AN ABSTRACT OF THE PROJECT OF

Ritika Avadhanula for the degree of Master of Science in Robotics presented on July 26, 2023.

Title: Cooperative Collision Avoidance on Small-sized Quadcopters With Indoor Loco Positioning System.

Abstract approved: _____

Julie A. Adams

Disaster response and surveillance operations will increasingly incorporate Unmanned Aerial Vehicles (UAV)s due to their cost-effectiveness and maneuverability. This trend has driven the research on small-sized quadcopters for varying indoor applications. Small-sized quadcopters' payload and computational capacity limit the usage of complex vision-based collision avoidance algorithms that are employed by larger quadcopters. The existing collision avoidance algorithms for Crazyflies assume near-accurate positioning from a vision-based motion capture system, which has higher acquisition costs and is impractical for large indoor spaces. This challenge is addressed by proposing an uncertainty-aware collision avoidance algorithm capable of handling the positioning uncertainty induced by the Loco Positioning System (LPS), an cost-effective alternative to motion capture. The algorithm aims to enable collision avoidance with dynamically moving peers, enhancing the

multiple quadcopter system's capabilities to achieve their goal with tighter tolerances in congested indoor environments with LPS. This project also presents an improved Crazyflie simulation setup by integrating Crazyflie's controller into a physics simulator. This comprehensive simulation setup enables thorough testing and validation for Crazyflies. Additionally, the distance-based performance metrics are designed with inter-Crazyflie distances between the Crazyflies flying in the shared space. These metrics aim to measure the effectiveness of the algorithms by quantifying the quadcopters' coordination and contribute to the development of multiple quadcopter system evaluations.

Corresponding e-mail address: avadhanr@oregonstate.edu

Cooperative Collision Avoidance on Small-sized Quadcopters With Indoor Loco Positioning System.

by

Ritika Avadhanula

A PROJECT

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented July 26, 2023
Commencement June 2024

ACKNOWLEDGEMENTS

I want to express my deepest gratitude to my advisor, Dr. Julie A. Adams, for her invaluable guidance, patience, and expertise throughout this project. The continuous feedback and encouragement have been instrumental in shaping the direction of this work. I also want to thank Jennifer Leaf and Thomas Snyder who have provided valuable discussions and inputs, enhancing my understanding of small-sized quadcopters along with the quality of this project. Their collaboration and willingness to share their knowledge have greatly contributed to the overall success of the project. I am grateful to Oregon State University for providing the necessary resources and facilities that enable me to carry out this project. Lastly, I want to express my gratitude to my family for their support and understanding throughout my academic journey. Their encouragement and belief in my abilities have been a constant source of motivation.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

UAV	**U**nmanned **A**erial **V**ehicle
LPS	**L**oco **P**ositioning **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
TDoA	**T**ime **D**ifference **o**f **A**rrival
HMTLab	**H**uman **M**achine **T**eaming **Lab**oratory
IMU	**I**nertial **M**easurement **U**nit
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
UWB	**U**ltra **W**ide **B**and
TWR	**T**wo **W**ay **R**anging
MPC	**M**odel **P**redictive **C**ontrol
BVC	**B**uffered **V**oronoi **C**ell
PID	**P**roportional **I**ntegral **D**erivative
SWIG	**S**implified **W**rapper and **I**nterface **G**enerator
NLopt	**N**on **L**inear **Opt**imization
EKF	**E**stimated **K**alman **F**ilter

LIST OF SYMBOLS

n	Number of Crazyflies.
r_{UAV}	Sphere's radius that encloses a Crazyflie completely.
$\mathbf{p}_i$	i^{th} Crazyflie's position vector in the inertial-fixed frame.
$\|\|.\|\|$	Euclidean distance.
ν_{ij}	Voronoi cell boundary between the i^{th} and j^{th} Crazyflies.
$\mathbf{a}_{ij}$	i^{th} and j^{th} Crazyflies' Voronoi cell boundary's vector normal.
b_{ij}	i^{th} and j^{th} Crazyflies' Voronoi cell boundary's offset.
$\mathbf{p}$	Any position vector in the inertial-fixed frame.
$\bar{\nu_{ij}}$	Buffered Voronoi cell boundary between i^{th} and j^{th} Crazyflies.
$\mathbf{g}_i$	i^{th} Crazyflie's goal location.
$\mathbf{p}_{g^*,i}$	Crazyflie's waypoint generated by the collision avoidance algorithm.
acc_{max}	Crazyflie's maximum reachable acceleration.
$\beta_{acc,i}$	i^{th} Crazyflie's acceleration buffer.
X^T	Transpose of the Matrix X.
$\mathbf{v}_i$	i^{th} Crazyflie's velocity.
$\beta_{covar,i}$	i^{th} Crazyflie's uncertainty-aware buffer.
erf(.)	Gaussian error function.

LIST OF SYMBOLS (Continued)

δ	Collision probability threshold.
$G(.)$	Gaussian distribution.
$\hat{\mathbf{p}}_i$	i^{th} Crazyflie's mean position vector.
σ_{ax}	Covariance matrix's variance component corresponding to the axis ax.
$Pr_i(.)$	i^{th} Crazyflie's maximal probability of misclassification.
$\Phi(.)$	Cumulative distribution function.
$\overbrace{\underset{i}{\nu}}$	i^{th} Crazyflie's modified buffered Voronoi cell boundary set.
RVO_{ij}	i^{th} and j^{th} Crazyflies' reciprocal collision cone.
VO_{ij}	i^{th} and j^{th} Crazyflies' collision cone.
$\mathbf{p}_{ij}$	Relative distance between i^{th} and j^{th} Crazyflies.
$\mathbf{D}(\mathbf{q}, r)$	Sphere of radius r and center $\mathbf{q}$.
R_{ij}	i^{th} and j^{th} Crazyflies' combined radius.
Δ	Reciprocal velocity obstacle's time horizon.
$RVO_{ij,tf}$	Set of Crazyflie's reciprocal collision cone velocities transformed to the positions.
FOV_i	The number of peer obstacles in Crazyflie's range.
k_f	Predefined thrust mapping parameter.

LIST OF SYMBOLS (Continued)

k_t Predefined torque mapping parameter.

ω_i Rotational speeds of the i^{th} motor on a Crazyflie.

F_i Thrust generated by the i^{th} Crazyflie's motor.

T_i Torque generated by the i^{th} Crazyflie's motor.

F_{net} Crazyflie's net thrust.

T_{net} Crazyflie's net torque.

Vel_t Crazyflie's velocity at time t.

$RPYrate_t$ The Crazyflie's angular velocity at time t.

pos_t The Crazyflie's position at time t.

RPY_t The Crazyflie's orientation at time t.

$grav$ Crazyflie's weight.

J Crazyflie's moment of inertial.

M Crazyflie's mass.

L Distance between the Crazyflie's center of mass and one of its motors.

dt Simulation's time step.

$\mathbf{k}_{sol}$ Wapyoint generated by the modified BVC approach.

$g(.)$ NLopt's linear constraints.

LIST OF SYMBOLS (Continued)

$h(.)$	NLopt's nonlinear constraints.
$f(.)$	NLopt's objective function.
$h_{grad}(BVC, ax)$	BVC constraints' gradient along the ax axis.
$h_{grad}(RVO, ax)$	Reciprocal velocity obstacle constraints' gradient along the ax axis.
$a_{ij,ax}$	i^{th} and j^{th} Crazyflies' boundary parameter a along the axis ax.
$p_{ax,k}$	kth Crazyflie's position along the axis ax.
$v_{ax,k}$	kth Crazyflie's velocity along the axis ax.
sd_ax	Standard deviation along the axis ax.
$random(sd_{ax})$	Gaussian noise profile with a standard deviation of sd_ax.
$Velin_{ax,new}$	Crazyflie's velocity along the axis ax after noise addition.
$posout_{ax,old}$	Crazyflie's position along the axis ax from the simulation.
$posin_{ax,new}$	Crazyflie's position along the axis ax after noise addition.
r_{lookup}	Crazyflie's maximum range to detect obstacles,
$t_{NavigationMode}$	Navigation mode's time duration required to integrate Crazyswarm API.
$start_{spacing}$	The initial minimum spacing between Crazyflies along the x and y coordinates.
h_{start}	Crazyflies' start altitude in the Navigation mode,

LIST OF SYMBOLS (Continued)

$goal_{spacing}$	The minimum spacing between Crazyflies' goals along the x and y coordinates.
h_{goal}	Crazyflies' goal altitude in the Navigation mode,
$goal_{\sigma}$	Standard deviation of the noise introduced in h_{goal} for altitude randomization.
$velocity_{max}$	Crazyflie's maximum allowable velocity.

Chapter 1: Introduction

Disaster response and surveillance have increasingly incorporated commercial multirotor Unmanned Aerial Vehicles (UAVs) due to their size and maneuverability [1], [2]. The decrease in the acquisition and recurring costs per flying hour has also increased the usage of UAVs during the last decade in civilian settings [3], [4]. The cost-effectiveness and maneuverability of the UAVs have in turn promoted small-sized quadcopter research for varying applications [5]. Multiple quadcopters deployed in these scenarios may require the ability to operate around humans in dynamic and cluttered environments. Safety in these critical environments is achieved with a collision avoidance pipeline that generates motion control commands at a rate faster than the quadcopter's response time. Real-time precision of collision avoidance is influenced by the substantial uncertainty present in the quadcopter state estimates that are generated by positioning systems (e.g., Global Positioning Systems (GPS)). Existing algorithms designed to handle uncertainty have a higher computational complexity which can degrade the quadcopter's real-time performance to avoid collision due to a delay in command generation. This project develops and tests a real-time collision avoidance algorithm to handle uncertainty in state estimation.

GPS is readily used on outdoor UAVs. The GPS signals are unavailable inside the buildings due to non-line-of-sight transmissions limiting their usage in-

doors. Collision avoidance for small-sized quadcopters in indoor environments has motivated the development of alternative positioning systems (e.g., Vision-based positioning or Ultra Wide Band-based (UWB)-based positioning). Vision-based positioning is expensive and difficult to scale. UWB-based indoor Loco Positioning System (LPS) developed by Bitcraze uses a Time Difference of Arrival (TDoA) mode similar to a global positioning system. This system can theoretically connect to an unlimited number of quadcopters at any given instance and can be scaled with the operational area's volume. This indoor positioning system has a degraded signal performance due to multi-path reflections from objects in the operational area. Improving accuracy in indoor positioning with the help of the existing research can increase the computational complexity in navigation command generation. Small-sized quadcopters can have narrow computation power and battery life; hence their ability to run uncertainty-lowering complex algorithms indoors is limited.

