공학석사 학위논문

간헐적 외란 및 추정 갱신 하에서의 드론의 안전비행을 위한 경로 추종 제어

Path Following Control for Safe Flight of Micro Aerial Vehicle under Intermittency in Disturbance and Estimation

2023 년 2 월

서울대학교 대학원

기계공학부

이 승 준

간헐적 외란 및 추정 갱신 하에서의 드론의 안전비행을 위한 경로 추종 제어

Path Following Control for Safe Flight of Micro Aerial Vehicle under Intermittency in Disturbance and Estimation

지도 교수 이 동 준

이 논문을 공학석사 학위논문으로 제출함
2022 년 10 월

서울대학교 대학원
기계공학부
이 승 준

이승준의 공학석사 학위논문을 인준함
2022 년 12 월

위 원 장 _____ 이 경 수 _____ (인)

부위원장 _____ 이 동 준 _____ (인)

위 원 _____ 이 윤 석 _____ (인)

Abstract

Path Following Control for Safe Flight of Micro Aerial Vehicle under Intermittency in Disturbance and Estimation

Seungjoon Lee
Mechanical Engineering
The Graduate School
Seoul National University

In this work, we present the velocity field-based path following control of UAV for the safe flight system under intermittency in disturbance and estimation. While the trajectory tracking control requires the vehicle to have desired position, velocity and acceleration at specific time, the path following control aims at just converging to and traversing the desired path. This difference makes the path following control safer than trajectory tracking control under not only intermittent disturbance like wind, but also intermittent estimation such as loop closure of SLAM. For the velocity field generation, we extend the existing work theoretically by separating coefficients, which enables the wider application of the algorithm. We also conduct realistic simulations and real-world experiments in various scenarios to verify the safety and feasibility of our system.

Keywords: Unmanned aerial vehicle (UAV), Autonomous flight, Path following, Velocity field
Student Number: *2021-21855*

Contents

List of Figures

Abbreviations

UAV	**U**nmanned **A**erial **V**ehicle
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
GPS	**G**lobal **P**ositioning **S**ystem
MoCap	**Mo**tion **Cap**ture
IMU	**I**nertial **M**easurement **U**nit
CAD	**C**omputer **A**ided **D**esign
ESC	**E**lectronic **S**peed **C**ontrollers
FCU	**F**light **C**ontrol **U**nit
PWM	**P**ulse **W**idth **M**odulation
AGV	**A**utomated **G**uided **V**ehicle
AMR	**A**utonomous **M**obile **R**obot

Physical Constants

$$\text{Standard gravity} \quad g \quad = \quad 9.806\ 65\ \text{ms}^2 \text{ (exact)}$$

Symbols

$\mathcal{C}$	Desired Path
$\mathbf{x}(t)$	Current Position of Vehicle at Time t
$\mathbf{V}(\mathbf{x}(t))$	Velocity Field Generated from $\mathbf{x}(t)$
$\mathbf{x}^*(t)$	Closest Point on $\mathcal{C}$ from $\mathbf{x}(t)$
$\mathbf{D}(\mathbf{x}(t))$	Distance Vector from $\mathbf{x}^*(t)$ to $\mathbf{x}(t)$
$D(\mathbf{x}(t))$	Norm of $\mathbf{D}(\mathbf{x}(t))$
$\mathbf{T}(\mathbf{x}(t))$	Tangent Vector of $\mathcal{C}$ at $\mathbf{x}^*(t)$
$G(\mathbf{x}(t))$	G Function Defined using $D(\mathbf{x}(t))$
$H(\mathbf{x}(t))$	H Function Defined using $G(\mathbf{x}(t))$
$\eta_{conv}(\mathbf{x}(t))$	Convergence Coefficient
$\eta_{trav}(\mathbf{x}(t))$	Traversal Coefficient
$\mathbf{v}(t)$	Current Velocity of Vehicle at Time t
$\mathbf{f}$	Desired Thrust
$\mathbf{M}$	Desired Moment

m	Total Mass
f	Norm of $\mathbf{f}$
$\mathbf{R}$	Rotation Matrix from the Body-fixed Frame to the Inertial Frame
$\mathbf{e_3}$	Basis Vector Specifying the Down Direction
$\mathbf{J}$	Inertia Matrix in the Body-fixed Frame
$\mathbf{\Omega}$	Angular Velocity in the Body-fixed Frame
$\mathbf{M_i}$	Desired Moment in Roll ($i = 1$), Pitch ($i = 2$), and Yaw ($i = 3$) Direction
c_f	Thrust - Rotor Speed Coefficient
l	Offset from Center of Mass to Each Rotor (Arm Length)
c_M	Yaw Reaction Moment - Rotor Speed Coefficient
ω_i	i-th Rotor Speed
f_i	i-th Rotor Thrust
$k_{v,vz,I,Iz,R,Rz,\Omega,\Omega z,RI,RIz}$	Controller Gains
$\mathbf{e_{v,vz,I,Iz,R,Rz,\Omega,\Omega z,RI,RIz}}$	Tracking Errors

Chapter 1

Introduction

1.1 Motivation

In recent years, unmanned aerial vehicles (UAVs) have been actively utilized in various fields due to their mobility in 3D space and affordability. Although the most-used case until now is filming such as movie shooting or sports broadcasting, industrial applications and research works are receiving substantial interests and growing rapidly. Inspection and measurement are representative cases of industrial applications, and the solutions are also being provided by companies such as Skydio [1] or DJI [2]. Both vendors offer ready-made drones with fully/semi-autonomous flight and mapping system so that they can be used for inspection of public infrastructure or for data acquisition and management in construction

FIGURE 1.1: Skydio X2 Thermal for the inspection work

site. Examples of the products are shown in Fig 1.1 and Fig 1.2. However, most applications until now are semiautonomous with intervention of manual control, not fully autonomous.

On the other hand, many research works concerning autonomous flight system have been conducted including localization, path planning, collision avoidance and control. For localization, global positioning system (GPS) is the most used system in the outdoor case, but it suffers from the signal loss in the environment like indoor or vicinity of the building. Instead, motion capture (MoCap) system is frequently used in the indoor case, which shows not only high accuracy in pose estimation, but also high rate in data acquisition. However, it also has a limit

FIGURE 1.2: DJI Phantom 4 RTK for accurate data capturing and mapping

that once the system is built, it is hard to move the system to another place so that the area where the pose can be estimated is constrained to specific place. As a countermeasure, visual odometry is one of the feasible localization methods, which is free from infrastructure and works in both indoor and outdoor. Although the visual odometry usually requires relatively high computation power and is vulnerable to lighting or scene conditions, it has been actively used and studied

in various form such as simultaneous localization and mapping (SLAM) or fusion with other sensors like inertial measurement unit (IMU).

For path planning, which often contains obstacle avoidance, A* or RRT based initial path generation and optimization method is used usually considering collision avoidance, minimum time, or minimum snap. The generated path then followed usually by trajectory tracking control. While the trajectory tracking control shows high tracking accuracy and applicability to high speed flight, it requires the vehicle to have specific pose, velocity, and acceleration at specific time, which can generate unsafe and unintended motion due to disturbance or imperfection of state estimation. Although the issue can be resolved by some algorithms in trajectory replanning level, the field engineers who apply UAVs to their fields usually do not have enough domain knowledge to implement them and in some situations, the replanning strategy could not succeed in real time.

On the other hand, the path following control, which is another method to follow the desired path, aims at just converging to and traversing the path regardless of time. In other words, the path following control can follow the desired path without supplementary algorithms even in the presence of the intermittent disturbance and estimation. Although the path following control has been commonly used in wheeled mobile robots [3]-[4], autonomous underwater vehicles [5], and manipulators [6], to our knowledge, only a few works have considered the path following control of UAVs. Therefore, we applicate the velocity field-based path following control to UAV for safe and simple autonomous flight system. To verify

the safety and feasibility of the system, we also conduct the realistic simulations and real-world experiments in various scenarios.

1.2 Related Works

Trajectory planning has been widely used and investigated for autonomous flight of UAV. Mellinger [7] proposed minimum snap trajectory generation and control algorithm for quadrotors, which generates optimal trajectories in real time with relaxation of small angle assumptions. In Mueller's work [8], the method for rapid generation of motion primitives is presented, which minimizes the cost function concerning input aggressiveness. It is shown that the method can generate and evaluate a million motion primitives per second on a general laptop computer. As a next step of trajectory generation, studies related to trajectory replanning have been widely conducted. Zhou [9] proposed perception-aware trajectory replanning algorithm based on their previous work [10], which considers potential risk of unknown obstacles by exploiting yaw planning strategy to actively observe unknown area. Furthermore, Romero [11] presented time-optimal online replanning method which enables agile flight of quadrotor over 60 km/h and endures wind disturbance up to 68 km/h. It efficiently generates trajectory with sampling-based method and tracks the trajectory using model predictive contouring control which also considers the full quadrotor dynamics and the single rotor thrust limits.

Besides trajectory tracking control, path following control is also used to navigate UAV autonomously, especially in fixed-wing UAV. As an early path following algorithm, nonlinear guidance logic was presented in Park's study [12]. It generates acceleration command that follows 2D straight line or curved path, based on the forward reference point of the desired path which is specified by pre-defined parameters. It was extended to 3D space using the Frenet-Serret frame in Cho's work [13], where the look-ahead vector is proposed by which the generated acceleration command attracts the vehicle's velocity to follow the desired path. Unlike the precedent works, vector field-based time-varying path following approach is presented in [14]. It computes artificial vector fields which allow the vehicle to converge to and circulate around n-dimensional desired curve. Based on that work, the control strategy to follow the vector field is proposed in [15] where the control inputs are thrust and angular rates. Furthermore, an obstacle avoidance method was also introduced with model predictive control (MPC) in Pereira's work [16], where the obstacles are assumed to be rigid vertical cylinders so that the vector fields are modified to circulate the obstacle's boundary.

