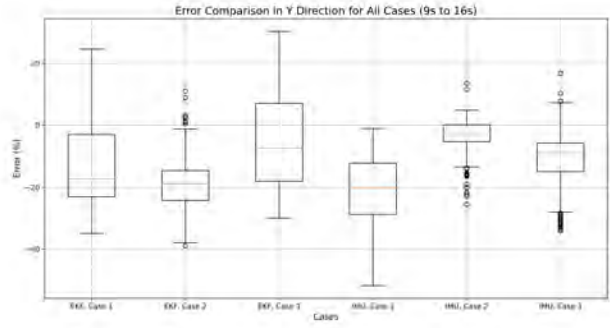
(d) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in Y_W direction.

Figure I.5: The comparison of drone positions by onboard/direct integration of IMU velocity/RGB camera estimation within time of case 3, 1D motion.

doi:10.6342/NTU202404031



(a) Drone position error comparison by onboard/direct integration of IMU velocity in X_W direction for all 1D cases.



(b) Drone position error comparison by onboard/direct integration of IMU velocity in Y_W direction for all 1D cases.

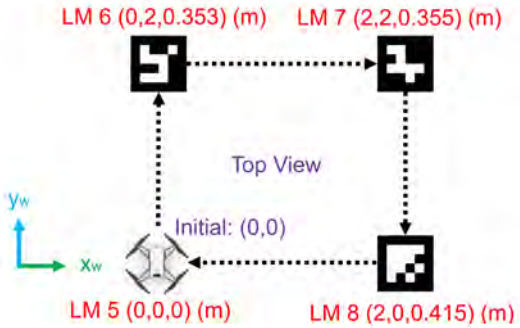
Figure I.6: The comparison of drone positions by onboard/direct integration of IMU velocity within time for all 1D cases.

doi:10.6342/NTU202404031

I.2.2 Experiment: 2D Case, Square Motion

In this experiment, as shown in Figure I.7 the process will go through 4 steps, including

1. Tello motor on (5 sec)
2. Takeoff and hovering (5 sec)
3. 0.5 m/s command along Y_W, X_W, $-Y_W$, $-X_W$ directions (5 sec each)
4. Hovering (5 sec) and landing.



(a) Schematic scenario of KF experiemnt, 2D motion case.

(b) Real scenario of KF experiemnt, 2D motion case.

Figure I.7: Scenario of KF experiemnt, 2D motion case.

The result of the position of the drone by direct integration of IMU velocity, KF-based onboard localization and RGB camera-based motion capture system for 2D motion case is shown in Figure I.8, Figure I.9 and Figure I.10. Because the command is open-loop, the motion of each case is unpredictable and has its characteristics. Figure I.8 lost the information of landmark ID:7 and 8, Figure I.9 lost the information of landmark ID:8, and Figure I.10 has the information of all landmarks. That is, in 2D motion cases, case 1 did not see landmark 7 and 8, case 2 did not see landmark 8, and case 3 saw all the landmarks during flight.

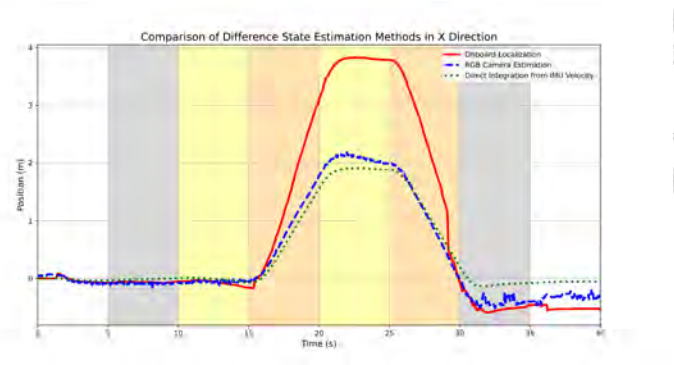
doi:10.6342/NTU202404031

Furthermore, the error comparison of all 2D cases is shown in Figure I.11. We examine the data from 9s to 31s because drone will see the landmark with high probability during this time interval and use the box plot to compare these experiments.

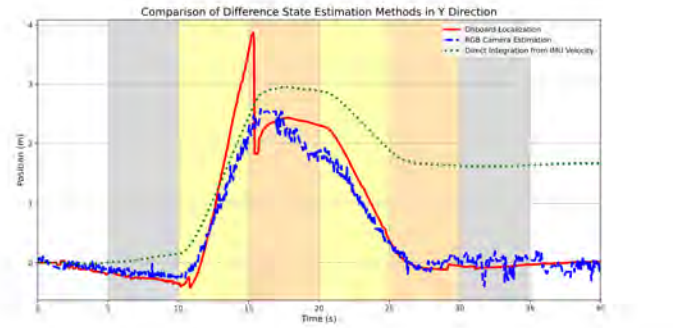
We can observe that in Figure I.8, because the drone did not see landmark 7 and 8, the position data of KF-based localization from 13s to 29s has larger error than IMU, and until 29s when the drone see the landmark 5, the position by KF will be accurate than the IMU data. In Figure I.9, because the drone did not see landmark 8, the position data of KF-based localization after 25s has large error but less than IMU. Furthermore, from 13s to 21s in case 2 between the detection of landmark 6 and 7, because it has been long time without landmark adjustment, the error by KF will be larger than IMU error. In Figure I.10, because the drone saw all the landmarks, the position data of KF-based localization has larger accuracy than IMU. Same phenomenon can be also observed in Figure I.11.