Crazyflie UAVs, as illustrated in Fig [1.1], are employed by the Human Machine Teaming Lab's (HMTLab)'s Swarm testbed to perform multiple UAV experiments indoors. Small-sized Crazyflie UAVs with computation constraints need real-time navigation command generation that consumes less power and maintains flight stability. Efficiency in power consumption also allows these UAVs to operate over a wide range of speeds, improving the overall task completion time. Scaling the tasks to multiple UAVs indoors adds dynamic objects in the operating space. Existing UAV collision avoidance techniques fail to handle the positioning uncertainty in the presence of dynamic objects, which hinders application scalability. Existing

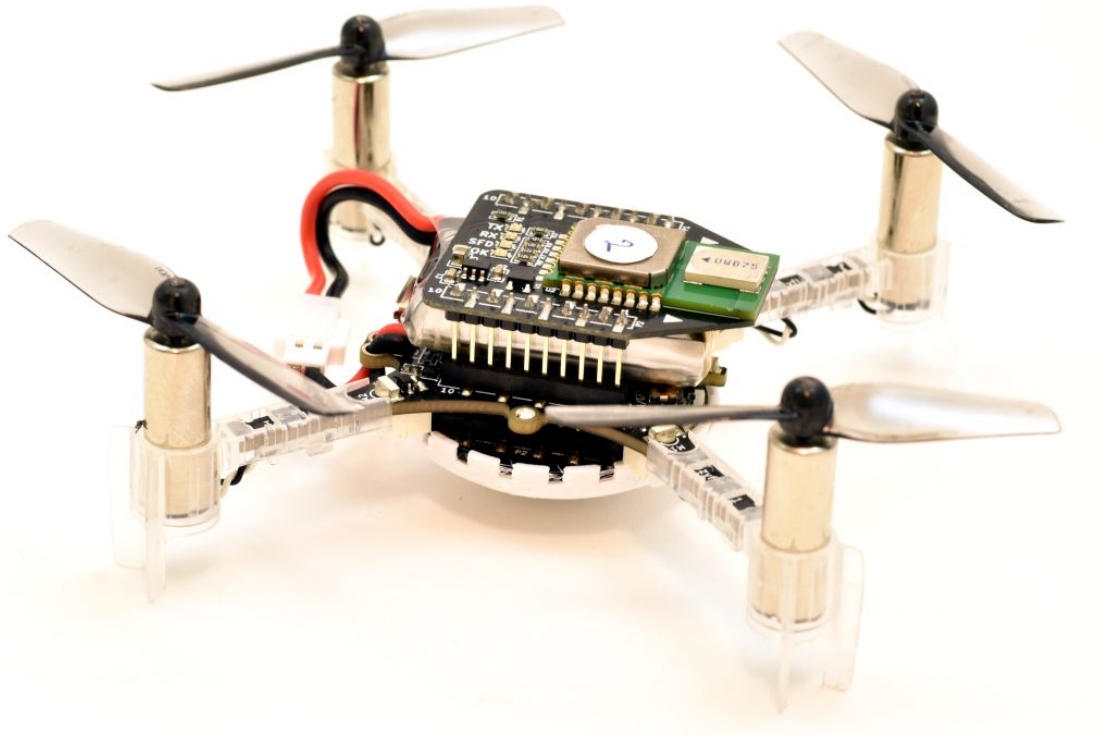


Figure 1.1: Bitcraze's small-sized quadcopter, Crazyflie 2.1 with an LPS receiver [6]. This project employs this UAV to test a use case of the proposed collision avoidance algorithm.

demonstrations with a larger number of UAVs were possible only with expensive vision-based positioning systems that are hard to scale in large spaces. This project aims to develop a real-time collision avoidance algorithm that can scale for a larger group of UAVs by introducing probabilistically collision-free zones around a UAV to accommodate uncertainty with LPS.

Collision avoidance algorithms can be generally classified into two categories, Classical methods (e.g., geometric, graph theory-based, or force-field-based meth-

ods), and Heuristic methods (e..g., optimization-based, or learning-based methods). Existing research states that the geometric methods solved with fast convex optimization have low computational complexity and are well suited for small-sized UAV operations in static and unobstructed spaces. These methods can extract the geometric properties of their surrounding objects to avoid collisions and have a strong theoretical background to analyze collision-free guarantees. Heuristic methods rely on intuition and experience, unlike geometric methods, to generate a solution. These intuition-based do not aim to perform an exhaustive search to generate a global optimum, instead, it finds a trade-off between optimality and real-time performance. This project implements a buffered Voronoi cell method's variant, which introduces linear position and velocity constraints to improve its robustness under substantial positioning uncertainty with LPS, indirectly improving the scalability in domains with dynamic obstacles.

The buffered Voronoi algorithm retracts the Voronoi cell's edge by a safety radius for one UAV, to always keep the UAV's body completely within the Voronoi cell. Crazyflies with limited sensing capabilities benefit from the Voronoi cell design that relies only on the positional information of its peers to generate pairwise boundaries. The Voronoi cell algorithm is modified by introducing positioning uncertainty into each Voronoi cell design, which expands the safety radius for each UAV. Additionally, each peer UAV's velocity is predicted and incorporated into the Voronoi cell design. The motivation for this design modification is to add a supplementary layer of safety for UAVs surrounded by dynamic obstacles.

Prior evaluations of collision avoidance of multiple UAVs focus on traditional

metrics centered around one UAV. These metrics do not completely address the inter-UAV interactions that are essential to measure algorithms' collision avoidance ability. The analysis performed with metrics utilizing these interactions can provide a better understanding of cooperation between UAVs navigating in the environment, thereby addressing both safety and efficiency in collision avoidance. This project devises inter-UAV interaction-based metrics to measure the effectiveness of the algorithms by quantifying the UAVs' coordination.

The project develops a computationally efficient algorithm for small-sized Crazyflie UAVs that also handles uncertainty in positioning measurements for multiple UAV asks. Chapter Two reviews key developments in relevant collision avoidance literature, covering classical and heuristic algorithms. This chapter also covers existing work on various positioning systems used in UAV localization. Chapter Three describes the geometrical collision avoidance used in the project. This chapter integrates uncertainty-aware buffered Voronoi cells with velocity collision cones to achieve real-time response on UAVs. Chapter Four describes the experimental design with the integrated physics simulation platform, introduces the novel metrics computed to quantify the algorithms' collision avoidance ability in multiple UAV scenarios, and presents the associated results. Finally, Chapter Five summarizes the contributions to the field and discusses the future work direction.

Chapter 2: Background

This chapter describes the architecture and functionality of small-sized UAVs used in the HMTLab's Swarm testbed. Furthermore, a brief overview of the existing indoor positioning systems is presented, along with a comparative analysis providing insights into the challenges and effectiveness of these systems. The chapter also delivers a survey on collision avoidance techniques designed for small-sized UAVs. Small-sized UAVs' ability to navigate without encountering collisions is crucial for indoor environments. A common trend in the design of multiple UAV systems is to automate local UAV interactions to achieve user-defined global tasks [7]; however, previous research on multiple UAV collision avoidance ignores the UAVs' second-ordered dynamics and assumes that the obstacles in the surroundings move with a constant velocity. The research on collision avoidance coupled with positioning uncertainty is also limited.

Algorithms factoring in the degree of accuracy in positioning promote flight stability and scalability [8]. There is a pressing need for extensive research on multiple UAV systems in congested environments that incorporate the obstacles' motion and integrate the prominent features of small-sized UAV's dynamics [9]; hence, this chapter addresses this gap and identifies potential solutions for achieving collision-free motion for small-sized UAVs in complex indoor environments.

2.1 Small-sized UAVs

UAVs that weigh less than 250 grams are referred to as small-sized UAVs [10]. The size of these UAVs presents unique challenges for their usage, including limitations on the payload capacity and reconfiguration of the onboard sensors. The cost and size reduction of new-generation sensors (e.g., Inertial measurement unit (IMU), monocular camera) and actuation (e.g., Coreless motors) can make them suitable for the small-sized UAV design. Contrastingly, the miniaturization of these electronic components limits their performance and control efficiency. The electronics size reduction increases the noise and drift in the sensor signal transmission and increases the motor saturation, limiting the small-sized UAV's motor rotation speed [11]. The UAV's size also determines the onboard battery's size, resulting in a limited battery life unsuitable for long-range applications. Size-induced limitations must be considered when choosing a small-sized UAV for a specific application.

2.1.1 Crazyflie 2.X Software Architecture

This project aims to develop a collision avoidance algorithm for Crazyflie 2.X, a small-sized UAV deployed in indoor research and development [12]. Previous studies have explored aspects of Crazyflie's design to understand its capabilities and challenges for indoor applications [13], [14]. The Crazyflie weighs 27 grams and has a recommended payload capacity weight of 15 grams. The Crazyflie is an open-source flying development platform with low latency and long-range radio communication with the host client. Expansion decks with varying capabilities

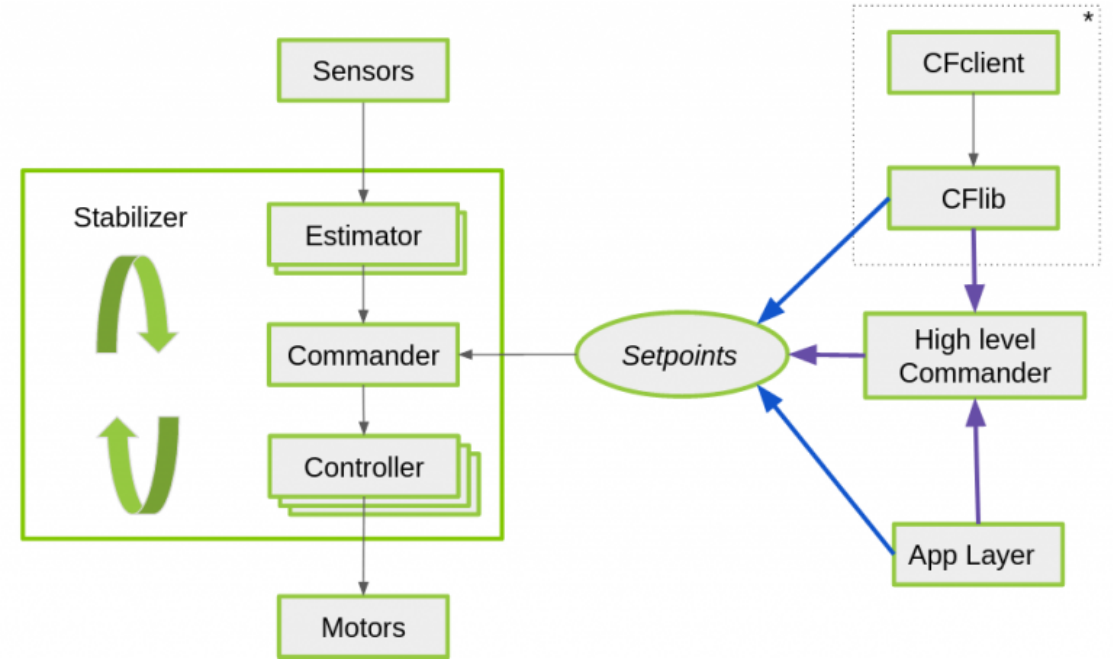


Figure 2.1: Crazyflie2.X platform's modular autonomy architecture keeps the UAV stable throughout the motion. This UAV's open-source design simplifies the collision avoidance algorithm's integration with the onboard subsystems [6].

(e.g., LPS, Flowdeck sensor, Multiranging sensor, Charging pad) can be mounted on this small-sized UAV to develop autonomous features for research, education, and teaching [15]. Roboticists have also extended this open-source platform to multiple UAV support; however, the radio bandwidth places an upper limit on the number of simultaneous reliable UAV connections to the host client. Decentralized communication networks (e.g., Flying Ad-Hoc Network [16]), interference reduction algorithms (e.g., Adaptive Channel selection [17]) address the challenges in communication for multiple UAV applications [18]. Radio communication can have complications like packet collisions, non-line-of-sight reflection, and radio in-

terference with other sources emitting data at similar frequencies [19].