1.3 Contribution

In this paper, we revive the velocity field-based path following control for autonomous flight of multi-rotor UAV. To generate the velocity field, we extend the algorithm of the existing work [17] theoretically. The extension is accomplished by coefficients separation, and makes our method feasible for wider applications

than the original one. Furthermore, we verify the safety and feasibility of our method by realistic simulations and experiments. For verification, we conduct comparison between our method and the trajectory tracking control under intermittent disturbance and estimation scenarios.

The rest of the paper is organized as follows. In chapter 2, we explain our velocity field generation method with the convergence analysis. We decouple the coefficients in our method and prove its independence to convergence property. In chapter 3, we describe our path following control system architecture including modeling and hardware specs of our custom hexa-rotor UAV. The system consists of velocity field generator mentioned above, and velocity tracking controller that tracks the generated velocity. Then we show the results of simulations and experiments in chapter 4. Both Gazebo [18] simulation and real-world experiment are conducted for intermittent disturbance scenario. For intermittent estimation scenario, only Gazebo simulation is conducted by assuming perceptual aliasing case. Finally, we conclude this paper with our future works in chapter 5. The future works include reactive coefficients planning which can be applicated to collision avoidance problem.

Chapter 2

Velocity Field Generation

2.1 Problem Setup

The problem setup, assumptions, and the basic elements for our velocity field generation method are almost same as those of [17] except that the desired path is time-invariant. Therefore, we only mention the problem with our own notation as below.

Problem 1. Let us define $\mathcal{C}$ as a desired path and $\mathbf{x}(t)$ as a current position of a vehicle at time t. Then, the objective is to generate a velocity field $\mathbf{V}(\mathbf{x}(t))$ that makes the trajectories of a system $\dot{\mathbf{x}}(t) = \mathbf{V}(\mathbf{x}(t))$ converge to and follow the desired path $\mathcal{C}$. ■

The detail of the velocity field calculation to solve problem 1 will be stated in section 2.2.

2.2 Methodology

For generation of the velocity field, the definitions of three elements and two functions are brought from [17] with our own notation and small modification.

Definition 1. Let us define $\mathbf{y}$ as an arbitrary point on the desired path $\mathcal{C}$. Then, the closest point $\mathbf{x}^*(t)$ on $\mathcal{C}$ from $\mathbf{x}(t)$ is defined as follows.

$$\mathbf{x}^*(t) = \underset{\mathbf{y}}{\operatorname{argmin}} \, ||\mathbf{x}(t) - \mathbf{y}|| \tag{2.1}$$

■

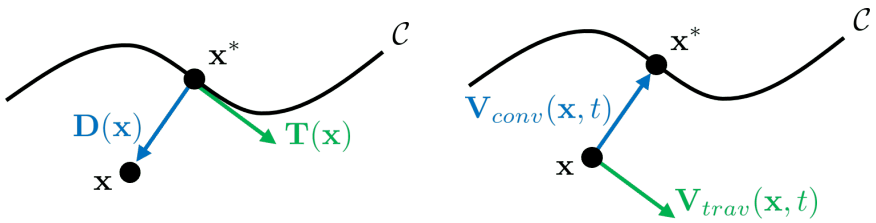
Definition 2. The distance vector $\mathbf{D}(\mathbf{x}(t))$ from $\mathbf{x}^*(t)$ to $\mathbf{x}(t)$, its norm $D(\mathbf{x}(t))$, and the tangent vector $\mathbf{T}(\mathbf{x}(t))$ of $\mathcal{C}$ at $\mathbf{x}^*(t)$ is defined as follows.

$$\mathbf{D}(\mathbf{x}(t)) = \mathbf{x}(t) - \mathbf{x}^*(t) \tag{2.2}$$

$$D(\mathbf{x}(t)) = ||\mathbf{D}(\mathbf{x}(t))|| \tag{2.3}$$

$$\mathbf{T}(\mathbf{x}(t)) = \frac{d\mathcal{C}}{d\mathbf{x}^*(t)} \tag{2.4}$$

■



(a) Calculation elements (b) Velocity components

FIGURE 2.1: Velocity field generation

Elements defined in Definition 1 and Definition 2 are described in Figure 2.1a. Also, functions $G(\mathbf{x}(t))$ and $H(\mathbf{x}(t))$ are defined using the distance $D(\mathbf{x}(t))$. Function $G(\mathbf{x}(t))$ can be any function that is Lipschitz continuous, strictly increasing, and $G(0) = 0$. The choice of both functions are given as follows.

$$G(\mathbf{x}(t)) = \frac{2}{\pi} \arctan(k_f D(\mathbf{x}(t))) \tag{2.5}$$

$$H(\mathbf{x}(t)) = \sqrt{1 - G(\mathbf{x}(t))^2} \tag{2.6}$$

In equation (2.5), k_f is a positive scalar gain that determines the convergence weight. Using these definitions and equations, the velocity field $\mathbf{V}(\mathbf{x}(t))$ to solve Problem 1 is generated as follows.

$$\begin{aligned} \mathbf{V}(\mathbf{x}(t)) = &- \eta_{conv}(\mathbf{x}(t)) G(\mathbf{x}(t)) \frac{\mathbf{D}(\mathbf{x}(t))}{D(\mathbf{x}(t))} \quad \text{(convergence)} \\ &+ \eta_{trav}(\mathbf{x}(t)) H(\mathbf{x}(t)) \mathbf{T}(\mathbf{x}(t)) \quad \text{(traversal)} \end{aligned} \tag{2.7}$$

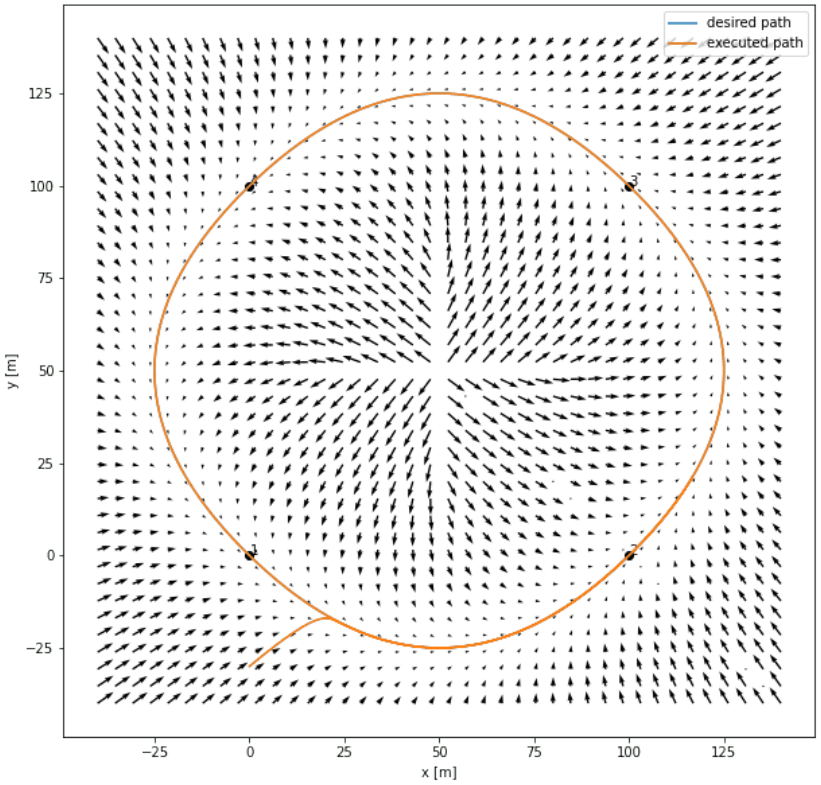


FIGURE 2.2: VF example

As written in equation (2.7) and shown in Figure 2.1b, the generated velocity consists of convergence and traversal components. $G(\mathbf{x}(t))$ and $H(\mathbf{x}(t))$ functions adjust the relative predominance of both components depending on the distance $D(\mathbf{x}(t))$. Also, $\eta_{conv}(\mathbf{x}(t))$ and $\eta_{trav}(\mathbf{x}(t))$ are the coefficients which define the maximum norm of each component. An example of velocity field generated by equation (2.7) is shown in Figure 2.2.

In the existing work [17], $\eta_{conv}(\mathbf{x}(t))$ and $\eta_{trav}(\mathbf{x}(t))$ are not separated but set to $\eta(\mathbf{x}(t))$ identically. This separation of coefficients is the key contribution of our work and largely extend the application coverage of the algorithm without any perturbation to the convergence property. Here, we can write two theorems as follows and the proof will be given in section 2.3.

Theorem 1 (Convergence of $\dot{\mathbf{x}(t)} = \mathbf{V}(\mathbf{x}(t))$)**.**
Let us define $\phi(\mathbf{x}; t)$ as a transition mapping of $\mathbf{V}(\mathbf{x}(t))$, and $\delta_{\mathbf{x}}(t) = D(\mathbf{x}(t)) \circ \phi(\mathbf{x}; t)$. Then, $\lim_{t\to\infty} \delta_{\mathbf{x}}(t) \to 0$ if $\eta_{conv}(\mathbf{x}(t)) > 0$. ■

Theorem 2. $\delta_{\mathbf{x}}(t)$ is independent of $\eta_{trav}(\mathbf{x}(t))$. ■

2.3 Convergence Analysis

Convergence analysis of Theorem 1 and Theorem 2 is not much different from the proof of [17]'s *Proposition 2* except that the desired path is time invariant in our method. Therefore, we only mention the parts of the analysis associated to our theoretical extension with our own notations.