In summary, as the landmarks are in the FOV more frequently, the position error by KF-based localization is less than IMU but grows larger than IMU velocity integration when not in FOV for a certain long time. As a result, in chapter 5 of the experiment result, we use the landmarks as more as possible.

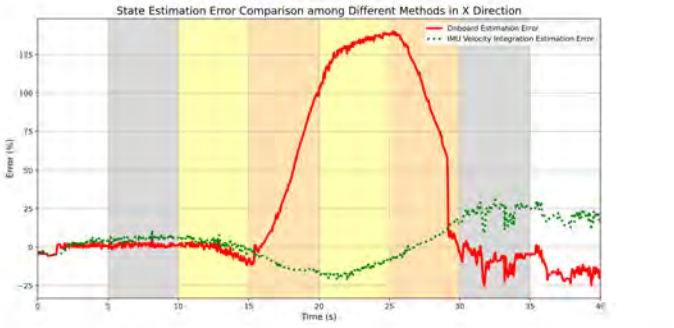
doi:10.6342/NTU202404031



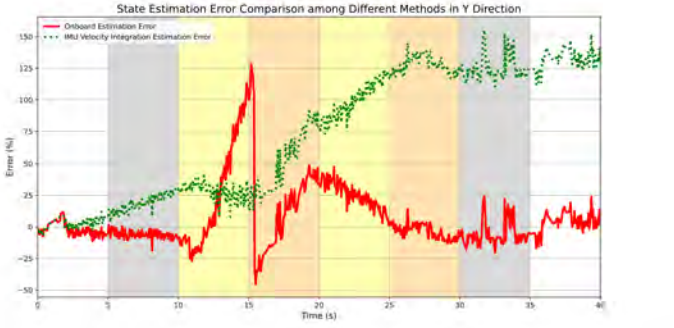
(a) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in X_W direction.



(b) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in Y_W direction.



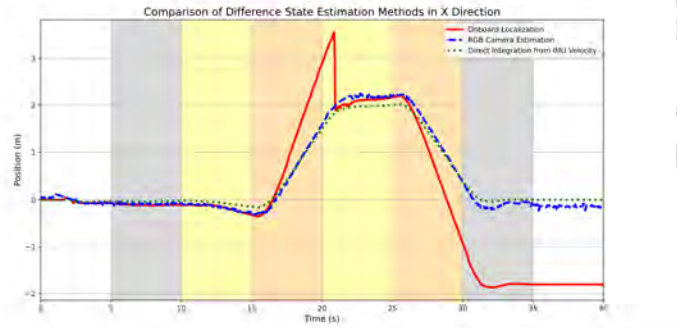
(c) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in X_W direction.



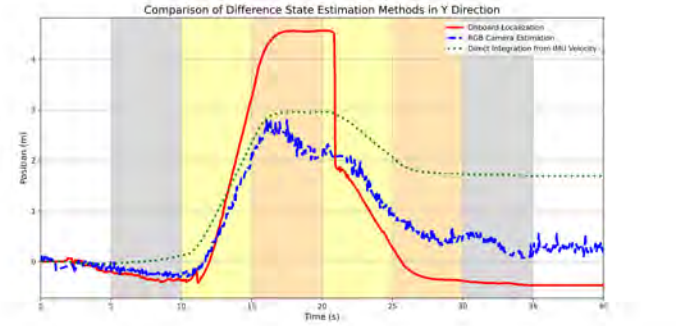
(d) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in Y_W direction.

Figure I.8: The comparison of drone positions by onboard/direct integration of IMU velocity/RGB camera estimation within time of case 1, lost of LM 7, 8, 2D motion.

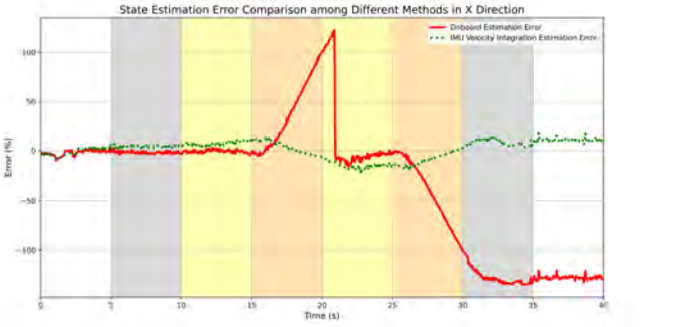
doi:10.6342/NTU202404031



(a) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in X_W direction.