Figure 2.1 depicts the Crazyflie 2.X UAV's software architecture [6]. This software architecture allows the user or a local navigation planner to send position waypoints to the low-level motion commander. The low-level commander receives additional real-time state estimates from the onboard Estimated Kalman Filter (EKF) to generate motor control commands and fly with stability.

2.1.2 Inertial Measurement Unit

IMUs predict the UAV's six degrees of freedom pose with the help of a magnetometer, accelerometer, and gyroscope. This inertial sensor is susceptible to vibration, which induces errors in the rotational velocity measured by the gyroscopes and the acceleration measured by the accelerometer. These error-prone measurements are later integrated to predict the pose. An extended period of operation with IMUs alone can result in a significant drift from the ground truth; hence, researchers utilize other sensor data to better estimate the UAV's location in a given functional space by compensating for the accumulated error [11]. UAVs use multiple sensor-fused state estimates to maintain stability during their flight [11], [20]–[22].

2.1.3 UAV Localization

Kalman filters are deterministic mathematical state estimators that work efficiently on Gaussian state variables. The UAV sensors are assumed to generate data with

a Gaussian profile to integrate with the Kalman filter-based localization. This filtering technique performs a multi-sensor fusion (e.g., IMU, flow deck sensor, LPS, multi-ranger sensor) to minimize the covariance in the estimated state [22]. These filters are recursive state estimators that use the last best state estimate to the dynamic motion model and sensor models to generate a weighted average. The aerospace industry has successfully implemented Kalman filter's variants for the guidance, navigation, and control protocols. Localization and mapping techniques with these Kalman filters were later adapted to mobile robots [23]. The precise state estimate from localization is essential for UAV navigation in cluttered environments. The EKF on the Crazyflie UAV generates state estimates with a standard deviation of 15 centimeters [19]. UAVs can be kept safe in cluttered uncertain environments by considering the positioning variance while devising a motion plan.

2.1.4 Collision Detection

Detection of potential collisions protects the UAV hardware and its payload from damage. Detection also enhances the UAV's autonomous capabilities to minimize human intervention while performing tasks. Limited computational power, battery life, and payload capacity make visual sensors on the Crazyflie UAVs an ineffective option for collision detection. Recent studies have investigated alternative sensing technologies for small-sized UAVs that can replace the requirement for UAV-UAV observations. An evaluation of small-sized UAVs proposes the use of lightweight,

protective cages for collision-based navigation instead of onboard collision detection sensors [24], [25]. Researchers suggest that the small-sized UAV configuration does not support accurate trajectory design due to the presence of noise induced by miniaturized localization and sensing. Furthermore, collisions at the small-sized UAV's scale have a low penalty; hence UAVs with protective cages and near accurate positioning from motion capture systems can move toward the goal positions via collision-based navigation at low velocities [24], [26]. Small-sized UAVs deployed for surveillance in safety-critical areas (e.g., sensing a gas leak inside a chemical plant [27]) cannot rely on collisions for navigation as described by Ackerman [24]. Collision-based navigation can result in contamination or spillage in indoor facilities, disruption in ongoing activities, and even result in life-threatening damage. This collision-based navigation relies heavily on accurate collision models and robust controllers that can handle recovery smoothly [26]. Additionally, this collision-based navigation design can result in string instability, a prevalent issue of noise amplification within a group of closely moving multiple Vehicle systems, which can degrade the overall mission performance [28].

Time division multiple access-based radio communication is another collision detection approach designed for small-sized UAVs that promotes vision-less collision detection. This communication-based method allows the exchange of each UAV's location and velocity [29], [30]. The Crazyflie's radio transmitters and receivers can exchange information among its peers and the ground station [6], which can be used for collision detection. This communication protocol is an alternative detection approach that supports extended battery life for a longer mission and

enhances Crazyflie's autonomy by addressing the challenges posed by its small size [31].

2.2 Positioning Systems

Positioning systems are essential to enhancing the autonomous capabilities of UAVs. These systems provide essential spatial information that can be used in UAV collision avoidance, trajectory design, and control. The strengths and weaknesses of various positioning systems employed on UAVs are presented in this section. The Crazyflies employed by the HMTLab's Swarm testbed rely on the LPS for positioning.

2.2.1 Global Positioning System

UAVs operating in outdoor environments have onboard sensors that include a GPS, a camera, and an IMU for navigation. A GPS updates position up to 2 meters with precision 95% of the time as signals are received from the transmitting satellites [32]. Distance triangulation between a receiver and at least three satellites generates accurate state estimates with the provided data [33]. The attenuation of signals inside the building and in non-line-of-sight locations degrades the accuracy and availability of GPS signals; hence, UAVs with GPS cannot function stably for indoor applications.

2.2.2 Ultrasonic Sensors

Ultrasonic acoustic sensors are another class of sensors employed in certain domains for UAV localization. These low-cost and lightweight sensors are attractive for indoor UAV research [34] but can be impacted by room humidity and temperature. Ultrasonic sensors suffer from multi-path and non-line-of-sight reflections, degrading the state estimation quality when localizing multiple small-sized UAVs indoors [35].

2.2.3 Vision-based Positioning Systems

A recent technological survey states that a sensor fusion of onboard IMU and vision-based positioning systems (e.g., motion capture positioning, flow-deck-based positioning, stereo camera-based positioning) dominates 50% of the ongoing indoor UAV research [36]. Cameras reobserve features in a given space and improve the state estimation quality [37]. Simultaneous Localization And Mapping (SLAM) or visual odometry heavily relies on cameras (e.g., flow deck-based positioning, stereo camera-based positioning) integrated with IMUs. Visual SLAM builds the world's maps world and estimates the UAV's pose by matching the camera's observed features with the updated map [38]. Visual odometry estimates the pose by scanning the image deviation captured by the onboard cameras [39]. These optical and laser-based camera processes have a low sampling rate and a higher processing time for feature extraction from the feed and are unsuitable for small-sized UAVs [22].

Additionally, visual positioning systems require good lighting conditions for accurate functioning indoors. These systems are also affected by shadows, reflections, and glare present in the sensor's field of view. An evaluation by Palivdis et al. [40] suggests a combination of visible light and infrared cameras can extend the usage of UAVs in low-light conditions; however, the higher payload and memory consumption of these sensors make them unsuitable for the small-sized UAV.

2.2.4 Motion Capture-based Positioning System

Motion capture technology can track indoor UAVs and other objects with passive markers or reflective indicators with the help of cameras placed around a functional space [27], [41]–[45]. Small-sized UAV researchers in indoor facilities readily use this motion capture technology. The cameras employed for these applications support frame capture at 2000 hertz and readings with sub-millimeter accuracy [46]. UAVs working with this positioning system are free from onboard computational latency; however, the image's post-processing on the central host is a limitation to localization calculations [47], [48]. Despite the benefits, limited range communication with the primary host to receive localization estimates, expensive setup charges, and difficulty scaling in large indoor spaces limit the usability of motion capture technology in multiple UAV applications.

2.2.5 Loco Positioning System

Another positioning technology employed on small-sized UAVs is the UWB-based positioning system, which can provide decimeter accuracy when combined with an IMU [19], [36]. Bitcraze's LPS uses UWB radio signals to communicate between miniature beacons or transmitters around the operating space area and receivers attached to the UAVs [19]. The LPS can independently connect to multiple UAVs in a given space, where each UAV has a UWB receiver on it. LPS technology with this feature stands out as a promising positioning system for multiple UAV operations.

UWB devices have a small, lightweight, low-power-consuming design that depends on long-range radio communication and is suitable for indoor applications [49]. The low-cost LPS is independent of the environment's lighting [35]. LPS technology has two sources of error: ranging and systematic errors. Placing UWB transmitters or anchors around the operation space results in undesirable non-line-of-sight signal reflections and attenuation, yielding systematic error [49], [50]. Motion capture technology has higher accuracy than UWB technology, but its acquisition and scaling costs are fifteen times higher, limiting the usage of motion capture technology in large indoor spaces [6], [51]. Indoor small-sized UAVs are limited to lower velocities; hence, EKF can sufficiently reduce estimation covariance and maintain decimeter-level accuracy indoors [19]. Cost-effectiveness and ease of scalability are the main reasons for the recent surge in the usage of UWB.

The positioning system can communicate unidirectionally with the receivers in

three modes: Two Way Ranging (TWR), TDoA2, and TDoA3. The TWR mode calculates the UWB radio signal's time of flight from transmitter to receiver and then computes the distance between the two devices by multiplying the time by the speed of light [52]. The two TDoA methods use the UWB radio signal's time difference of arrival at two or more transmitting anchors to calculate the receiver's location. TDoA2 mode utilizes scheduled radio signal transmission, whereas TDoA3 has a randomized signal transmission [19]. Manipulating the arrival time can affect the distance estimate, making the system vulnerable to cyber-attacks [53]; however, these security concerns are out of this project's scope.

Researchers use a Cramer-Rao lower-bound analysis to identify flyable regions surrounded by UWB transmitters for UAV localization which determines the convex hull enclosed by the UWB transmitters as the flyable region with the highest precision and bifurcation envelopes. The LPS is expected to perform with minimal positioning uncertainty within the convex hull [54]. Researchers have derived a six UWB transmitter configuration from the Cramer-Rao analysis, shown in Figure 2.2, that provides an ideal flying condition on the experimental testbed. This project adopts the six transmitter design to investigate the navigation performance in LPS-based operations.

A notable approach in Crazyflie research is to fuse sensory reading from LPS and vision-based systems (e.g., visual odometry [55], optical flow-based system [56]). This configuration of sensor fusion introduces a camera element on the UAV, which acts as an extra payload. The drawbacks of this camera-based LPS configuration outweigh its benefits, influencing indoor small-sized UAV researchers to de-

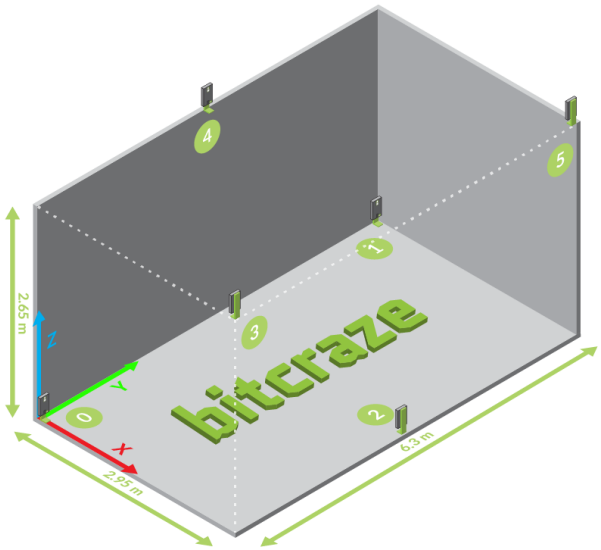


Figure 2.2: UWB Anchor placement around a custom platform designed to keep the LPS estimates stable throughout the UAV's motion within the testbed [19].

velop new sensing and navigation solutions resilient to decimeter-level uncertainty in state estimation. This project also develops a collision avoidance algorithm for autonomous navigation that can handle localization uncertainty in cluttered environments.

2.3 Collision Avoidance

Multiple indoor small-sized UAVs can boost productivity in warehouse surveillance and monitor spaces hazardous to human beings. The entertainment industry, agriculture, and commercial surveillance have also seen increased interest in multiple small-sized UAVs. These UAV applications necessitate the development of robust

collision avoidance to provide safety. UAVs deployed in the current multiple agent applications move along predefined paths. These UAVs are continuously monitored by human operators using joysticks; however, controlling a group of UAVs simultaneously by identifying and assigning motion commands is difficult in real-time [57].