Proof: If we consider the Lyapunov candidate function $P(\mathbf{x}(t)) = \frac{1}{2}D^2(\mathbf{x}(t))$, following results can be obtained.

$$\dot{P}(\mathbf{x}(t)) = -\eta_{conv}(\mathbf{x}(t))G(\mathbf{x}(t))D(\mathbf{x}(t)) \tag{2.8}$$

From equation 2.8 and $\dot{P}(\mathbf{x}(t)) = D(\mathbf{x}(t))\dot{D}(\mathbf{x}(t))$, we have that

$$\dot{D}(\mathbf{x}(t)) = -\eta_{conv}(\mathbf{x}(t))G(\mathbf{x}(t)) \tag{2.9}$$

Therefore, $\lim_{t\to\infty} D(\mathbf{x}(t)) \to 0$ if $\eta_{conv}(\mathbf{x}(t)) > 0$, and it proves the Theorem 1. Also, from equation (2.9), we can see that the convergence property does not depend on the traversal coefficient $\eta_{trav}(\mathbf{x}(t))$, and thus, Theorem 2 is proven as well. ■

The fact that the traversal coefficient does not affect the convergence property makes our method more flexible than the existing work and widens an application scope. For example, we can make a vehicle stop or move backward by setting the traversal coefficient to 0 or negative value, which is impossible in the existing work. Blending these motions, we can implement features such as a collision avoidance with range sensor, or hovering of multi-rotor UAVs.

2.4 Implementation

For implementation, we defined the desired paths as combination of lines, circles, and hovering which are also common in the practical applications. As mentioned in section 2.3, our method can implement the hovering task by setting $\eta_{trav} = 0$ after the previous path is terminated. Then, we smoothly ramp up η_{trav} from zero after the hovering task to proceed to the next path. This process is repeated

until the entire paths are terminated, and the details are described in Algorithm 1.

Algorithm 1 Velocity Field Generation

$i \leftarrow 0$, $n_p \leftarrow \textbf{Len}(Paths)$
while $i < n_p$ **do**
  $\mathbf{x} \leftarrow current\ position$, $Path \leftarrow Paths[i]$
  $v_{trav}, v_{conv} \leftarrow traversal\ speed, convergence\ speed$
  if $\textbf{Terminated}(Path)$ **then**
    $i \leftarrow i + 1$
    Continue
  if $Path == Hover$ **then**
    $\eta_{trav}, \eta_{conv} \leftarrow 0, v_{conv}$
  else if $t_{current} \leq t_{ramping}$ **then**
    $\eta_{trav}, \eta_{conv} \leftarrow \textbf{Ramping}(v_{trav}), v_c$
  else
    $\eta_{trav}, \eta_{conv} \leftarrow v_{trav}, v_{conv}$
  $\mathbf{x}^* \leftarrow \textbf{FindClosestPoint}(Path, \mathbf{x})$
  $\mathbf{D} \leftarrow \mathbf{x} - \mathbf{x}^*$, $\mathbf{T} \leftarrow \textbf{GetTangent}(\mathbf{x}^*)$
  $G, H \leftarrow \textbf{CalculateGH}(k_f, D)$
  $\mathbf{V} \leftarrow -\eta_{conv} G \frac{\mathbf{D}}{D} + \eta_{trav} H \mathbf{T}$

Chapter 3

System Architecture

3.1 Overview

The overview of our path following control system is described in Figure 3.1. As described in Chapter 2, the velocity field generator takes the desired path $\mathcal{C}$ and the current position $\mathbf{x}(t)$ as inputs, and generates desired velocity $\mathbf{V}(\mathbf{x}(t))$ as an output. Then, the velocity tracking controller, which will be discussed in section 3.3, takes a current velocity $\mathbf{v}(t)$ and the desired velocity $\mathbf{V}(\mathbf{x}(t))$ as inputs, and computes a desired thrust $\mathbf{f}$ and desired moment $\mathbf{M}$ as outputs to track the desired velocity. These control inputs are sent to PX4 Autopilot [19], which is an open source software for flight control of drones. In the software, the desired thrust and moment are distributed and converted to a rotation speed of

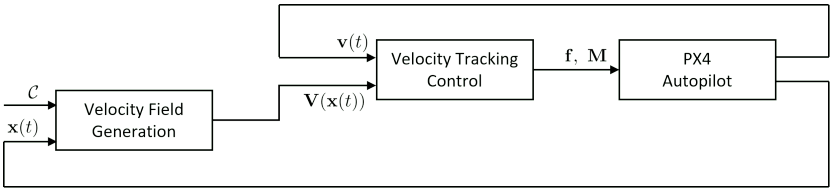


FIGURE 3.1: System overview

each rotor. This conversion is conducted using hexa-rotor dynamics which will be described in section 3.2.

3.2 Custom Hexa-rotor UAV

3.2.1 System modeling

The dynamics of multi-rotor drone can be written by the following Newton-Euler rigid body dynamics equation [20]:

$$m\ddot{\mathbf{x}} = -f\mathbf{R}\mathbf{e_3} + mg\mathbf{e_3} \tag{3.1}$$

$$\mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} = \mathbf{M}, \quad \dot{\mathbf{R}} = \mathbf{R}\hat{\mathbf{\Omega}} \tag{3.2}$$

where $m > 0$ is the mass, $f \in \Re$ is a norm of the total thrust $\mathbf{f} \in \Re^3$, $\mathbf{R} \in \mathrm{SO}(3)$ represents the rotation of the body-frame with respect to inertial frame, g is the gravitation constant, and $\mathbf{e_3} = [0, 0, 1]^T$ is the basis vector specifying the down

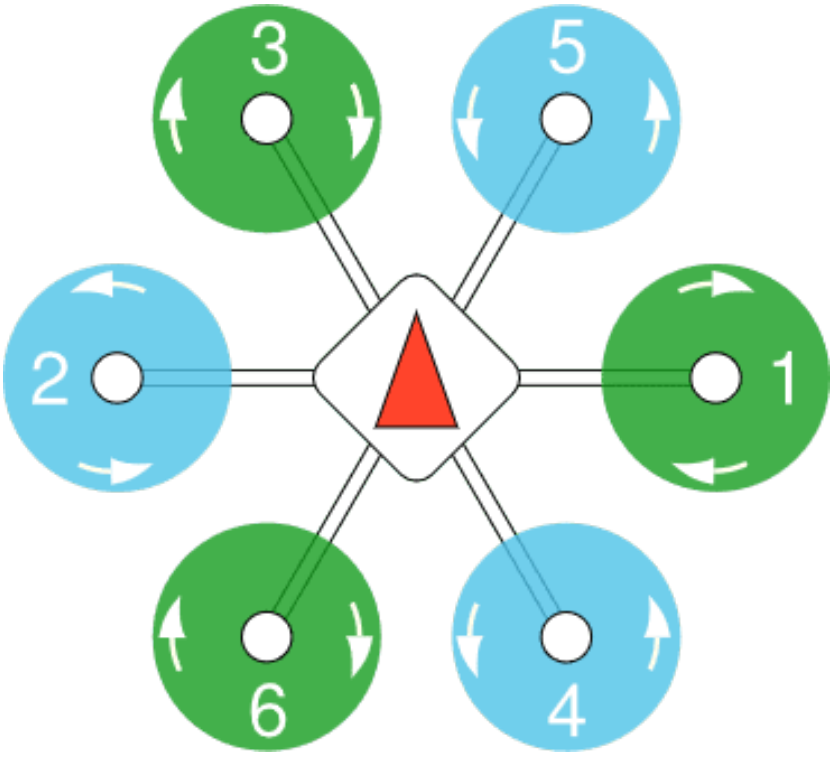


FIGURE 3.2: Hexarotor X of PX4 airframe reference

direction. Also, for equation (3.2), $\mathbf{J} \in \Re^3$ is the body-fixed rotational inertia, $\mathbf{\Omega} := [\Omega_1, \Omega_2, \Omega_3]^T \in \Re^3$ is the angular velocity of the body-frame relative to the inertial frame represented in the body frame, $\mathbf{M} := [M_1, M_2, M_3]^T \in \Re^3$ is the moment input represented in the body-frame, and the hat map $\hat{\cdot} : \Re^3 \to \mathrm{SO}(3)$ is defined by the condition that $\hat{x}y = x \times y$ for all $x, y \in \Re^3$.

In equation (3.1)-(3.2), the control inputs are the thrust f and the moment $\mathbf{M}$. For our custom hexa-rotor UAV, we refered to Hexarotor X of PX4 airframe

reference [21] which is also shown in Figure 3.2. This $(f, \mathbf{M})$ can often be written by the following simplified relation:

$$\begin{pmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{pmatrix} = \underbrace{\begin{bmatrix} c_f & c_f & c_f & c_f & c_f & c_f \\ -lc_f & lc_f & \frac{1}{2}lc_f & -\frac{1}{2}lc_f & -\frac{1}{2}lc_f & \frac{1}{2}lc_f \\ 0 & 0 & -\frac{\sqrt{3}}{2}lc_f & \frac{\sqrt{3}}{2}lc_f & -\frac{\sqrt{3}}{2}lc_f & \frac{\sqrt{3}}{2}lc_f \\ c_M & -c_M & c_M & -c_M & -c_M & c_M \end{bmatrix}}_{=:\mathbf{\Gamma}} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \\ \omega_5^2 \\ \omega_6^2 \end{pmatrix} \tag{3.3}$$

where c_f [Ns^2/rad^2] $\in \Re$ is a thrust-rotor speed coefficient, l [m] $\in \Re$ is an offset from centre of mass to each rotor (arm length), c_M [Nms^2/rad^2] $\in \Re$ is a yaw reaction moment-rotor speed coefficient, and ω_i [rad/s] $\in \Re$ is a i-th rotor speed. The PX4 Autopilot software uses a desired thrust of each rotor to control the drone. Therefore, we should rewrite the relation of equation (3.3) to compute the desired thrust of each rotor from the desired total thrust and moments. It can be implemented by computing a pseudo inverse matrix of $\mathbf{\Gamma}$ as follows.