(b) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in Y_W direction.

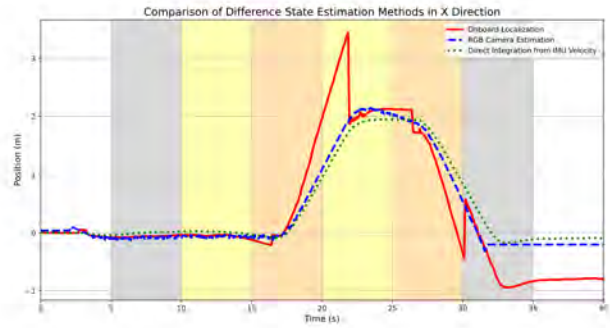


(c) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in X_W direction.

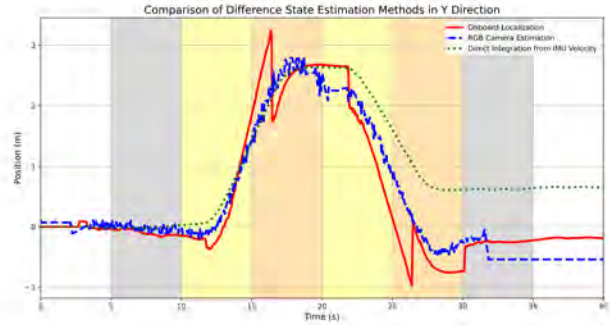
(d) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in Y_W direction.

Figure I.9: The comparison of drone positions by onboard/direct integration of IMU velocity/RGB camera estimation within time of case 2, lost of LM 8, 2D motion.

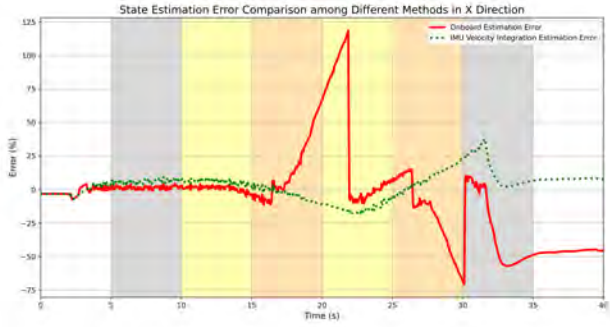
doi:10.6342/NTU202404031



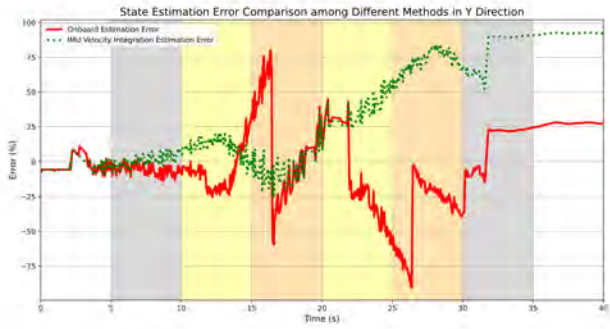
(a) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in X_W direction.



(b) Drone position by onboard/direct integration of IMU velocity/RGB camera estimation in Y_W direction.



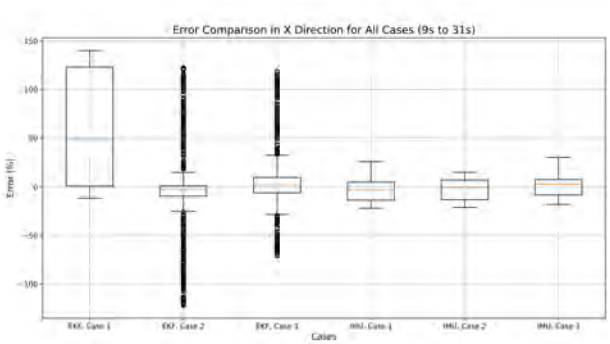
(c) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in X_W direction.



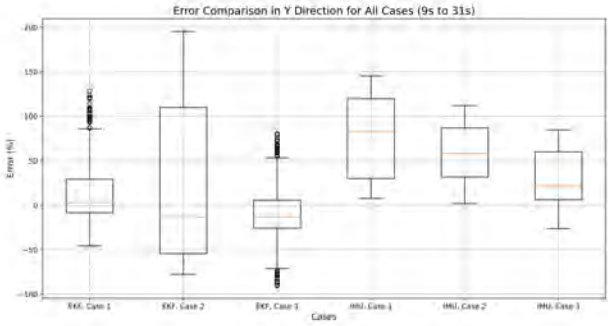
(d) Drone position error by onboard/direct integration of IMU velocity relative to RGB camera estimation in Y_W direction.

Figure I.10: The comparison of drone positions by onboard/direct integration of IMU velocity/RGB camera estimation within time of case 3, lost of nothing, 2D motion.

doi:10.6342/NTU202404031



(a) Drone position error comparison by onboard/direct integration of IMU velocity in X_W direction for all 2D cases.