UAV Navigation tasks encompass activities that include, (1) sensing the environment, (2) mapping the environment, (3) localization in the environment, (4) collision avoidance, and (5) actuator control. Achieving autonomy in navigation is the ability to reach a goal location reliably with partial knowledge of the environment accumulated with uncertain sensor readings. Autonomy to navigate around obstacles, including other moving UAVs, becomes an essential addition to the UAV software architecture for success in diverse missions [58].

Prior studies have focused on a well-defined modularize software architecture for navigation autonomy to localize each feature [59]–[62]. These software architectures help integrate processes with varying real-time and non-real-time demands and define the contribution of each control layer.

Without planning, the control effort will not guide the overall UAV behavior to reach a distant goal location within a time frame [11]. Global navigation algorithms work well on UAVs with complete information regarding their obstacles throughout the motion; however, these navigation algorithms fail to work in multiple UAV operations that require guaranteed collision avoidance. The need for local navigation is highlighted when local environment information alone is available through sensors in multiple UAV scenarios [58]. These local navigation

algorithms feature iterative trajectory design over a short time horizon to react quickly to local adversities [11].

Real-time navigation for multiple small-sized UAVs is limited by computation power, and overall mission time and needs to operate in continuous space with discrete control. Navigation laws are simplified by assumptions made on the UAV motion model, obstacle boundary, uncertainty in state estimation, and actuation [42]. Rather than redesigning an indoor space and making it structured and simplistic for multiple UAV navigation, it is practical to devise a navigation stack with collision avoidance capabilities that allow task accomplishment without continuous human intervention. This section aims to highlight challenges and possible opportunities for improvement in the existing research on collision avoidance for small-sized UAVs deployed in multiple agent applications indoors.

2.3.1 Bug Algorithms

Bug algorithms represent primitive, computationally efficient collision avoidance algorithms that assume local knowledge of the environment, follow the walls of sensed obstacles, and maintain a straight path to reach the goal location [11]. Early collision avoidance for surveillance relied on these algorithms for robot safety and navigation [63]–[65]. The maximum sensor ranges and the wall following behavior (e.g., Turning right around an obstruction) dictate the performance of these collision avoidance algorithms. UAVs following bug algorithms have advantages over other algorithms because UAVs do not get stuck in local minima (e.g., UAVs

following potential field collision avoidance) and have non-oscillating trajectories towards the goal [66].

Traditional bug algorithms thrive in static worlds with one UAV moving in the 2-dimensional environment ignoring the UAV's propeller thrust-based control; hence, direct implementation of bug algorithms 3-dimensional space in dynamic multiple UAV applications is not reliable [67]. UAVs with traditional bug algorithms can result in longer mission times when confronted with obstacles of different shapes and sizes [68]. Researchers have also developed hybrid methods integrating other advanced techniques with bug algorithms to optimize UAV collision avoidance during 3-dimensional navigation [66], [68], [69]. These hybrid algorithms heavily rely on visual and range sensors; hence, small-sized UAVs with low computational capabilities for sensor processing cannot rely on these hybrid bug algorithms.

2.3.2 Artificial Potential Field Algorithms

Artificial potential field methods represent another set of actively researched real-time collision avoidance algorithms. These algorithms employ virtual potential fields around obstacles to navigate the environment. This class of algorithms fundamentally designs an attractive virtual potential field around desired goal locations and a repulsive virtual potential field around obstacles when navigating [11]. The force obtained from the net potential field generates a control command to move the UAV safely.

Potential field methods were initially developed for robot manipulators for collision-free trajectory design [70], [71] and were later extended to mobile robotics domains [68], [72]–[75], including UAVs in static and dynamic environments [58], [76], [77]. Despite their extensive use in robotics, potential fields have certain limitations. The biggest drawback of this category of collision avoidance methods is that the generated trajectory has significant oscillations and a tendency to get stuck in local minima [78], [79]. Researchers have explored modifying this limitation by varying the geometric constraints in the potential field design; however, these methods still remain sensitive to increased clutter around the UAV and sensitive towards disturbances in actuation and sensing [58], [77]. Other variants reduce these methods' shortcomings by improving upon repulsive potential field design [73], [76], [80], [81] and the safety distance threshold [82]. Researchers still assume UAVs' perfect localization in their experiments which leads to less resilient UAV collision avoidance with limited scalability.

2.3.3 Learning Based Algorithms

The lightweight and cost-effectiveness of small-sized UAVs have motivated researchers to explore alternatives that are not downscaled versions of their larger counterparts [41]. Prior studies have shown that obstacle avoidance on indoor UAVs commonly relies on vision-based sensing integrated with artificial intelligence to generate control commands [83]–[85]. The learning-based approaches benefit from their decentralized design but suffer from the computational burden of using

vision-based sensors [41], [86]. Additionally, retraining the onboard network to accommodate the variations in the operational spaces also limits the learning-based methods' generalizability on UAVs [84], [87], [88]. Prior studies with learning-based approaches are model-based implementations with single integrator dynamics, limiting collision avoidance ability for congested airspaces for a large group of UAVs [89]–[92]. Recent advances in the physics simulation engines for small-sized UAVs have allowed researchers to explore learning-based methods for decentralized collision avoidance[93]; however, motion capture positioning technology for localization with near accurate estimation still poses as a drawback.

2.3.4 Model Predictive Control

Researchers have also integrated Model Predictive Control (MPC) frameworks with existing collision avoidance algorithms to aid safety for multiple UAV applications [58]. Linear MPC algorithms are tightly coupled with the sensor measurements and fail to show stability and robustness under positioning uncertainty [94], [95]. Researchers have also expanded the MPC model design to accommodate complex UAV's second-order dynamics to improve the collision-free trajectory design for UAVs [27], [96]; however, additional constraints are to address deadlock conditions that may arise in congested environments to ensure efficient navigation in multiple UAV applications [97], [98].

2.3.5 Velocity Obstacle-based Methods

Dynamic obstacles in an environment exhibit varying velocities over time, which necessitates collision avoidance algorithms that can adapt to these velocity changes. Velocity obstacle methods are a class of collision avoidance methods that assume the velocity of each obstacle in the environment is always known during the trajectory generation [99]–[103]. A range of velocities resulting in a collision is computed to generate a desirable UAV collision-free waypoint. The preliminary velocity obstacle method has been extended to omnidirectional UAVs for collision avoidance but fails to scale in dense environments. This method cannot handle uncertainty in obstacles' trajectory within a desired computational time; hence, alternative approaches are proposed to extend this method in dense environments with multiple UAVs [100]–[104]. Communication loss, excessive oscillations in the output trajectory, uncertainty in positioning, and computational effort to detect the obstacles' velocities are also common issues limiting the algorithm's scalability.

Reciprocal velocity obstacles are an extended version of the velocity obstacle methods that assume agents in the environment share an equal responsibility during collision avoidance maneuvers [102]. This concept can promote smoother trajectory design in congested environments, where each individual contributes to collision-free motion and the overall distance between each UAV pair is relatively shorter than the predecessor method. Extensive research has been focused to devise an optimal reciprocal velocity method for UAVs in 3-dimensional space; however, the positioning uncertainty in the trajectory computation is overlooked

in its design. The resulting trajectory is also susceptible to oscillations [105], [106]. Previous research on small-sized UAVs utilizes simulation to demonstrate that existing velocity obstacle methods generate large turning angles while avoiding dynamic obstacles [107]. The flexible design of velocity obstacle approaches helps enhance flight performance by constraining the UAV's angular deviation [107]. Additionally, collision-free trajectory design for a UAV to handle obstacle trajectory uncertainty, as well as its positioning uncertainty, has necessitated the need for improvements in the reciprocal velocity obstacle method [67].

The reciprocal velocity obstacle algorithm's simplicity has also motivated the researchers to extend it to a decentralized collision avoidance domain. This extension has been achieved by combining velocity obstacle methods with Buffered Voronoi Cell (BVC), which provides deterministic collision avoidance guarantees [108]. This velocity obstacle algorithm's variant works in complex environments with tens of agents and has a similar runtime performance as the optimal reciprocal velocity obstacle. This algorithm was tested only on experiments designed for a two-dimensional plane and ignored the uncertainty in the state estimation. This assumption of near-accurate state estimation limits the research's extension to real-world small-sized UAVs with uncertain positioning. Further modifications are suggested in this project that utilizes a combination of BVC and reciprocal velocity obstacle methods to test the algorithm's scalability. The experiments emulate uncertainty in positioning for each agent to better understand the capabilities of small-sized UAVs in the real world.

2.3.6 Voronoi-based Methods

BVC was initially proposed as a decentralized collision avoidance method on multiple UAVs by defining collision-free Voronoi cells around convex obstacles [45]. This Voronoi cell method modifies UAV's global plan by allowing modifications to the local path around known convex obstacles. The BVC method does not require velocity information; hence, it is a superior choice for collision avoidance over velocity obstacle methods [109]. The researchers have developed variants of BVC that account for the sensor-induced uncertainties in positioning to compute navigation commands for the UAVs in 3-dimensional space [45], [110], [111]. These variants modify the buffer boundary around each UAV to improve collision avoidance accuracy. Subsequently, an MPC is employed to generate control commands that respect UAV omnidirectional and BVC constraints. The increase in the number of obstacles around a given UAV increases the number of BVC constraints used to construct safe boundaries around the UAV. This growth in the number of constraints can impact the MPC controller's runtime impacting the UAV's response time against collisions in a congested environment [112].

Flight stability can be ensured by simplifying the trajectory generation process on the UAV and employing reactive collision avoidance methods to a simplified controller (e.g., a decoupled attitude and position's Proportional-Integral-Derivative (PID) controller) for small-sized UAVs. This project relies on Crazyflie's PID controller and enhances its collision-free trajectory design by modifying the BVC constraints developed in the prior research.

2.4 Summary

UAVs are versatile, omnidirectional systems governed by second-order differential dynamics. The UAV's control is based on acceleration generated by its motor thrust. UAVs have physical limitations on their acceleration and velocity, which impose time constraints on reaching a particular orientation or position in a specified timeframe [11], [113]. These UAV motion constraints must be accounted for via a local navigation method that generates collision-free paths.

Additionally, the collision avoidance algorithm for location navigation must factor in the obstacles' dynamic movements, while also accounting for the overall uncertainty associated with the sensors. This project's implemented technique is motivated by an uncertainty-aware BVC [110] and a reciprocal velocity obstacle [108] approach. This method accounts for the three constraints including, sensor uncertainty, UAV's second-ordered dynamics, and the obstacles motion in order to improve the safety of multiple UAV applications employed in congested spaces.