$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} = \begin{bmatrix} \frac{1}{6} & -\frac{1}{3l} & 0 & -\frac{1}{6c_M} \\ \frac{1}{6} & \frac{1}{3l} & 0 & \frac{1}{6c_M} \\ \frac{1}{6} & \frac{1}{3l} & \frac{\sqrt{3}}{6l} & -\frac{1}{6c_M} \\ \frac{1}{6} & -\frac{1}{3l} & -\frac{\sqrt{3}}{6l} & \frac{1}{6c_M} \\ \frac{1}{6} & -\frac{1}{3l} & \frac{\sqrt{3}}{6l} & \frac{1}{6c_M} \\ \frac{1}{6} & \frac{1}{3l} & -\frac{\sqrt{3}}{6l} & -\frac{1}{6c_M} \end{bmatrix} \begin{pmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{pmatrix} \tag{3.4}$$

3.2.2 Hardware specification

We designed and made a custom hexa-rotor UAV as a hardware for the real-world experiment. A CAD model and real hardware are shown in Figure 3.3 and 3.4. Also, dimensions and specifications are shown in Table 3.1 and 3.2. The velocity field generation is conducted on onboard computer, and the control inputs to track the velocity field is computed on FCU. Then, the control inputs are converted into desired thrust of each rotor by equation (3.4), and again, these are converted into PWM signals to rotate motors via ESCs. Although we do not use in this work, RealSense D435 depth camera can be used for collision detection or SLAM. Also, GPS can be mounted on the top of our UAV for an outdoor flying.

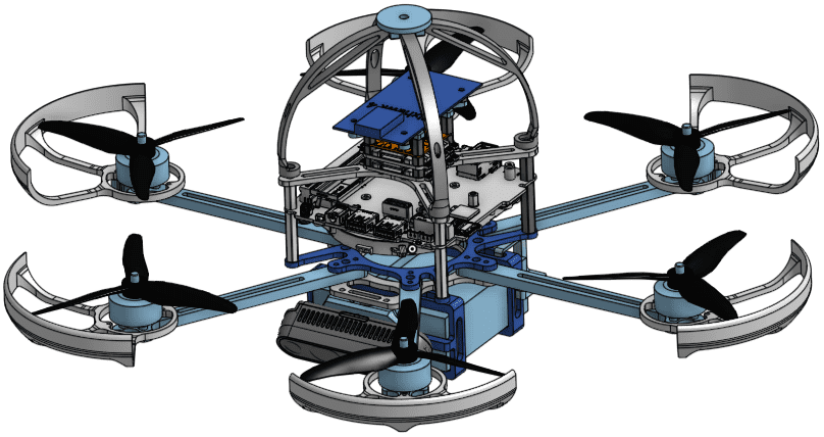


FIGURE 3.3: CAD model of custom hexa-rotor UAV

FIGURE 3.4: Real hardware of custom hexa-rotor UAV

Arm length [m]	Weight [kg]
0.171	1.39

TABLE 3.1: Dimension of custom Hexa-rotor UAV

Onboard computer	Intel NUC8i7BEH
Camera	Intel RealSense Depth Camera D435
ESC	Flycolor X-Cross 60A 3-6S BLHeli32 4-in-1 ESC
FCU	Holybro Pixhawk 4
Power distribution board	Holybro Micro Power Module (PM06 v2)
Motor	T-MOTOR P2306 V2 KV2550
Propeller	Gemfan Hurricane MCK 51466-3

TABLE 3.2: Specs of custom Hexa-rotor UAV

3.3 Velocity Tracking Control

To track the desired velocity generated by equation (2.7), we use Geometric Tracking Control [22] method as a velocity tracking controller. The original controller computes the desired thrust and moment as control inputs by using position error, velocity error, rotation error, and angular velocity error. In our case, there is no desired position but desired velocity, and thus, with the addition of the integral term and the separation of xy and z terms, the control inputs are calculated as follows.

$$
\begin{aligned}
\mathbf{f} &= -\left(-k_v\mathbf{e_v} - k_{vz}\mathbf{e_{vz}} - k_I\mathbf{e_I} - k_{Iz}\mathbf{e_{Iz}} - mg\mathbf{e_3}\right)\cdot\mathbf{Re_3} \\
\mathbf{M} &= -k_R\mathbf{e_R} - k_{Rz}\mathbf{e_{Rz}} - k_\Omega\mathbf{e_\Omega} - k_{\Omega z}\mathbf{e_{\Omega z}} - k_{RI}\mathbf{e_{RI}} - k_{RIz}\mathbf{e_{RIz}} \\
&\quad + \mathbf{\Omega}\times\mathbf{J\Omega} - \mathbf{J}(\hat{\mathbf{\Omega}}\mathbf{R}^T\mathbf{R_d}\mathbf{\Omega_d} - \mathbf{R}^T\mathbf{R_d}\dot{\mathbf{\Omega}}_\mathbf{d})
\end{aligned}
\tag{3.5}
$$

In equation (3.5), $\mathbf{e_v}, \mathbf{e_R}, \mathbf{e_\Omega} \in \Re^2$ are tracking error of velocity, rotation, and angular velocity for xy elements, $\mathbf{e_{vz}}, \mathbf{e_{Rz}}, \mathbf{e_{\Omega z}} \in \Re$ are those for z element, $\mathbf{e_I}, \mathbf{e_{RI}} \in \Re^2$ are integral error related to velocity and rotation for xy elements, and $\mathbf{e_{Iz}}, \mathbf{e_{RIz}} \in \Re$ are those for z element. Also, $k_v, k_{vz}, k_R, k_{Rz}, k_\Omega, k_{\Omega z}, k_I, k_{Iz}, k_{RI}, k_{RIz} \in \Re$ are corresponding controller gains and subscript d means "desired".

Chapter 4

Results

In this section, we show the verification results of our path following control by comparing to existing trajectory tracking control in various scenarios. The first scenario emulates an intermittent disturbance by assuming the UAV flies in circular path under the situation that wind blows intermittently. The second scenario emulates an intermittent estimation by assuming a perceptual aliasing situation while the UAV patrols in predefined path. Both Gazebo simulation and real-world experiment are conducted for the first scenario, and the only Gazebo simulation is conducted for the second scenario. For Gazebo simulation, we customized Gazebo's Typhoon H480 hexa-rotor model [23] shown in Figure 4.1. For real-world experiment, we used our custom hexa-rotor UAV shown in Figure 3.4. Also, parameters and controller gains for velocity field generation and velocity tracking controller are described in Table 4.1a and Table 4.1b, which

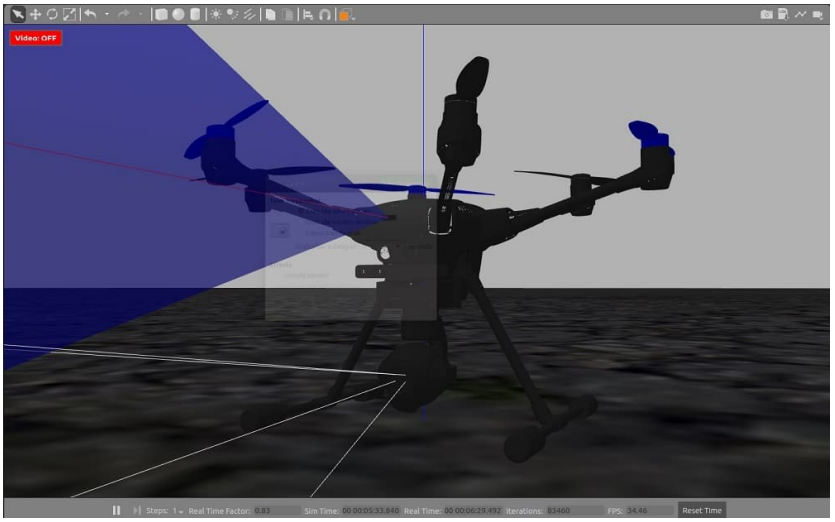


FIGURE 4.1: Gazebo's Typhoon H480 hexa-rotor model

η_{conv}	0.20	η_{trav}	0.20
k_v	8.50	k_{vz}	11.60
k_R	5.00	k_{Rz}	1.00
k_Ω	0.50	$k_{\Omega z}$	0.30
k_I	4.00	k_{Iz}	10.00
k_{RI}	0.00	k_{RIz}	0.00
m [kg]	2.25	l [m]	0.24

(a) Gazebo simulations

η_{conv}	0.20	η_{trav}	0.20
k_v	5.30	k_{vz}	10.10
k_R	1.50	k_{Rz}	0.50
k_Ω	0.21	$k_{\Omega z}$	0.15
k_I	0.00	k_{Iz}	5.00
k_{RI}	0.50	k_{RIz}	0.00
m [kg]	1.39	l [m]	0.17

(b) Real-world experiment

TABLE 4.1: Parameters and controller gains

are used in the Gazebo simulations and real-world experiment in order. The convergence gain k_f of equation (2.5) is set to 8.0 for the both.