(b) Drone position error comparison by onboard/direct integration of IMU velocity in Y_W direction for all 2D cases.

Figure I.11: The comparison of drone positions by onboard/direct integration of IMU velocity within time for all 2D cases.

doi:10.6342/NTU202404031

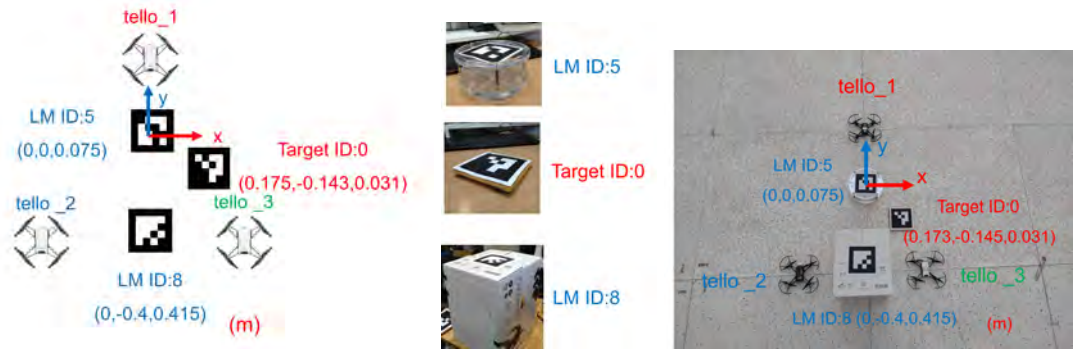
Appendix J

Consensus-Based Formation Tracking Control toward a Stationary Ground Target

In this experiemnt, we demonstrate the capability of the consensus-based formation controller of the planar motion toward a stationary ground target.

J.1 Experiment Setup

In this experiment, we utilize the marker of ID: 0 (at $(0.175, -0.143, 0.031)$(m)) as the stationary target with the markers of ID: 5 (at $(0, 0, 0.075)$(m)) and ID: 8 (at $(0, -0.4, 0.415)$(m)) on the obstacles as the landmarks with known 3D positions, as shown in Figure J.1. The desired (X_W, Y_W) positions of tello_1, tello_2, and tello_3 are $(0.175, -0.143 + 0.2\sqrt{3})$ (m), $(0.175 - 0.3, -0.143 - 0.1\sqrt{3})$ (m), and $(0.175 + 0.3, -0.143 - 0.1\sqrt{3})$ (m) respectively. The process will go through 4 steps, including



(a) The schematic figure of experiment setup. **(b) The objects in the scenario.** **(c) The real scenario figure of experiment setup.**

Figure J.1: The experiment setup.

doi:10.6342/NTU202404031

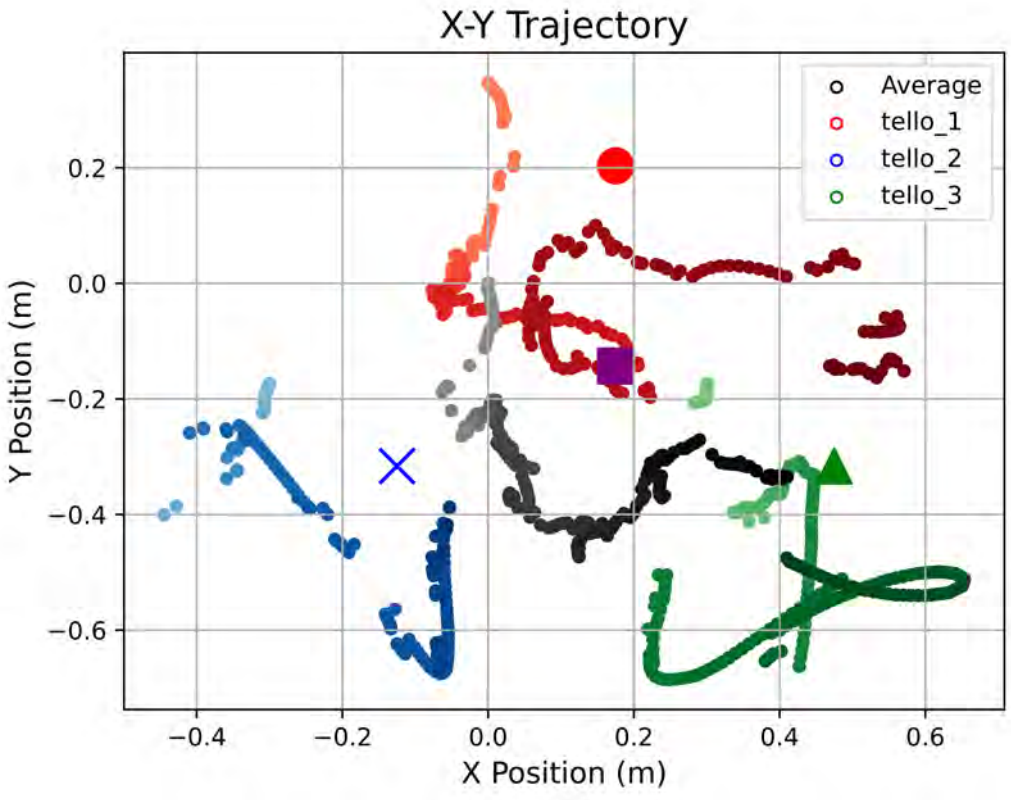


Figure J.2: Planar trajectories of the agents by KF-based localization.