Chapter 3: Systems Design

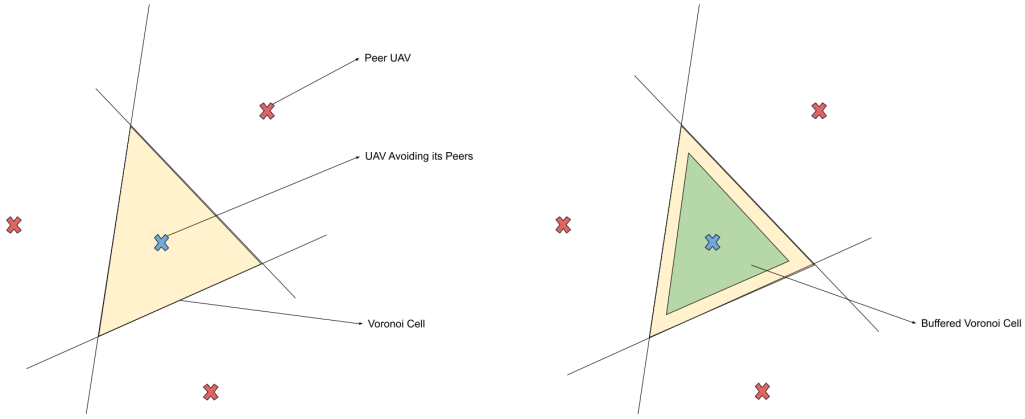
This chapter delves into the implementation of the modified BVC approach for small-sized UAVs (i.e., Crazyflie 2.X), specifically in congested indoor spaces that incorporate the LPS. A brief presentation of the baseline BVC approach originally developed for small groups of Crazyflie UAVs operating with a motion capture positioning system is provided. The LPS induces substantial uncertainty in the Crazyflies' positioning and affects the accuracy of obstacle position detection. This use of LPS with Crazyflies necessitates modifying the baseline to account for the degraded positioning measurements. The baseline algorithm ensures Crazyflie's safety in sparse environments, but its scalability was not tested on Crazyflies; hence, a combination of uncertainty-aware BVC with reciprocal velocity obstacle is employed to account for the uncertainty in both the Crazyflie and its peers' position measurements. The trajectory design is enhanced for congested environments where collision avoidance becomes a crucial autonomous capability for small-sized UAVs.

The baseline BVC approach's research was developed on the Crazyswarm platform that does not account for the Crazyflies' second-ordered dynamics. This Crazyswarm platform bypasses the LPS, controller, and estimation submodules of the Crazyflie and only integrates the planning modules in simulation; hence, this project integrates the controller submodules developed by the gym-pybullet-drones

simulator into the Crazyswarm platform. This integration of platforms enables a comprehensive testing platform. The design decisions for Crazyflie's localization, controller, and motion model are also provided.

3.1 Buffered Voronoi Cell Approach

The baseline BVC approach is a deterministic collision avoidance algorithm with low computational complexity. Crazyflies employing this algorithm autonomously navigate by following waypoints directed toward their goals while adeptly avoiding collisions with other Crazyflies and convex obstacles. This baseline approach constrains the Crazyflies to move to a waypoint within their dynamically changing Voronoi cells over time. This approach is superior to the potential field method discussed in Chapter 2 because of its reduced sensitivity toward parameter tuning.



(a) Voronoi cell (b) Buffered Voronoi cell

Figure 3.1: The highlighted regions correspond to the (a) Voronoi cell and (b) buffered Voronoi cell around the Crazyflie (blue) for trajectory generation to enable safety against its peer obstacles (red).

Voronoi cell boundaries are the planar tessellations around n point-sized objects present in the space, subdividing the space into exactly n cells. Each Voronoi cell encloses a region closest to its respective point-sized object. Crazyflies are not point-sized objects; hence, it is necessary to consider their physical volume while computing non-colliding Voronoi cells for trajectory design. Crazyflies are assumed to be completely enclosed inside a sphere of radius r_{UAV} in order to simplify the cell boundary calculations. The BVC, unlike Voronoi cells, considers the Crazyflie's size to modify the simple Voronoi cells into a smaller buffered cell proportional to their size. Figure 3.1 visually compares a simple Voronoi cell to its buffered counterpart, illustrating the impact of considering the Crazyflie's size in the baseline BVC approach. The highlighted cell in Figure 3.1a represents a simple Voronoi cell, while the shrunken green cell in Figure 3.1b is the buffered cell generated by retracting the sum of the radii of both the Crazyflie and its associated obstacle along each cell segment. This smaller green cell constrains the Crazyflie to a region where no component of the Crazyflie can collide with its peer obstacles' components.

Each Crazyflie is located at a position $\mathbf{p}_i$ where $i \in [1, n]$. Crazyflies i and j are in the collision-free configuration when $||\mathbf{p}_i - \mathbf{p}_j|| > 2r_{UAV}$ where $||.||$ is the Euclidean distance between the position vectors $\mathbf{p}_i$ and $\mathbf{p}_j$. The i^{th} Crazyflie's Voronoi cell boundary segment ν_{ij} with respect to its peer j^{th} Crazyflie is represented by a three-dimensional hyperplane:

$$\nu_{ij} = \left\{ \mathbf{a}_{ij}^T \mathbf{p} - b_{ij} \leq 0, \forall j \neq i \right\}. \tag{3.1}$$

The hyperplane separating the i^{th} and j^{th} Crazyflies has $\mathbf{a}_{ij} \in \mathbb{R}^3$ as its normal vector, b_{ij} as its offset, where $\mathbf{p}$ is any position vector lying inside the Voronoi cell. The baseline BVC's boundary $\bar{\nu_{ij}}$ factoring in the i^{th} and j^{th} Crazyflies' sizes is:

$$\bar{\nu}_{ij} = \left\{ \mathbf{a}_{ij}^T \mathbf{p} - b_{ij} + r_{UAV} ||\mathbf{a}_{ij}|| \leq 0, \forall j \neq i \right\}. \tag{3.2}$$

3.1.1 Goal Directed Waypoint

Each Crazyflie in the system is assigned a specific task of reaching its goal location $\mathbf{g}_i$ from its current location $\mathbf{p}_i$. The Crazyflie's flyable region is constrained to a convex BVC enclosed within separating hyperplanes $\bar{\nu}_{ij}$ to ensure safe navigation. An optimal waypoint $\mathbf{p}_{g^*,i}$ within the BVC minimizing the angular deviation from the goal $\mathbf{g}_i$ is generated by solving the nonlinear objective function in Equation 3.3 satisfying the constraints presented in Equation 3.2.

$$\mathbf{p}_{g^*,i} = \underset{\mathbf{p} \in \bar{\nu_i}}{\operatorname{argmin}} \, arccos\left(\frac{\mathbf{p}(\mathbf{p}_i - \mathbf{g}_i)}{||\mathbf{p}||||\mathbf{p}_i - \mathbf{g}_i||}\right). \tag{3.3}$$

3.1.2 Braking Awareness

Crazyflies are thrust-controlled small-sized UAVs with physical limitations on their acceleration that impose time constraints on their ability to change position and velocity. Equation 3.3 works under the assumption that the UAVs are velocity-controlled with position constraints that are enough to keep the Crazyflie within its BVC. This Equation 3.3 guides each Crazyflie toward their BVC boundary with

a non-zero velocity at each time instant [45]. Physical limitations to accelerate, denoted by acc_{max}, within a given time frame are essential to keep the Crazyflies safe. There arises a challenge to decelerate within a given time horizon to stop on the boundary and avoid overshooting into its obstacle's cell.

Braking awareness is instilled into the BVC boundary design to factor in the physical limitations of the Crazyflie to avoid collisions. i^{th} Crazyflie's acceleration buffer $\beta_{acc,i}$ is defined in Equation 3.4 representing this braking awareness that heuristically decelerates the Crazyflie before the cell's boundary is reached. $\mathbf{a}_{ij}^T$ is the transpose of the BVC's hyperparameter $\mathbf{a}_{ij}$ corresponding to the i^{th} and the j^{th} Crazyflies' boundary segment's normal vector. The cell's size is modified proportionally to the Crazyflie's current velocity $\mathbf{v}_i$.

$$\beta_{acc,i} = \begin{cases} \frac{||\mathbf{a}_{ij}^T \mathbf{v}_i||^2}{acc_{max}}, & ||\mathbf{a}_{ij}^T \mathbf{v}_i|| > 0 \\ 0, & otherwise \end{cases} \tag{3.4}$$

3.1.3 Uncertainty Awareness

Uncertainty in positioning estimates also degrades Crazyflie's ability to stay within a well-defined boundary. Crazyflie's planning and control subsystems rely on the onboard EKF, which receives the state information from the sensors, LPS, and IMU. These sensors induce substantial signal uncertainty into their output. The EKF generates a weighted mean of the predicted state and the sensor outputs to result in a noisy positioning estimate that converges to the ground truth over time. This positioning estimate is represented as a Gaussian distribution with mean and

standard deviation for each component of the 3-dimensional space.

The traditional BVC approach does not consider the positioning estimate's Gaussian nature; hence, an additional safety buffer $\beta_{covar,i}$ is provided in Equation 3.5 to modify the BVC. This uncertainty-aware buffer utilizes the inverse of the Gaussian error function **erf(.)** to modify the cell's boundaries. Crazyflies incorporating this additional safety buffer into their BVC design are expected to maintain a probability of collision below the threshold $1 - \delta$, where δ is represented by the probability that the Crazyflie is safe [45]. The cell's size is modified proportionally to the state covariance matrix σ.

$$\beta_{covar,i} = \sqrt{2\mathbf{a}_{ij}^T \sigma \mathbf{a}_{ij}} \mathbf{erf}^{-1}(2\sqrt{1-\delta} - 1). \tag{3.5}$$

3.1.3.1 Hyperplane Parameters

The i^{th} Crazyflie's position is denoted as p_i, represented as a Gaussian $G(\hat{\mathbf{p}}_i, \sigma_i), i \in [1, n]$, where $\hat{\mathbf{p}}_i$ represents the mean position and σ_i represents the covariance of the corresponding position. Apart from the uncertainty-aware safety buffer, the hyperplane parameters representing the BVC boundaries must factor in the Crazyflies' uncertain positioning. These hyperplane parameters include the normal vector $\mathbf{a}_{ij}$, and the offset b_{ij}, which are estimated by minimizing the maximal

probability of misclassification $Pr_i(.)$ of the i^{th} Crazyflie [45], presented as:

$$\begin{aligned} Pr_i(\mathbf{a}_{ij}^T\mathbf{p} > b_{ij}) &= Pr_i(\frac{\mathbf{a}_{ij}^T\mathbf{p} - \mathbf{a}_{ij}^T\hat{\mathbf{p}}_i}{\sqrt{\mathbf{a}_{ij}^T\sum_i \mathbf{a}_{ij}}} > \frac{b_{ij} - \mathbf{a}_{ij}^T\hat{\mathbf{p}}_i}{\sqrt{\mathbf{a}_{ij}^T\sum_i \mathbf{a}_{ij}}}) \\ &= 1 - \Phi(\frac{b_{ij} - \mathbf{a}_{ij}^T\hat{\mathbf{p}}_i}{\sqrt{\mathbf{a}_{ij}^T\sum_i \mathbf{a}_{ij}}}), \end{aligned}$$

where $\Phi(.)$ is the cumulative distribution function implemented to find the BVC boundaries' hyperplane parameters and $\mathbf{p}$ is any position vector in the 3-dimensional space. The maximal probability in Equation 3.6 for the i^{th} and the j^{th} Crazyflies are later utilized to solve the min-max problem:

$$(\mathbf{a}_{ij}, b_{ij}) = \underset{\mathbf{a}_{ij}\in R^3, b_{ij}\in R}{\operatorname{argmin}} (Pr_i, Pr_j). \tag{3.6}$$