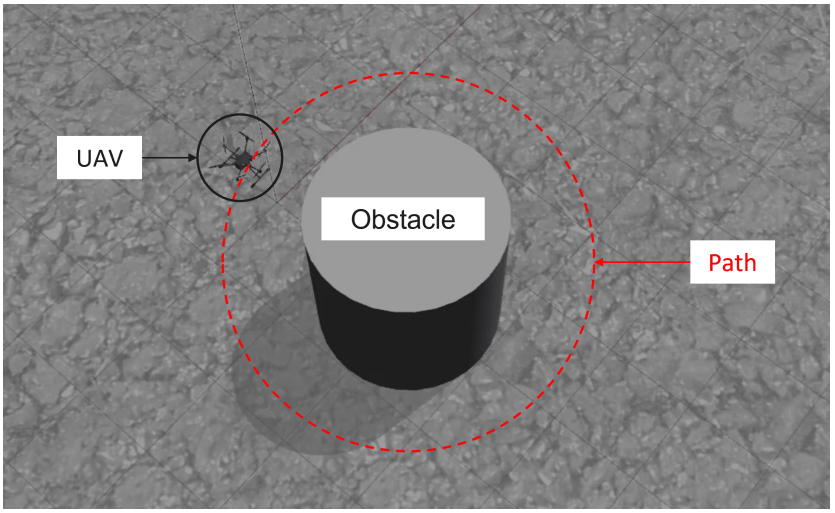
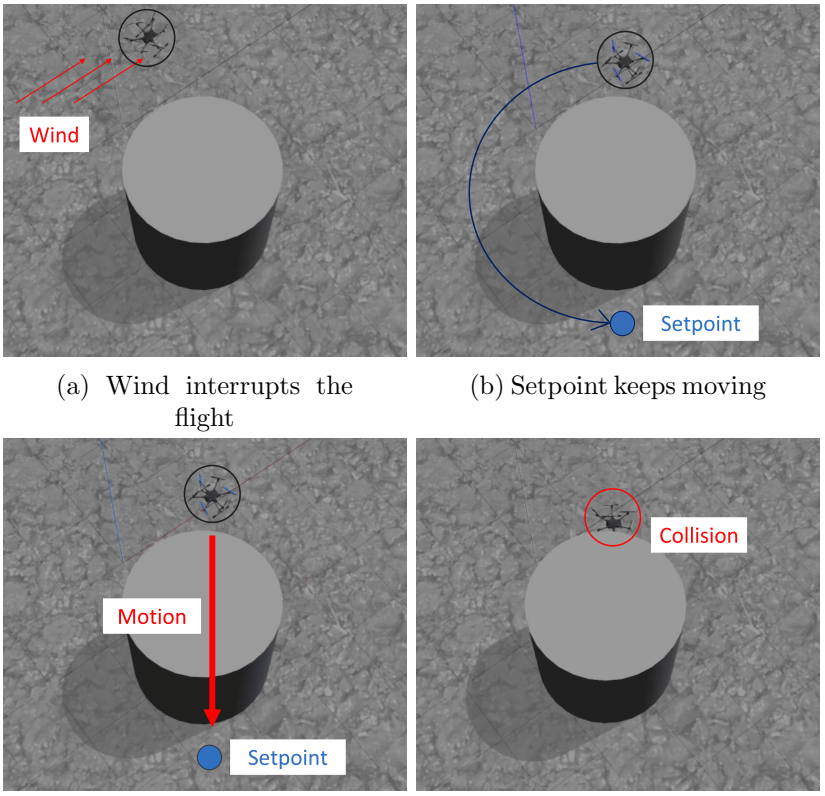


FIGURE 4.2: Gazebo simulation setup for the intermittent disturbance

4.1 Intermittent Disturbance

For verification under the intermittent disturbance, we implemented both Gazebo simulation and real-world experiment in same scenario as follows. First, the UAV starts flight in a circular path. The path has an obstacle in the center of it. Next, turn on the wind at specific time to interrupt the flight of the UAV. After a while, turn off the wind and observe how different motions are in existing trajectory tracking control and our path following control.



(a) Wind interrupts the flight

(b) Setpoint keeps moving

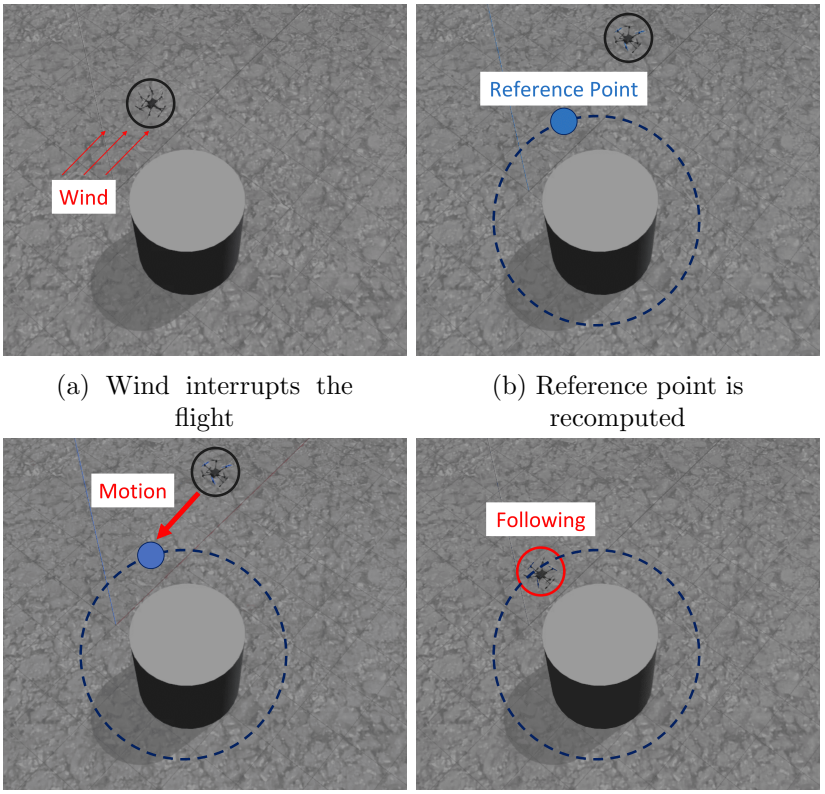
(c) UAV tries to track the setpoint

(d) Collision occurs

FIGURE 4.3: Gazebo simulation results for the trajectory tracking control under the intermittent disturbance

4.1.1 Gazebo simulation

We used Gazebo wind plug-in to generate a wind for specific time duration, and the simulation setup is shown in Figure 4.2. The results of the trajectory tracking control is shown in Figure 4.3. Figure 4.3a shows the moment that the wind is



(a) Wind interrupts the flight

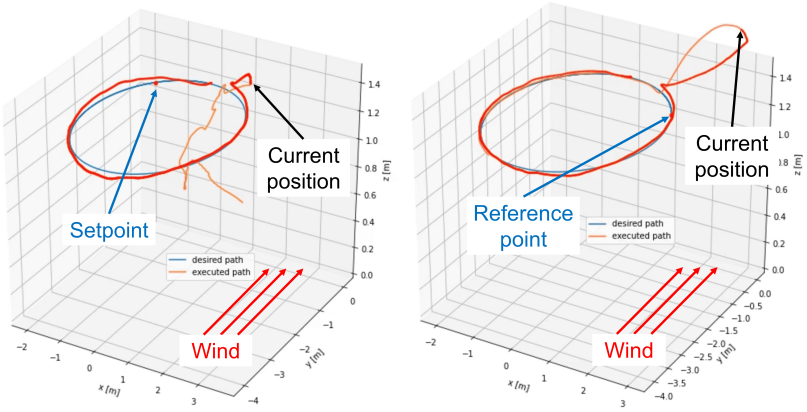
(b) Reference point is recomputed

(c) UAV flies to new reference point

(d) Converge to and follow the original path

FIGURE 4.4: Gazebo simulation results for the path following control under the intermittent disturbance

turned on and the flight of the UAV is interrupted by the wind. Meanwhile, the setpoint keeps moving in circular trajectory over time as shown in Figure 4.3b. After the wind is turned off, the UAV flies in a direction that tracks the setpoint, which makes the UAV fly through the obstacle, and as a result, the UAV collides to the obstacle as shown in Figure 4.3c - 4.3d.



(a) Trajectory tracking (b) Path following

FIGURE 4.5: Setpoint and reference point under the wind disturbance in Gazebo simulation

On the other hand, as shown in Figure 4.4, the path following control shows more safe motion than trajectory tracking control. Although the flight of the UAV is interrupted as well as the case of the trajectory tracking control, the reference point is recomputed depending on the UAV's current position as shown in Figure 4.4b. The recomputed reference point is the closest point on the path from the UAV's current position, and thus, we can see from Figure 4.4c that the generated motion does not have a risk of collision to the obstacle. As a result, the UAV converges and follows the original path without collision as shown in Figure 4.4d.

This difference can be also shown by position data in plots as Figure 4.5 which are snapshots of animated plots at the moment that the flight is interrupted. The blue line represents the desired path, the orange line represents the executed path, and the red line represents the trace until the snapped moment. The setpoint