1. Tello motor on (10 sec)

2. Takeoff and hovering (5 sec)

3. Controlling the planar motion (10 sec)

4. Hovering (5 sec) and landing.

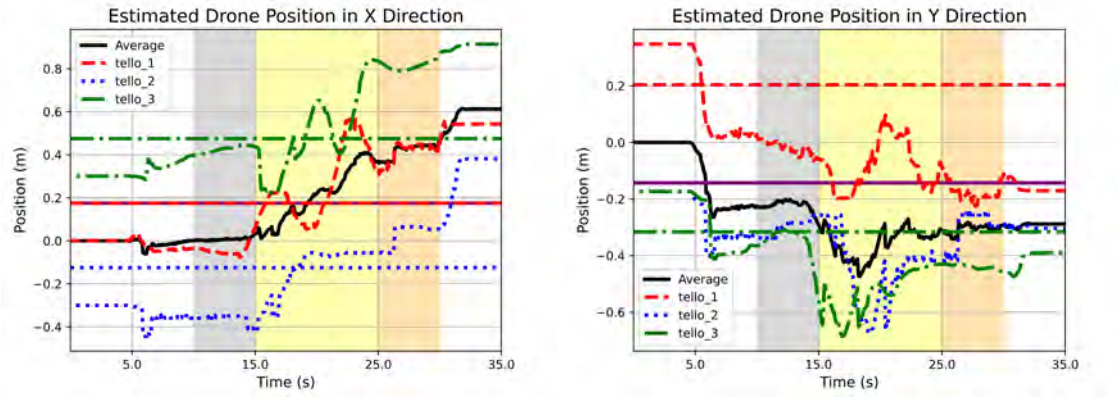
J.2 Planar Trajectories of the Agents

The trajectories of the agents is shown in Figure J.2. The square dot is the actual position of the target and the circle, cross and triangle dots are the desired position of the agents. Furthermore, the gradient curves are the trajectories of the tello_1, tello_2, tello_3 and the average position, which can be obtained from KF-based localization. We can observe that they converge to a certain formation according to the stationary target.

doi:10.6342/NTU202404031

J.3 Comparison of Tracking Positions within Time

The comparison of the X_W and Y_W positions within time can be shown in Figure J.3. The dashed, dotted and dashed-dotted curves and lines are the actual and desired positions



(a) X_W positions v.s. time. **(b) Y_W positions v.s. time.**

Figure J.3: X_W and Y_W positions of the agents within time.

of tello_1, tello_2, tello_3 respectively. The solid curves and lines represent the average positions in X_W and Y_W directions respectively and the actual positions of the target. We can observe that during the control process, the agents can control itself according to their desired positions with similar tendencies.

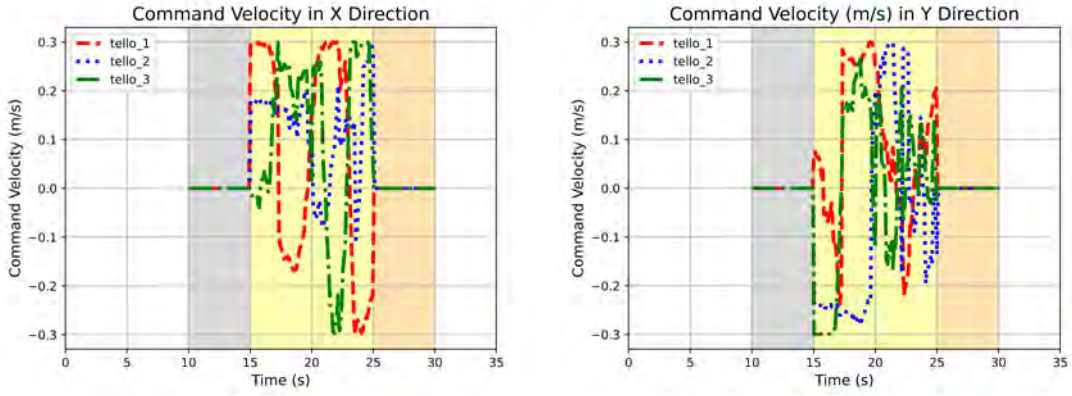
J.4 Command Velocities within Time

The command velocities in X_W and Y_W directions are shown in Figure J.4, which is also the designed control input u_{if}. We can observe that the command will execute only during the control process.