The resulting hyperplane boundaries' parameters coincide with the parameters in Equation 3.1 when the standard deviation σ_{ax} of each position component along each axis ax is identical. Incorporating this assumption results in $\mathbf{a}_{ij} = \frac{2}{\sigma_{ax}}(\hat{\mathbf{p}}_j - \hat{\mathbf{p}}_i)$ and $b_{ij} = \frac{1}{\sigma_{ax}}(\hat{\mathbf{p}}_j - \hat{\mathbf{p}}_i)^T(\hat{\mathbf{p}}_j + \hat{\mathbf{p}}_i)$, where $\sigma_i = \sigma_j = \sigma_{ax}^2 I$ and $\hat{\mathbf{p}}_i$ is the state estimate's mean of the i^{th} Crazyflie and σ_i is the covariance matrix of the i^{th} Crazyflie's EKF output. Furthermore $\overbrace{\underset{i}{\nu}}$ in Equation 3.7 represents the i^{th} Crazyflie's BVC constraints that include both acceleration and uncertainty awareness safety buffers. These constraints are used in Equation 3.3 to generate a position waypoint with a probability of collision below $1 - \delta$.

$$\overbrace{\underset{i}{\nu}} = \left\{ \mathbf{a}_{ij}^T \mathbf{p} - b_{ij} + r_{UAV} ||\mathbf{a}_{ij}|| - \beta_{covar,i} - \beta_{acc,i} \leq 0, \forall j \neq i \right\}. \tag{3.7}$$

3.2 Velocity Obstacle Approach

BVC's guarantees of collision avoidance are limited to single-order systems; however, this method is still used to generate collision-free paths for second-ordered UAVs. A limited collision-free guarantee can lead to unnecessary collisions around other moving Crazyflies in congested scenarios. This necessitates the need for a method that can design trajectories with improved guarantees of collision avoidance. The velocity obstacle method determines the Crazyflie's potential velocities that may result in a collision within a given time horizon; hence, they are desirable to improve the system's collision-free guarantees. Additionally, the reciprocal velocity obstacle method takes into account the relative velocity between the Crazyflie and its obstacle enabling the generation of a smoother and safe trajectory [114]. The design of reciprocal velocity shares the collision avoidance responsibility between the Crazyflie and its peer obstacles to decrease the Crazyflie's angular deviation. The Figure 3.2 depicts the red-colored reciprocal velocity obstacle cones for each i^{th} Crazyflie (blue) and its j^{th} peer (red). These cones are not centered at the obstacle's current location because they are associated with the relative velocity space. The cone's intersection with the triangular BVC, ABC, results in the yellow space, where the i^{th} Crazyflie can safely move without any collisions from its j^{th} peer.

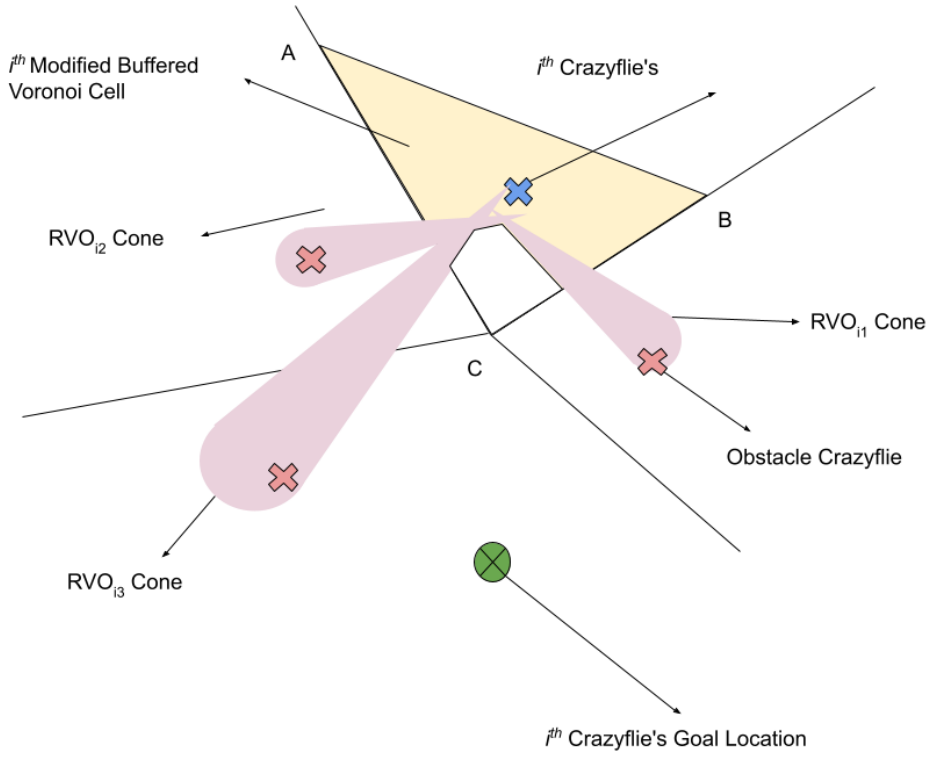


Figure 3.2: The superimposition of the reciprocal velocity obstacle and the triangular BVC, ABC, results in a region for the i^{th} Crazyflie to traverse without collisions.

The reciprocal velocity cone constraints, denoted as RVO_{ij}, define a set of relative velocities between the Crazyflie i and its obstacles $j \in [1, n] - i$ that can result in a collision within a given horizon Δ. These constraints are computed as:

$$\begin{aligned} VO_{ij} &= \left\{\mathbf{v} \exists \Delta > 0 :: t(\mathbf{v} - \mathbf{v}_j) \in \mathbf{D}(\mathbf{p}_{ij}, R_{ij})\right\} \\ RVO_{ij} &= \left\{\mathbf{v} \middle| \quad (2\mathbf{v} - \mathbf{v}_i) \in VO_{ij}\right\}, \end{aligned} \tag{3.8}$$

where VO_{ij} represents the baseline collision cone generated without considering the relative velocities between the i^{th} and j^{th} Crazyflies. Equation 3.8 geometrically modified VO_{ij} resulting in RVO_{ij}, where $\mathbf{D}(\mathbf{p}_{ij}, R_{ij}) = \left\{\mathbf{q}|\ ||\mathbf{q} - \mathbf{p}_{ij}|| \leq R_{ij}\right\}$ represents a sphere of radius R_{ij}, $\mathbf{q}$ represents any point inside this sphere $\mathbf{D}$ which is centered at $\mathbf{p}_{ij}$. The parameter $\mathbf{p}_{ij}$ denotes the relative position vector of the i^{th} and the j^{th} Crazyflies and R_{ij} denotes their combined radius. The reciprocal velocity obstacle constraints in the velocity space are transformed into the position space by time scaling Equation 3.8 using a time horizon Δ. The transformed constraints are:

$$RVO_{ij,tf} = \left\{4r^2 - ||2\mathbf{p}_i - (\mathbf{v}_i + \mathbf{v}_j)\Delta||^2 \leq 0\right\}. \tag{3.9}$$

Finally, the reciprocal velocity constraints $RVO_{ij,tf}$ in Equation 3.9 are superimposed on the BVC constraints in Equation 3.7 to generate an appropriate position waypoint with higher guaranteed collision avoidance for Crazyflies with positioning uncertainty in environments with moving obstacles.

3.3 Modified Collision Avoidance Approach

The baseline BVC algorithm is the Finding Closest Point algorithm introduced by Zhou et al. [112]. This algorithm designs the cell boundaries defined in Equation 3.2 to provide a basic framework for multiple Crazyflie collision avoidance. The algorithm takes the Crazyflie's radius and their current locations as input. Each

Crazyflie begins iterating over all of its peers that are acting as obstacles in order to compute the Crazyflie's cell boundaries. The cell boundaries are hyperplanes whose parameters $\mathbf{a}_{ij}$ and b_{ij} that are computed analytically to generate a waypoint $\mathbf{p}_{g^*,i}$. When the Finding Closest Point algorithm fails to generate a waypoint, the Crazyflie hovers in a place to avoid potential collisions.

The modified collision avoidance approach presented in Algorithm 1 is developed by combining the BVC's uncertainty and acceleration safety buffers with the reciprocal velocity obstacle constraints. The variable FOV_i represents the number of obstacles present inside the i^{th} Crazyflie's range. The modified baseline BVC algorithm is reflected in Lines 2-8 where the uncertainty-aware and acceleration safety buffers are used to generate the hyperplane parameters $\mathbf{a}_{ij}$ and b_{ij} and the reciprocal velocity constraints are transformed into the position coordinates. Finally, the nonlinear cost function is minimized by constraining the optimization by the values generated in Line 8. Based on the minimization's success, Crazyflie either hovers or moves towards its goal along the line joining its current location to the waypoint.

Algorithm 1 Modified BVC with reciprocal velocity obstacle-based collision avoidance algorithm.

INPUT: $r_{UAV}, \mathbf{p}_i, \mathbf{v}_i \qquad i \in FOV_i$
OUTPUT: $\mathbf{p}_{waypoint}\mathbf{p}_{waypoint}$

for $j \in FOV_i$ **do** ▷ Get hyperplane parameters.
  $\mathbf{a}_{ij} = \frac{2}{\sigma}(\hat{\mathbf{p}}_j - \hat{\mathbf{p}}_i)$.
  $b_{ij} = \frac{1}{\sigma}(\hat{\mathbf{p}}_j - \hat{\mathbf{p}}_i)^T(\hat{\mathbf{p}}_j + \hat{\mathbf{p}}_i) + r_{UAV}||\mathbf{a}_{ij}||$. ▷ Get uncertainty awareness and acceleration awareness parameters.
  $b_{ij} = b_{ij} + \beta_{acc,i} + \beta_{covar,i}$. ▷ Get buffered Voronoi boundary wrt j^{th} Crazyflie.
  Calculate $\overbrace{\underset{ij}{\nu}}$. ▷ Get time-scaled reciprocal velocity obstacle cones.
  Calculate $RVO_{ij,tf}$.
end for
▷ Calculate the inverse cosine cost function.
$\mathbf{p}_{g^*,i} = \text{argmin}_{\mathbf{p}\ni\overbrace{\nu_i},RVO_{ij,tf}} \; arccos(\frac{\mathbf{p}(\mathbf{p}_i - \mathbf{g}_i)}{||\mathbf{p}||||\mathbf{p}_i - \mathbf{g}_i||})$.
if NLopt succeeds **then**
  Move the Crazyflie towards the new waypoint $\mathbf{p}_{g^*,i}$.
else
  Hover the Crazyflie at its location.
end if

3.3.1 Collision-free Guarantee

All Crazyflies in the system initialize in a collision-free state outside each other's reciprocal velocity obstacle cones. Each Crazyflie has the same onboard collision avoidance algorithm employed on them to generate collision-free trajectories toward their respective goal locations. A probabilistic collision-free condition under these assumptions is referred to as a condition where the probability of the in-

ter Crazyflie distance $||p_{ij}|| > 2r_{UAV}$ is always less than the $1 - \delta$, where δ is a parameter used to devise the uncertainty-aware buffer in Equation 3.5. Crazyflies receive approximate state estimates from their sensors; hence, their safety can only be specified in a probabilistic manner. The buffer $\beta_{covar,i}$ factors in the state uncertainty and allows the Crazyflies to achieve probabilistic collision-free conditions under uncertainty. Additionally, the Crazyflie has time constraints to change its velocity and position to a desired state; hence, the buffers $\beta_{acc,i}$ indirectly allow the Crazyflies to safely reach its waypoint in a given time horizon. The acceleration awareness buffer retracts the Voronoi cell size even more and heuristically allows Crazyflies to attain the probabilistic collision-free condition. Voronoi cells' convexity always restricts the Crazyflie's path within the cell promoting safety.