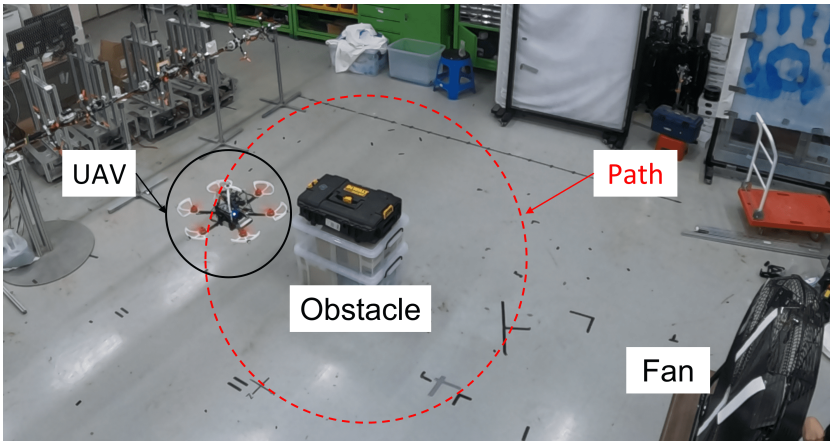
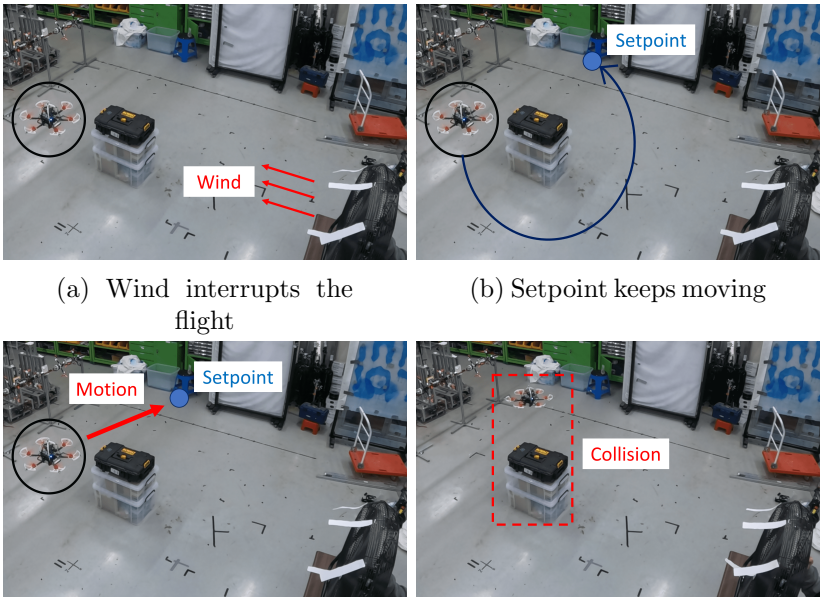


FIGURE 4.6: Real-world experiment setup for the intermittent disturbance

of the trajectory tracking control and the reference point of the path following control are marked in red dots. From Figure 4.5a and 4.5b, we can see that during the interruption of flight, while the setpoint of the trajectory tracking control proceeds along the desired path, the reference point of the path following control stays on the closest point until the UAV returns to the desired path. Consequently, as displayed in the orange line, UAV collides to the obstacle and falls in the trajectory tracking control, and on the other hand, UAV returns to and follows the desired path in the path following control.

4.1.2 Real-world experiment

The setup for real-world experiment is shown in Figure 4.6. The scenario is same as that of Gazebo simulation which is described in section 4.1.1. The only



(a) Wind interrupts the flight

(b) Setpoint keeps moving

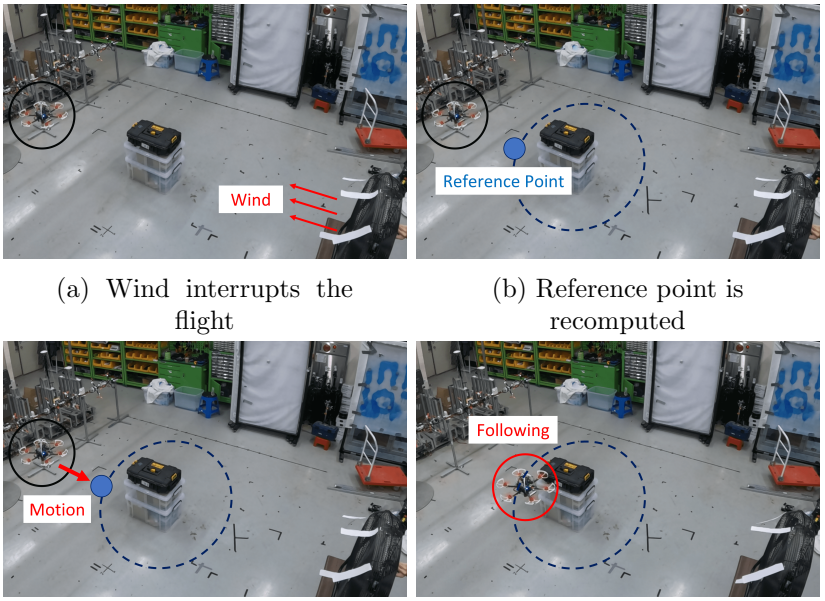
(c) UAV tries to track the setpoint

(d) Collision occurs

FIGURE 4.7: Real-world experiment results for the trajectory tracking control under the intermittent disturbance

difference is that a flight altitude is higher than a height of the obstacle to avoid real collision for safety. First, the result of trajectory tracking control is shown in Figure 4.7. Similar to the result of the Gazebo simulation, the setpoint keeps moving (Figure 4.7b) while the flight is being interrupted so that the UAV flies above the obstacle (Figure 4.7c), which could be the collision if the flight altitude was lower than the height of the obstacle (Figure 4.7d).

On the other hand, as shown in Figure 4.8, the new reference point is set to the closest point on the desired path from the current position of the UAV. Therefore,



(a) Wind interrupts the flight

(b) Reference point is recomputed

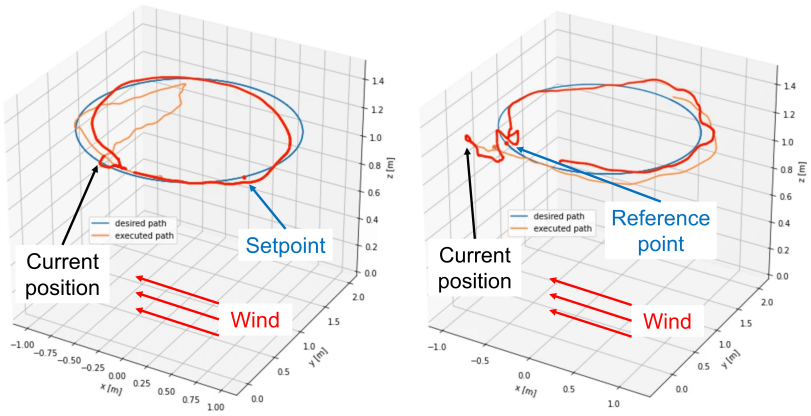
(c) UAV flies to new reference point

(d) Converge to and follow the original path

FIGURE 4.8: Real-world experiment results for the path following control under the intermittent disturbance

similar to the result of the Gazebo simulation, UAV was able to converge to and follow the original path without colliding to the obstacle.

Also, similar to the Figure 4.5, computations of the setpoint and the reference point can be seen in Figure 4.9. In trajectory tracking control, the setpoint proceeds to the opposite side of the circular path, and due to the influence of the wind, the UAV flies to y direction where the obstacle exists to track the setpoint. In path following control, the reference point is recomputed depending on the



(a) Trajectory tracking (b) Path following

FIGURE 4.9: Setpoint and reference point under the wind disturbance in real-world experiment

current position, and thus, the UAV converges to and follow the original path after the wind is turned off.

4.2 Intermittent Estimation

For verification under the intermittent estimation, we implemented only the Gazebo simulation as the following scenario. First, we assumed the situation in which the UAV is commanded to patrol the warehouse-like environment with SLAM. Second, we also assumed that the pose estimation from SLAM is incorrect due to the perceptual aliasing at the beginning. The perceptual aliasing is a phenomenon that different places generate a similar visual footprint so that

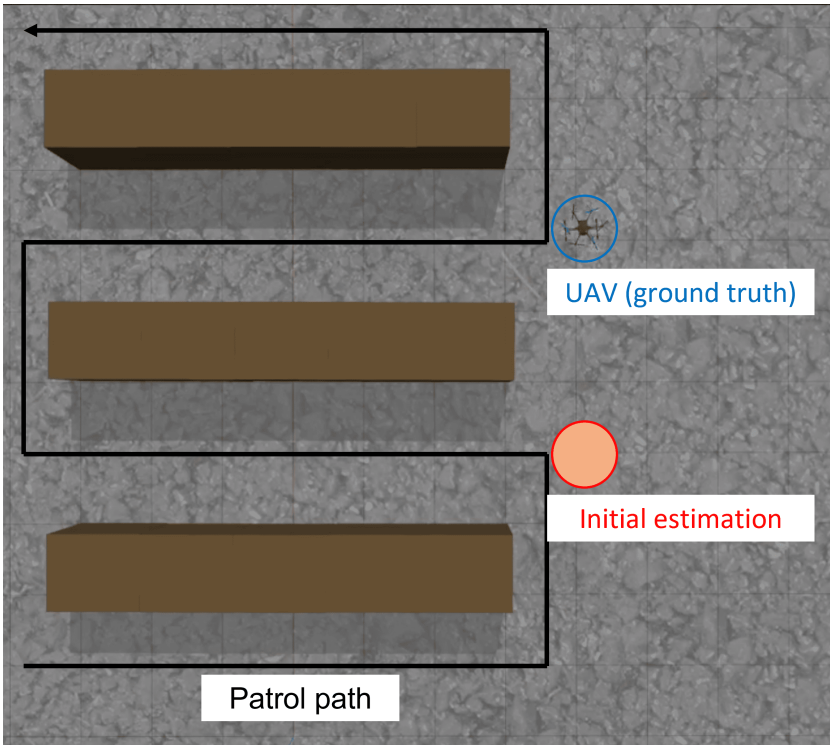
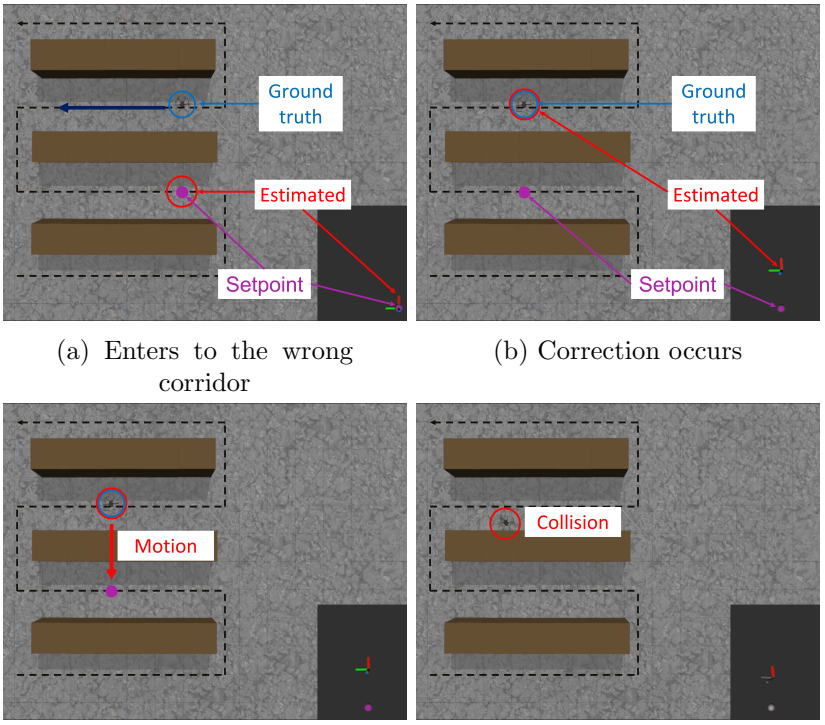