J.5 Comparison of Estimated/Predicted Target Position v.s. Time without/with Curve Fitting

Figure J.5 shows the comparison of the target position without/with prediction using

doi:10.6342/NTU202404031

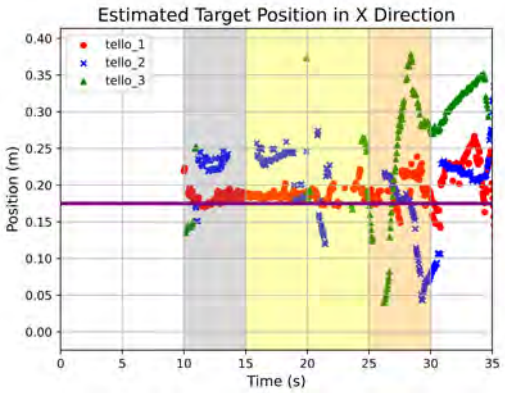


(a) X_W command velocities v.s. time. **(b) Y_W command velocities v.s. time.**

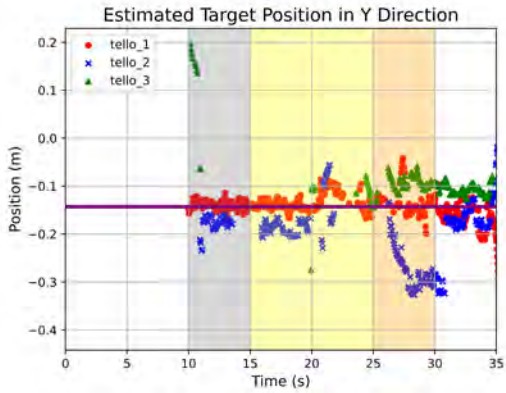
Figure J.4: X_W and Y_W command velocities of the agents within time.

curve fitting. Figure J.5(a) and Figure J.5(b) shows that during the detection process, the estimation of the target from tello_1 (circle), tello_2 (cross), tello_3 (triangle) will be lost at some time sequence unpredictably. For instance, at about 14s-16s and 22s-26s, the detection of tello_2 will fail due to some incidents such as occlusion or correction failure. Same problem occurs on tello_3 at 11s-20s and 21s-24s. However, we can compensate this kind of problem by curve fitting and predict the position of the target even if some estimation failure occurs. Consequently, the predicted position data will approach the exact position as the solid horizontal lines. Furthermore, we can observe that from about 21s-26s during long time period of estimation failure on tello_2 which will make the control process fail. Nonetheless, we can utilize the multi-agent control algorithm to make the agent detect the target again, which is our objective.

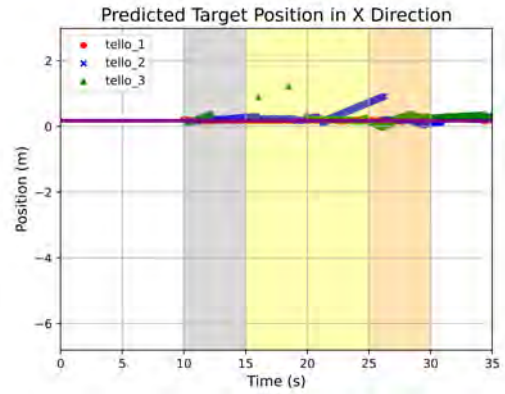
doi:10.6342/NTU202404031



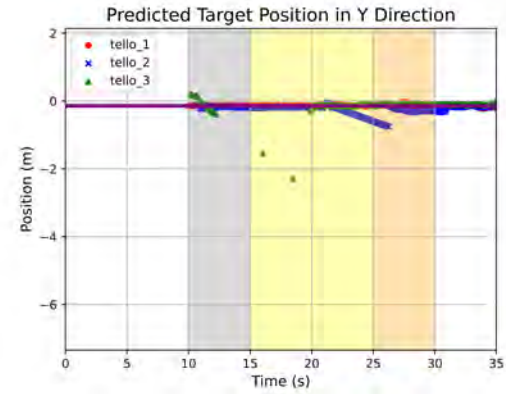
(a) Target position detected in X_W direction v.s. time.



(b) Target position detected in Y_W direction v.s. time.



(c) Target position predicted by curve fitting in X_W direction v.s. time.



(d) Target position predicted by curve fitting in Y_W direction v.s. time.

Figure J.5: Comparison of target position estimation/prediction without/with curve fitting within time.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Appendix K

LiDAR Odometry from the Pointcloud

LiDAR odometry is a method to localize itself from the pointcloud by offline SLAM. After doing SLAM, we obtain the global map. The localization framework is demonstrated based on the global map, as shown in the second part of Figure K.1. The concept of this strategy is from [84: koide3]. We can refer to this reference for more information. As for the first part of SLAM, we can refer to [85: ROS.org] for more information, including hardware and software installation. The overall steps for LiDAR odometry are

1. Performing pointcloud data (roslaunch velodyne_pointcloud VLP16_points.launch)
2. Recording the bag file with pointcloud data (rosbag record -a)
3. Performing localization package (roslaunch hdl_localization hdl_localization.launch)
4. Playing the bag file (rosbag play –clock current_bag_file_name.bag)
5. Recording the estimated LiDAR pose of in the map frame.