The baseline BVC was originally designed to only provide passive safety among single-ordered agents and was extended to UAVs with motion control based on model predictive control [45]. The Crazyflies are small-sized UAVs with a simpler controller design that prioritizes computational efficiency. The baseline BVC is modified to seamlessly support Crazyflie's controller design, to allow more flexibility and computational room for future development on the system. Adding reciprocal velocity obstacle constraints to the baseline BVC allows for improved collision avoidance guarantee for Crazyflie-like systems that have a simpler onboard controller. The reciprocal velocity obstacle constraints satisfy the collision-free condition in deterministic settings without state uncertainty. A probabilistic collision-free state is attained by introducing a longer time horizon Δ greater than the Crazyflie state machine update rate of 0.002 seconds. The longer time horizon

indirectly compensates for the uncertainty in the Crazyflie states by increasing the collision cone's size. These reciprocal velocity obstacle constraints heuristically limit the Crazyflie's speed by generating a waypoint closer to the Crazyflie's current location.

3.4 Simulation Environment

The simulation environment in Figure 3.3 for multiple Crazyflie navigation can fly the Crazyflies in three different modes, (1) Takeoff, (2) Navigation, and (3) Land. Each Crazyflie is assigned two positions, a start and a goal to demonstrate the algorithm's collision avoidance ability. Crazyflies must begin from the first Takeoff mode and ascend to a certain altitude to reach their start locations. The Crazyflies later switch to the Navigation mode to reach their assigned goal locations in $t_{NavigationMode}$ seconds while avoiding obstacles. This mode utilizes the modified collision algorithm developed in Algorithm 1 with BVC and reciprocal velocity obstacle constraints including the uncertainty-aware and acceleration safety buffers. Once the Crazyflies reach their goal locations, they switch to the final Land mode. Once all the Crazyflies land, the simulation is terminated.

3.4.1 Crazyswarm Simulator

Simulations are performed using the Crazyswarm platform that allows multiple Crazyflie 2.X flight visualization. The platform supports integration with the LPS

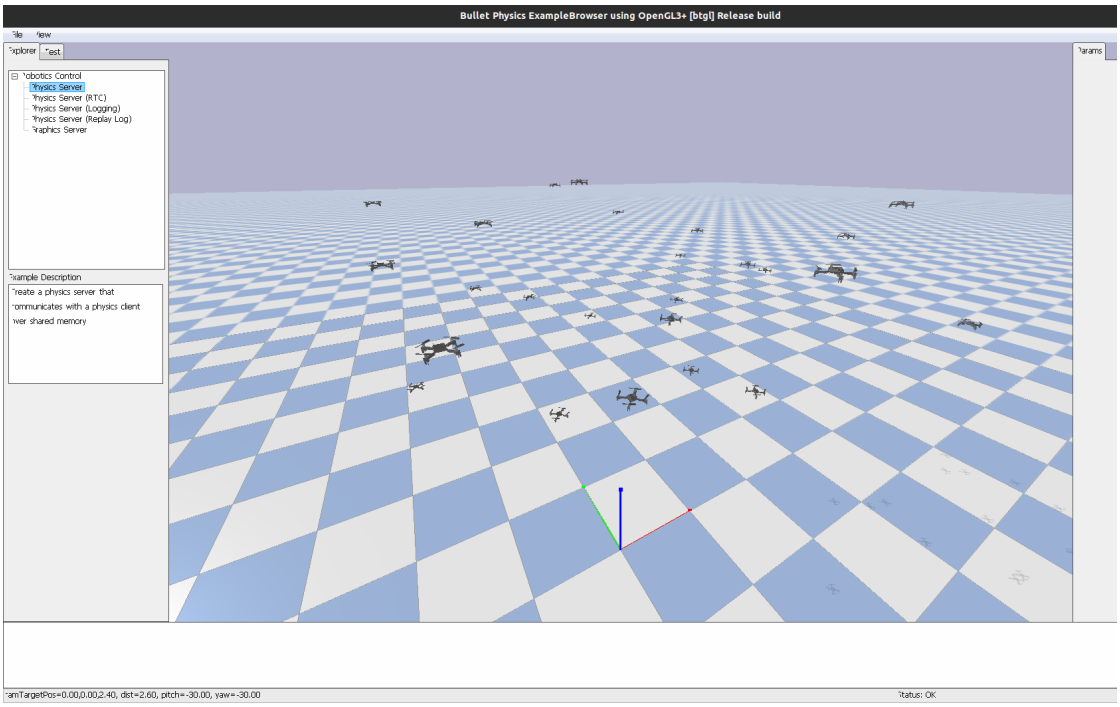


Figure 3.3: The integrated physics simulation environment incorporating the Crazyswarm API with the gym_pybullet_drones package.

localization hardware and provides Python wrappers to integrate the Crazyflie firmware modules. The platform also supports fewer radios per Crazyflies by design, unlike the official Bitcraze's unicast communication. This project's collision avoidance method is introduced as an app layer on the Crazyflie firmware stack. Corresponding Python wrappers to translate the collision avoidance method to the simulator are generated using the Simplified Wrapper and Interface Generator (SWIG) Python wrapper library. A firmware binding in Python is generated for certain Crazyflie submodules to aid in debugging and visualization of mathematically complex onboard algorithms.

The onboard collision avoidance design minimizes the communication bandwidth between the base station and the Crazyflies. A single packet containing the state position, goal position, and the maximum time frame to reach the goal is provided to the Crazyflie at startup without changing the Crazyswarm simulator's software architecture.

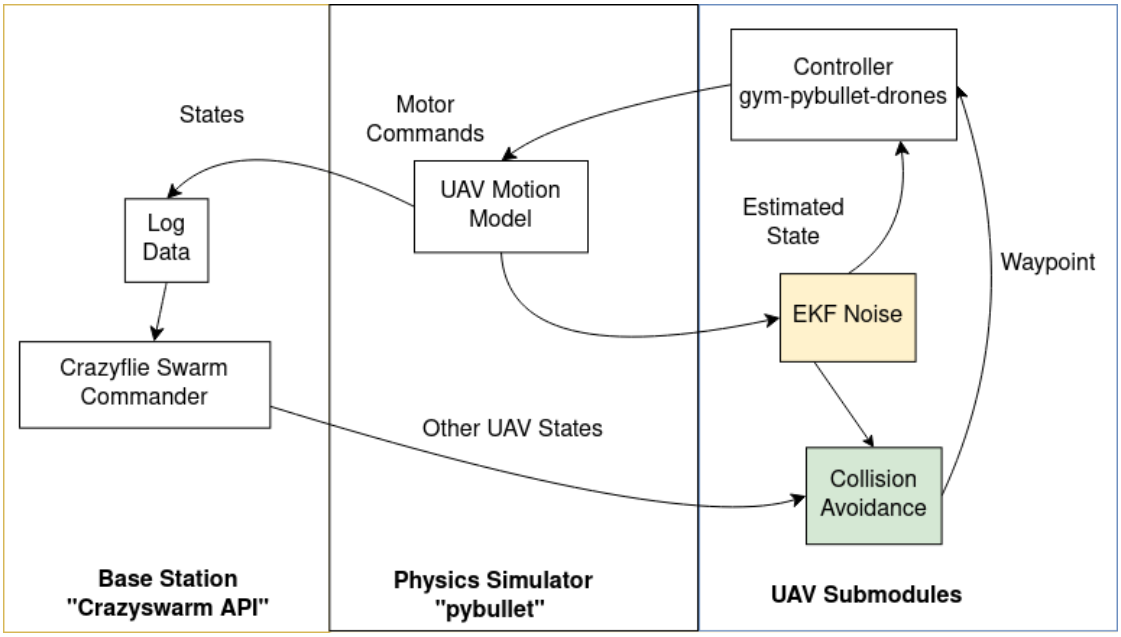


Figure 3.4: Crazyswarm API integrated with the physics simulator operates at 500 hertz. This integration incorporates the collision avoidance (green) containing the modified Buffered Voronoi Cell (BVC) algorithm, with the controller, and Estimated Kalman Filter (EKF) (yellow) submodules.

The Crazyswarm simulation system diagram is provided in Figure 3.4. The Crazyflie objects are initialized and launched in the base station using Crazyswarm API. The base station sends Takeoff, Navigation, and Land commands to all the Crazyflies in the simulation in a parallel manner. These Crazyflie objects

move across the simulated testbed throughout the simulation trial [115]. The Crazyswarm API interacts with the collision avoidance submodule with the modified BVC algorithm along with the controller, and EKF submodules in Figure 3.4, which is provided by the gym-pybullet-drones library. The collision avoidance submodule developed for this project generates a high-level trajectory waypoint for the controller and the estimated Kalman filter submodule generates noise positioning data in the simulation. The simulated Peer-to-Peer communication system tracks each flying Crazyflie in the simulation and sends their pose information to the collision avoidance submodules of all other Crazyflies. Messages from each Crazyflie's individual submodules arrive asynchronously. The base station also logs the active Crazyflies' pose information. The Crazyswarm API runs the Crazyflie instances at 500 hertz, a frequency similar to Crazyflie's onboard state machine.

3.4.2 Motion Model

The Crazyswarm simulator does not consider physics-based simulations that include its second-order dynamics controlled by the motor rotation speeds. The simulator also ignores aerodynamic forces. This project's algorithm developed is tested in simulation; hence, to improve the algorithm's reliability on the real hardware, a high-fidelity simulation package, gym-pybullet-drones library is integrated with the Crazyswarm platform [116]. The gym-pybullet-drones library provides a Crazyflie dynamics model that simulates its PID controller in Python [93]. This package leverages the system identification data generated by Bitcraze to create

Crazyflie's motion model in the simulation that is tested against the Crazyswarm commands and the collision avoidance waypoint generator.

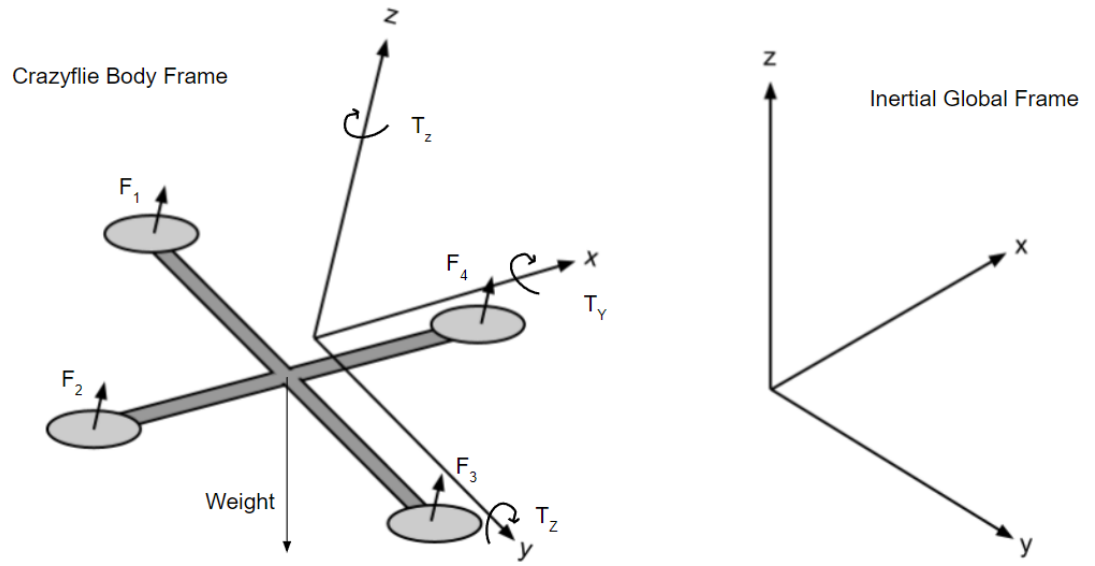


Figure 3.5: The free body diagram representing the forces and torques experienced by the Crazyflie along the body-fixed frame.