FIGURE 4.10: Gazebo simulation setup for the intermittent estimation

it causes the SLAM to result in incorrect estimation [24]. Third, due to the incorrect initial pose, the UAV enters to the wrong corridor. Here, the desired corridor is the second corridor and the wrong corridor is the third corridor from the top. Finally, the correction of the estimation occurs in the middle due to some distinguishable features. Under this scenario, we observed the difference of the trajectory tracking control and the path following control. The simulation setup is shown in Figure 4.10.



(a) Enters to the wrong corridor

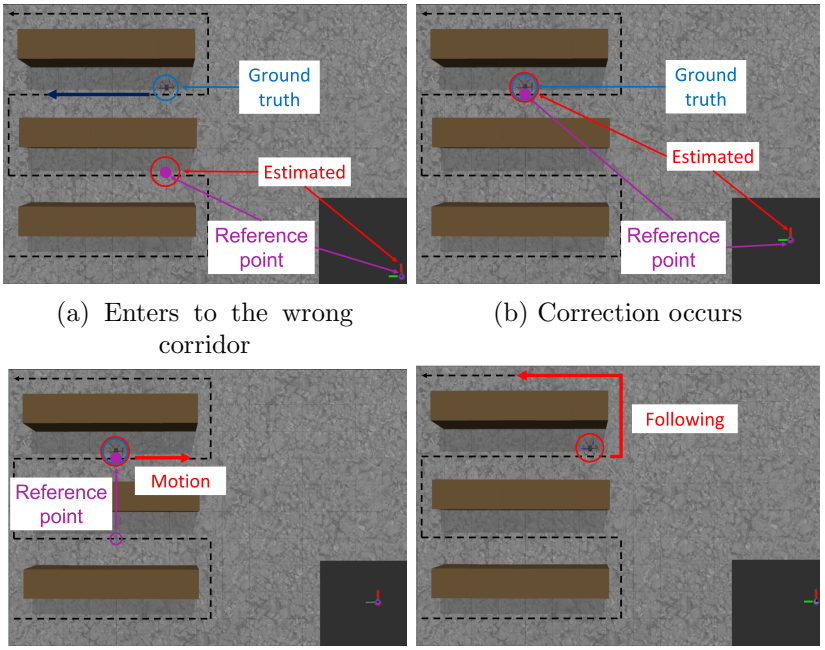
(b) Correction occurs

(c) UAV tries to track the setpoint

(d) Collision occurs

FIGURE 4.11: Gazebo simulation results for the trajectory tracking control under the intermittent estimation

First, the result of the trajectory tracking control is shown in Figure 4.11. At the beginning, while the ground truth is in the second corridor, estimated pose is in the third corridor due to the perceptual aliasing. As a result, the UAV flies to the left which is the patrol direction of the third corridor as shown in Figure 4.11a. When the correction occurs, the estimated pose becomes consistent with the ground truth (Figure 4.11b). However, the setpoint is still in the third



(a) Enters to the wrong corridor

(b) Correction occurs

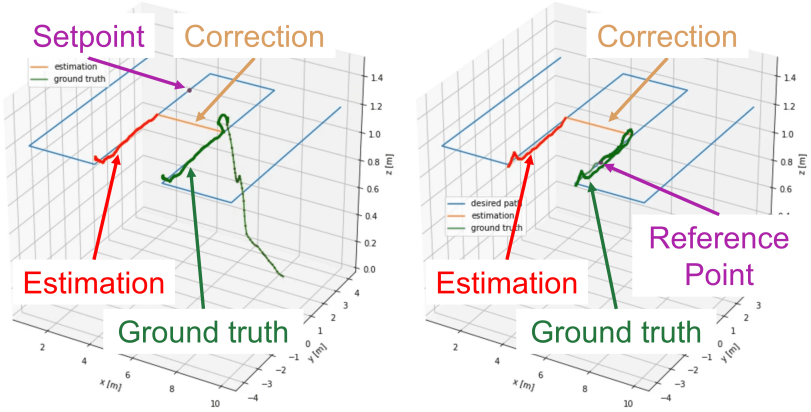
(c) Reference point is recomputed

(d) Converge to and follow the new path

FIGURE 4.12: Gazebo simulation results for the trajectory tracking control under the intermittent estimation

corridor and keeps moving to the left, which makes the UAV flies to the third corridor through the obstacles and the collision occur (Figure 4.11c-4.11d).

Next, the result of the path following control is shown in Figure 4.12. Unlike the case of the trajectory tracking control, the reference point is recomputed to the closest point from the corrected pose estimation so that the new reference point is on the patrol path of the second corridor (Figure 4.12b-4.11c). As a result, the



(a) Trajectory tracking (b) Path following

FIGURE 4.13: Setpoint and reference point under the estimation update in Gazebo simulation

UAV follows the new path segment in the desired direction without any collision (Figure 4.12d).

Finally, snapshots of animated position plots are attached in Figure 4.13. Here, the blue line represents the entire desired path, the green and the red lines represent the trace of the ground truth and the estimated pose until the snapped moment, and the purple dot represents the setpoint in the trajectory tracking control and the reference point in the path following control. Also, the orange line represents the entire estimation so that we can know that the correction occurs where the orange line jumps from the second corridor to the third corridor. From Figure 4.13a, we can see that the setpoint keeps moving in the third corridor after the correction occurs. As a result, the UAV collides to the obstacles and falls which can be also known by the falling green line. On the other hand, the

reference point is reset to the path segment in the second corridor as shown in Figure 4.13b, which prevents the collision and makes the UAV fly in the right direction.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this paper, we presented the velocity field-based path following control system for the safe flight of UAV. The system consists of the velocity field generation and the velocity tracking control. For the velocity field generation, we theoretically extend the existing work [17] by separating the traversal coefficient and the convergence coefficient. Then, we showed that the coefficients separation does not have any effect on the convergence analysis. We also showed that the convergence property is independent of the traversal coefficient, which makes our method more flexible and applicable than the existing method. For the velocity tracking control, we used the Geometric Tracking Control [22] with the modifications such

as the addition of the integral term and the separation of controller gains to xy and z components.

To verify that our method is safer than the trajectory tracking control which is the most common method for the autonomous flight, we conducted the Gazebo simulations and the real-world experiment under the intermittent disturbance and estimation. We used the Gazebo's Typhoon H480 hexa-rotor model for the simulations and the custom hexa-rotor UAV for the experiment. In the intermittent disturbance scenario, the UAV safely followed the desired path after the wind had turned off in the path following control, while the collision occurred in the trajectory tracking control. Also when the correction of the pose estimation occurred in the intermittent estimation scenario, the UAV followed the new path segment in the path following control, while the collision occurred again in the trajectory tracking control.

5.2 Future Work

In future works, we intend to applicate our method to wider area by presenting the coefficient planning method. One example of the possible applications is AGV-like collision avoidance. The AGV follows the desired path marked by such as wires, radio waves, or lasers as shown in Figure 5.1 [25]. When the obstacle is on the path so that the AGV can not proceed, different from AMR, it stops and

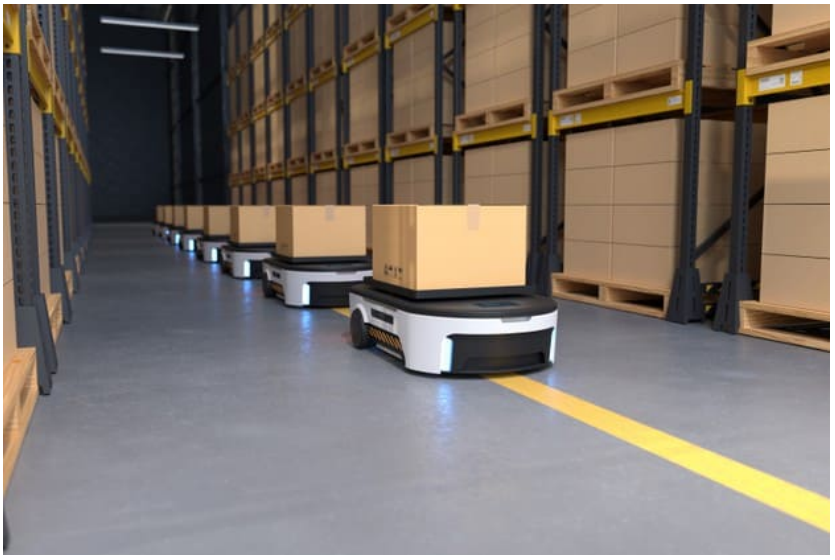


FIGURE 5.1: Automated guided vehicle (AGV)

waits until the obstacle is removed. Based on the same idea, we can simply implement the collision avoidance by reactive coefficient planning without complex algorithms.