K.1 Experiment Setup

In this experiment, we use Velodyne VLP-16 LiDAR above the cart to perform lo-

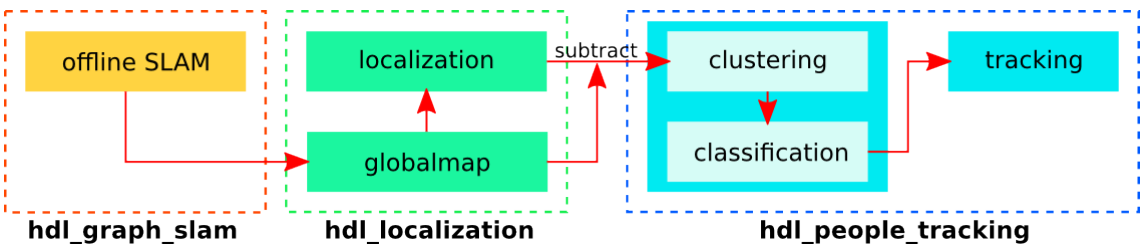


Figure K.1: The overall workflow of SLAM, localization and people tracking.

doi:10.6342/NTU202404031

(a) VLP-16 LiDAR [86: Hot Robotics]. (b) Experiment setup.

Figure K.2: The hardware for LiDAR-based localization.

calization for the ground truth of the target, as shown in Figure K.2(b). The cart will move nearly straight along Y_W direction in the world frame and the LiDAR above the cart will record the pointcloud data. We use the recorded pointcloud data for calculating the pose of the LiDAR. Note that the coordinate of the LiDAR has no rotation from the world frame.

K.2 Experiment Results

The experiment result of the position of the LiDAR within time is shown in Figure K.3. We can see in Z_W direction, the position remains nearly constant because there is no movement along this direction. In X_W direction, there is not much movement, so the position change is not such drastic as well. However, in Y_W direction, although there is a larger movement, the position data will not increase as we want. On the contrary, the position decreases, which is not an ideal situation. As a result, the localization frameworks using LiDAR is not an ideal strategy for precise localization for obtaining the ground truth of the target.

doi:10.6342/NTU202404031

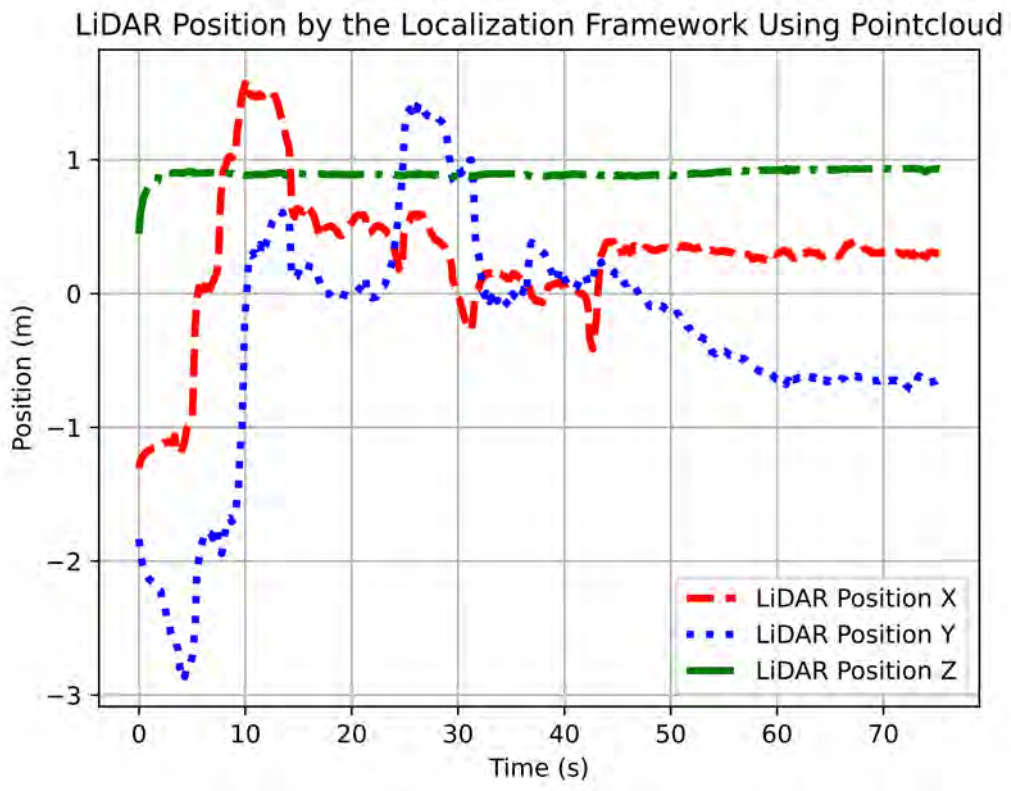


Figure K.3: LiDAR position by the localization framework using pointcloud within time.

doi:10.6342/NTU202404031

doi:10.6342/NTU202404031

Appendix L

Introduction of the Adjacency Matrix and Control Method

Adjacency matrix is a representation to describe the information flow between the agents in the multi-agent systems. We consider a graph defined as

$$\mathcal{G} = (\mathcal{V}, \varepsilon), \tag{L.1}$$

where $\mathcal{V}(vertices) = 1, 2, ..., n$ is composed of the indices of agents and $\varepsilon(edges) \subseteq \mathcal{V} \times \mathcal{V}$. If the two vertices are adjacent, an edge should exists between two vertices, which means the information will exchange with each other.

Adjacency matrix is a matrix to define this kind of protocol, which can be defined as

$$A_{adj} = \begin{bmatrix} a_{11}^v & a_{12}^v & \cdots & a_{1n}^v \\ a_{21}^v & a_{22}^v & \cdots & a_{2n}^v \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^v & a_{n2}^v & \cdots & a_{nn}^v \end{bmatrix}, \tag{L.2}$$

where $a_{ij}^v = 1$ if agent i and j are adjacent and $a_{ij}^v = 0$ if agent i and j are not adjacent. The adjacency matrix can be designed in the consensus-based formation control algorithm even if there is no information flow between some agents. That is, the control input can be further described as

$$u_i = \dot{r}_i^d - \alpha_i(r_i - r_i^d) - \sum_{j=1}^{n} a_{ij}^v[(r_i - r_i^d) - (r_j - r_j^d)], \tag{L.3}$$

doi:10.6342/NTU202404031

where $\alpha_i > 0$ is a scalar, r_i^d is the desired state of the i^{th} agent and a_{ij}^v is the (i, j) entry of the adjacency matrix A_{adj}. We can design this matrix as the information flow to achieve the desired control topology of the system. Here the states r_i and r_i^d are the position information with the velocity control input u_i.

doi:10.6342/NTU202404031

Appendix M

Stability Analysis of the Consensus-Based Formation Controller

To prove the stability of the consensus-based formation control input given by

$$u_i = \dot{r}_i^d - \alpha_i(r_i - r_i^d) - \sum_{j=1}^{n} a_{ij}^v[(r_i - r_i^d) - (r_j - r_j^d)], \tag{M.1}$$

we need to show that the system reaches the desired formation and remains stable around it. The stability analysis of the system is shown below.

First we define the formation control error as

$$e_i = r_i - r_i^d. \tag{M.2}$$

The desired goal is to let $e_i \to 0$ as $t \to \infty$.

Since the control input is a velocity command, the system dynamics can be written as

$$\dot{r}_i = u_i = \dot{r}_i^d - \alpha_i e_i - \sum_{j=1}^{n} a_{ij}^v(e_i - e_j). \tag{M.3}$$

Thus, the error dynamics is given by

$$\dot{e}_i = \dot{r}_i - \dot{r}_i^d = -\alpha_i e_i - \sum_{j=1}^{n} a_{ij}^v(e_i - e_j). \tag{M.4}$$

doi:10.6342/NTU202404031

The error dynamics can be represented the in matrix form of

$$e = [e_1, e_2, \ldots, e_n]^T. \tag{M.5}$$

Simultaneously, we let L_v be the Laplacian matrix of the graph with edge weights a_{ij}^v. The system of error dynamics can be written as

$$\dot{e} = -A_\alpha e - L_v e, \tag{M.6}$$

where $A_\alpha = \text{diag}(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

We then consider the Lyapunov candidate function and take the time derivative of it.

$$V(e) = \frac{1}{2} e^T e. \tag{M.7}$$

Thus,

$$\dot{V}(e) = e^T \dot{e} = e^T(-A_\alpha e - L_v e). \tag{M.8}$$

By simplifying it, we can obtain that

$$\dot{V}(e) = -e^T A_\alpha e - e^T L_v e. \tag{M.9}$$

Since A_α is a diagonal matrix with positive entries α_i, we have

$$e^T A_\alpha e = \sum_{i=1}^{n} \alpha_i e_i^2 > 0 \quad \text{for } e \neq 0. \tag{M.10}$$

doi:10.6342/NTU202404031

For the Laplacian matrix L_v, it is known that

$$e^T L_v e = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^v (e_i - e_j)^2 \geq 0. \tag{M.11}$$

Therefore,

$$\dot{V}(e) = -\sum_{i=1}^{n} \alpha_i e_i^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^v (e_i - e_j)^2 \leq 0. \tag{M.12}$$

Since $\dot{V}(e) \leq 0$, the Lyapunov function $V(e)$ is non-increasing, implying that e_i will not grow unbounded. Moreover, $\dot{V}(e) = 0$ if and only if $e = 0$, which means $e_i = 0$ for all i. As a result, by Lyapunov stability analysis, the error dynamics are globally asymptotically stable. As a result, the control law ensures that the agents achieve the desired formation and remain stable around it.

doi:10.6342/NTU202404031