To show that our method can be applicated in such situation, we conducted simple simulation that changes the flying direction during the flight by changing the traversal coefficient. The simulation result is shown in Figure 5.2. First, we commanded the UAV to fly from $(0.0, 0.0, 1.2)$ to $(10.0, 0.0, 1.2)$ with the value of traversal coefficient $\eta_{trav} = 0.2$. Then we changed the traversal coefficient to $\eta_{trav} = -0.2$ at the time of $15s$ when it arrived to $(3.0, 0.0, 1.2)$. As we can see in the plot, the UAV successfully changed direction and flew backward although the

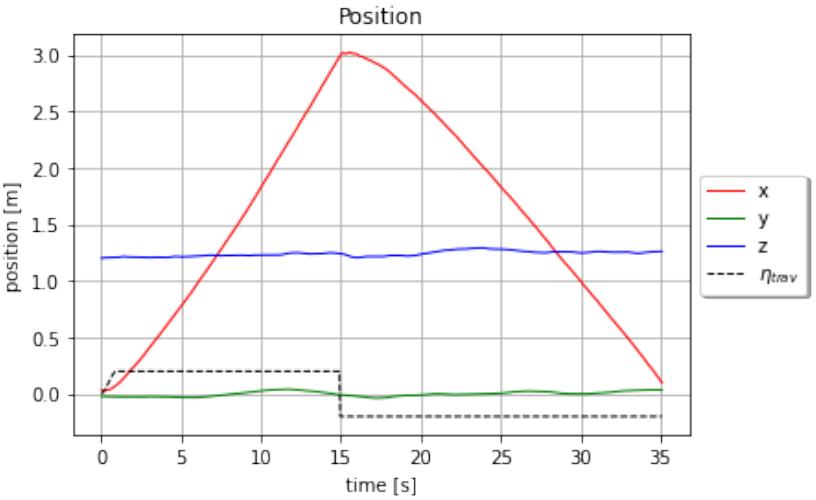


FIGURE 5.2: Position plot with the traversal coefficient transition

traversal coefficient was changed discontinuously. This result indicates that our method can be applicated for various purposes using reactive coefficient planning.

Bibliography

[1] Skydio 2+™ and x2™ - skydio inc. https://www.skydio.com. Accessed: 2022-12-19.

[2] Dji - official website. https://www.dji.com. Accessed: 2022-12-19.

[3] Yuri A. Kapitanyuk, Anton V. Proskurnikov, and Ming Cao. A guiding vector-field algorithm for path-following control of nonholonomic mobile robots. In *IEEE Transactions on Control Systems Technology*, pages 1372–1385, 2018.

[4] Ji-Wook Kwon, Cheol-Joong Kim, and Dongkyoung Chwa. Vector field trajectory tracking control for wheeled mobile robots. In *IEEE International Conference on Industrial Technology*, pages 1–6, 2009.

[5] Bilal Abdurahman, A. Savvaris, and A. Tsourdos. A comparison between guidance laws for auvs using relative kinematics. In *IEEE OCEANS 2017 - Aberdeen*, pages 1–6, 2017.

[6] Murilo Marques Marinho, Bruno Vilhena Adorno, Kanako Harada, and Mamoru Mitsuishi. Dynamic active constraints for surgical robots using vector-field inequalities. In *IEEE Transactions on Robotics*, pages 1166–1185, 2019.

[7] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[8] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. A computationally efficient motion primitive for quadrocopter trajectory generation. In *IEEE Transactions on Robotics*, pages 1294–1310, 2015.

[9] Boyu Zhou, Jie Pan, Fei Gao, and Shaojie Shen. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, pages 1992–2009, 2021.

[10] Boyu Zhou, Fei Gao, Jie Pan, and Shaojie Shen. Robust real-time uav replanning using guided gradient-based optimization and topological paths. In *IEEE International Conference on Robotics and Automation*, pages 1208–1214, 2020.

[11] Angel Romero, Robert Penicka, and Davide Scaramuzza. Time-optimal online replanning for agile quadrotor flight. In *IEEE Robotics and Automation Letters*, pages 7730–7737, 2022.

[12] Sanghyuk Park, John Deyst, and Jonathan How. A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[13] Namhoon Cho, Youdan Kim, and Sanghyuk Park. Three-dimensional nonlinear differential geometric path-following guidance law. In *Journal of Guidance, Control, and Dynamics*, pages 2366–2385, 2015.

[14] Vinícius M. Goncalves, Luciano C. A. Pimenta, Carlos A. Maia, Bruno C. O. Dutra, and Guilherme A. S. Pereira. Vector fields for robot navigation along time-varying curves in n -dimensions. *IEEE Transactions on Robotics*, pages 647–659, 2010.

[15] Adriano M. C. Rezende, Vinicius M. Gonçalves, Arthur H. D. Nunes, and Luciano C. A. Pimenta. Robust quadcopter control with artificial vector fields. In *IEEE International Conference on Robotics and Automation*, pages 6381–6387, 2020.

[16] Leonardo A. A. Pereira, Arthur H. D. Nunes, Adriano M. C. Rezende, Vinicius M. Gonçalves, Guilherme V. Raffo, and Luciano C. A. Pimenta. Collision-free vector field guidance and mpc for a fixed-wing uav. In *IEEE International Conference on Robotics and Automation*, pages 176–182, 2021.

[17] Adriano M. C. Rezende, Vinicius M. Goncalves, and Luciano C. A. Pimenta. Constructive time-varying vector fields for robot navigation. *IEEE Transactions on Robotics*, pages 852–867, 2022.

[18] Gazebo. https://gazebosim.org/home, . Accessed: 2022-12-29.

[19] Github - px4/px4-autopilot: Px4 autopilot software. https://github.com/PX4/PX4-Autopilot, . Accessed: 2022-12-28.

[20] Hyunsoo Yang, Yongseok Lee, Sang-Yun Jeon, and Dongjun Lee. Multi-rotor drone tutorial: systems, mechanics, control and state estimation. *Intelligent Service Robotics*, 2017.

[21] Airframes reference — px4 user guide. https://docs.px4.io/main/en/airframes/airframe_reference.html#hexarotor-x, . Accessed: 2022-12-28.

[22] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric tracking control of a quadrotor uav on se(3). *IEEE Conference on Decision and Control*, pages 5420–5425, 2010.

[23] PX4-user_guide/gazebo_vehicles.md at main · PX4/PX4-user_guide · GitHub. https://github.com/PX4/PX4-user_guide/blob/main/ko/simulation/gazebo_vehicles.md#typhoon_h480, . Accessed: 2022-12-29.

[24] Pierre-Yves Lajoie, Siyi Hu, Giovanni Beltrame, and Luca Carlone. Modeling perceptual aliasing in slam via discrete–continuous graphical models. *IEEE Robotics and Automation Letters*, pages 1232–1239, 2019.

[25] How automated guided vehicles contribute in industry - sentralog. https://www.sentralog.com/

how-automated-guided-vehicles-contribute-in-industry. Accessed: 2023-01-05.

요약

본 논문에서는 간헐적 외란 및 추정 갱신 하에서도 드론의 안전 비행을 수행하기 위한 velocity field 기반 path following 기법을 제시한다. 특정 시간에 특정 상태를 가져야 하는 trajectory tracking 기법과는 달리, path following 기법은 목표 경로에 수렴 및 추종하는 것을 목표로 한다. 이러한 차이는 바람이나 SLAM의 loop closure 같은 간헐적 외란 및 추정 갱신 환경에서, path following 기법이 더욱 안전한 특성을 보이도록 한다. Velocity field 생성에는 기존 알고리즘에서 계수를 분리하는 이론적 확장을 통해 보다 널리 적용 가능한 알고리즘을 제안한다. 또한, 다양한 상황을 가정한 시뮬레이션 및 실험을 통해 본 논문에서 제안하는 시스템의 안전성 및 활용성 검증을 진행하였다.

주요어: Unmanned aerial vehicle (UAV), Autonomous flight, Path following, Velocity field

학번: *2021-21855*

Acknowledgements

우선 부족한 저를 받아주시고 2년간 지도해 주신 이동준 교수님께 진심으로 감사드립니다. 학생들에게 연구 방향을 잡아주시고 그 방향대로 수행할 수 있도록 학생들의 능력을 이끌어내주시는 모습을 보며 존경하지 않을 수 없었습니다. 또한 항상 학생들의 입장을 생각해 주시며 넘치게 배려해 주셔서 정말 감사하다는 말씀을 드리고 싶습니다.

그리고 2년간 함께했던 연구실 동료들에게도 감사의 말을 전합니다. 특별히 MoSF 그룹에서 같이 고생하며 정말 많은 것을 배울 수 있게 해준 상윤이형, 지석이, 태균이, 민형이, 승욱이에게 감사한다는 말을 전하고 싶습니다. 또한 동기로써 같이 고생했던 창호형, 재휘, 소망이, 영선이, 연구실 적응에 도움을 주신 용석이형, 창우형, 부건이형, 정섭이, 용혁이, 제가 별 도움이 되어 주지 못했지만 같이 즐겁게 생활했던 민성이형, 재우형, 석진이, 석현이, 현수 모두 감사하다는 말씀드리고 싶습니다. 이 외에도 많은 교류는 못했지만 많은 것을 배우고 깨닫게 해준 다른 동료분들에게도 감사드리고, 특히 학생들이 연구에만 집중할 수 있도록 큰 도움을 주신 주남희 선생님에게도 감사의 말씀을 전합니다.

무엇보다도 항상 기도와 사랑으로 대학원 생활과 취업 활동에 힘이 되어 주셨던 외할머니, 어머니, 이모, 외삼촌을 비롯한 가족들께 감사드리고 사랑한다는 말을 전합니다. 헌신적인 사랑과 지지 없이는 석사학위와 취업의 결실을 맺지 못했을 것입니다.

2023년 2월

이승준