Crazyflie's free body diagram is depicted in Figure 3.5 along with the corresponding reference frames. The Crazyflie's weight acts along the inertial Z axis that is counteracted by its motor thrust and torques. The body-fixed frame is centered at the Crazyflie's center of mass. The motor forces and torques on the Crazyflie are controlled instantaneously by the motor rotational speeds in the simulator, as provided in Equations 3.10, where k_f and k_t are predefined constants and the Crazyflie's motor rotational speeds are represented as $\omega_i, i \in [1, 4]$. The motor rotational speeds are provided by the controllers to move the Crazyflies to

appropriate reference locations.

$$F_i = k_f * \omega_i^2$$
$$T_i = k_f * \omega_i^2.$$

The net body forces and torques on the Crazyflie are represented as:

$$\begin{aligned} F_{net} &= \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 - grav \end{bmatrix} \\ T_{net} &= \begin{bmatrix} \frac{F_1 + F_2 - F_3 - F_4}{\frac{L}{\sqrt{2}}} \\ \frac{-F_1 + F_2 + F_3 - F_4}{\frac{L}{\sqrt{2}}} \\ -T_1 + T_2 - T_3 + T_4 \end{bmatrix}, \end{aligned} \tag{3.10}$$

where $grav$ represents the Crazyflie's weight and L represents the distance between the center of mass and one of the Crazyflie's motors. The gym-pybullet-drones library further provides the drag, ground effect, and downwash in the motion model independently. These effects are also included in the present design provided by the gym-pybullet-drones package. The net dynamic quantities are obtained by summing the forces and torques listed in Equation 3.10 and are implemented in the simulator to update the Crazyflie position and velocities at each time step t. The velocity Vel, position pos, angular rates $RPYrate$, and angles RPY in the

body-fixed frame are updated as:

$$\begin{aligned} Vel_{t+dt} &= Vel_t + \frac{F_{net}}{M} dt \\ RPYrate_{t+dt} &= RPYrate_t + T_{net} J^{-1} dt \\ RPY_{t+dt} &= RPY_t + RPYrate_t dt \\ pos_{t+dt} &= pos_t + Vel_t dt, \end{aligned} \tag{3.11}$$

where M represents the Crazyflie's mass and J represents the Crazyflie's moment of inertia. The Euler $3-1-2$ rotation scheme is leveraged to generate transformations from the body-fixed to the inertial frame of references and update the Crazyflie's position instances in the simulator [93].

3.4.3 Controller

The onboard controller receives current state estimates from the localization module with EKF and minimizes the error between its current and reference states. The Crazyflie 2.X controller in the gym-pybullet-drones library has a PID design that relies on the position error, its derivative, and its integral to generate desired control inputs and keep the Crazyflie stable [116]. This Crazyflie's cascaded decoupled PID controller is shown in Figure 3.6 where position and attitude are controlled separately. The attitude control block generates control inputs as the motor rotation speed for each Crazyflie's motor. These control inputs are sent into motor dynamics blocks that generate a target thrust implemented to move the

Crazyflie in the simulator.

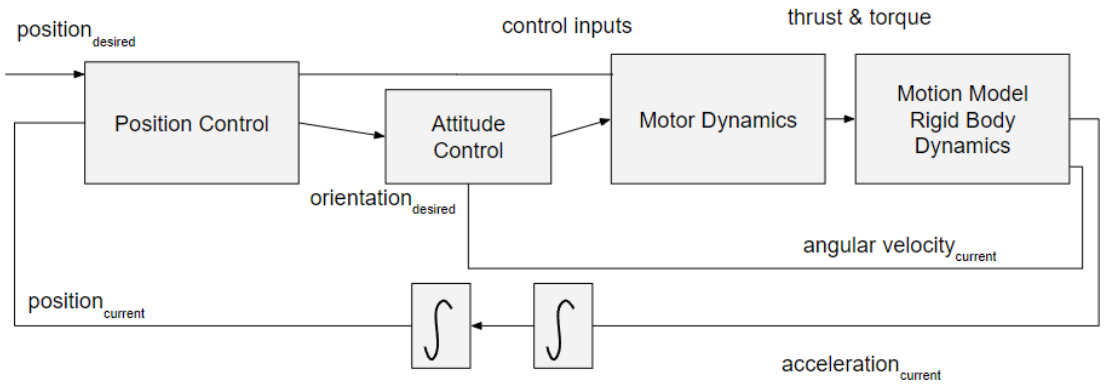


Figure 3.6: The Crazyflie's cascaded decoupled control loop implemented in the gym-pybullet-drones' controller submodule [116].

3.4.4 Nonlinear Optimization Setup

The open-source Non-Linear optimization (NLopt) library supports the optimization presented in Equation 3.12, where $f(k)$ is the cost function of the state variables represented by $k \in \mathbb{R}^n$, where n is the space dimensionality [117]. This cost function is subjected to both linear constraints represented by $g(k) = 0$, and nonlinear constraints represented by $h(k) = 0$. The collision avoidance algorithm uses Equation 3.12 as a baseline.

$$\min_{\{k \mid h(k) \leq 0 \mid g(k) = 0\}} f(k) \tag{3.12}$$

The NLopt library that solves this project's cost function is provided in Equation 3.3 and is constrained by inequalities in Equations 3.7 and 3.8. The cost

function in Equation 3.3 is a strictly increasing inverse cosine function that results in the global minimum solution. The library supports four algorithms for this cost function, among which the Constrained Optimization BY Linear Approximations is implemented in this project [118]. The library has a well-defined API to represent a list of constraints and their corresponding derivatives in the form of a single matrix $c : \mathbb{R}^n \to \mathbb{R}^m$, where m represents the constraint's dimensionality, to improve the solver's space complexity. The optimization results in a waypoint for the Crazyflie in the three-dimensional space. The output vector $\mathbf{k}_{sol}$ is a point in the space represented by $\mathbb{R}^3$ that satisfies all its constraints over the entire feasible region. The state's dimensionality is limited to three, making the local optimization search work within a reasonable computational time. The solver generates an error code when the output state does not abide by the set constraints. The Crazyflie with this solver error is commanded to hover at its previous location instead of moving to the erroneous waypoint generates as an approximate solution along with the error code, to maintain a collision-free guarantee. All the Crazyflies in the air have an onboard collision avoidance system; hence, other Crazyflies with optimization success avoid the hovering Crazyflies and prevent collisions.

The library has C language support; hence, the algorithm is written as an app layer running in parallel with other subsystems on the Crazyflie firmware. The SWIG package is implemented to generate Python firmware bindings that allow firmware-in-the-loop testing with the Crazyswarm simulator [115]. The automatic differentiation method was implemented in MATLAB to generate the derivatives of each constraint. The derivatives are presented in Equation 3.13,

where $h_{grad}(BVC, ax)$ denotes the BVC constraints' gradients, $h_{grad}(RVO, ax)$ denotes the gradient of the reciprocal velocity constraints, $a_{ij,ax}$ is the component of the hyperplane parameter $\mathbf{a}_{ij}$, $p_{ax,i}$ is the current Crazyflie's position vector, $v_{ax,i}$ its corresponding velocity component, $v_{ax,j}$ corresponds to the j^{th} obstacle Crazyflie's velocity along the ax axis for $ax \in (x, y, z)$ and Δ depicts the time horizon considered to generate reciprocal velocity obstacle cones.

$$
\begin{aligned}
h_{grad}(BVC, ax) &= \mathbf{a}_{ij,ax} \\
h_{grad}(RVO, ax) &= -\frac{2p_{ax,i} - (v_{ax,i} + v_{ax,j}) * \Delta}{\sqrt{\sum_{ax \in (x,y,z)} 2p_{ax,i} - 2\Delta(v_{ax,i} + v_{ax,j})p_{ax,i} - (v_{ax,i} + v_{ax,j})\Delta}}
\end{aligned}
\tag{3.13}
$$

3.4.5 Estimator

The LPS system in theory has a standard deviation of 10 centimeters, but the performance of this system relies heavily on the environment space [19]. Bitcraze has further proposed that the onboard estimated Kalman filter has a standard deviation of 15 centimeters along each of the position vector's components when there are no interfering WiFi signals around the LPS setup [6]. The preliminary experiments were performed with the hardware placed individually across various locations on the HMTLab's Swarm testbed to estimate the LPS signals' standard deviation. The Crazyflies in empty spaces, away from static obstructions (e.g., wooden hub, metallic testbed frame) have a positioning uncertainty within the

standard specification. A standard deviation of 20 centimeters along the body-fixed X and Y axes and up to 30 centimeters of standard deviation along the Z axis for the estimated Kalman filter outputs was measured in the locations near static obstructions [119]. The Crazyflies are controlled using acceleration generated by their motors along the body-fixed Z axis; hence, the estimated Kalman filter design impacts the Z component's accuracy more than the other two.

Static obstructions (e.g., metallic testbed frame, wooden hub) result in non-line-of-sight reflections degrading the positioning accuracy. Multiple Crazyflies flying in a densely populated environment will also increase the non-line-of-sight reflections, degrading the positioning accuracy; hence, the standard deviation from the preliminary experiments is incorporated to simulate the estimated Kalman filter design in Crazyswarm simulations as:

$$posin_{ax,new} = posout_{ax,old} + Velin_{ax,new}dt + random(sd_{ax}),$$

where the $random(sd_{ax})$ with $ax \in (x, y, z)$, represents the Gaussian noise with a standard deviation of sd_{ax} centimeters added to the Crazyflie position vector $posin_{ax,old}$ in the simulator, resulting in the simulated Kalman filter position output $posin_{ax,old}$ along the axis ax.

3.4.6 Obstacle Detection

The BVC on Crazyflies was first implemented for localization with a motion capture system [45], [112], [114]. Each Crazyflie with this localization receives a radio packet containing the state information of all other Crazyflies on a common radio channel. The information in these packets is used to locate and avoid other flying Crazyflies in the environment. Extending the system to Crazyflies with LPS does not support state information sharing among the Crazyflies; hence, Bitcraze has developed a Peer-to-Peer communication system to share Crazyflie locations among their peers for collision avoidance. Peer-to-Peer radio packets of size up to 60 bytes can be implemented to encode information while maintaining communication compatibility with the base station radio [120]. These packets are transmitted and received by the 2.4 giga hertz onboard Crazyflie radio. This communication link is independent of the link between the base station radio dongle and the Crazyflie.

The data received by Crazyflie's Peer-to-Peer module is filtered with respect to its range, the region highlighted in blue in Figure 3.7. Only the obstacles present in the blue segment are considered an active threat. This segment lies within a sphere of the radius of r_{lookup} centered at the body fixed Crazyflie's X-axis and has an arc of 200°. Limiting the Crazyflie's range will result in fewer constraints associated with the BVC generation; hence, this change in obstacle determination improves the algorithm's space and time complexity, which increases with an increasing number of obstacles. This algorithmic design to determine in-range obstacles also

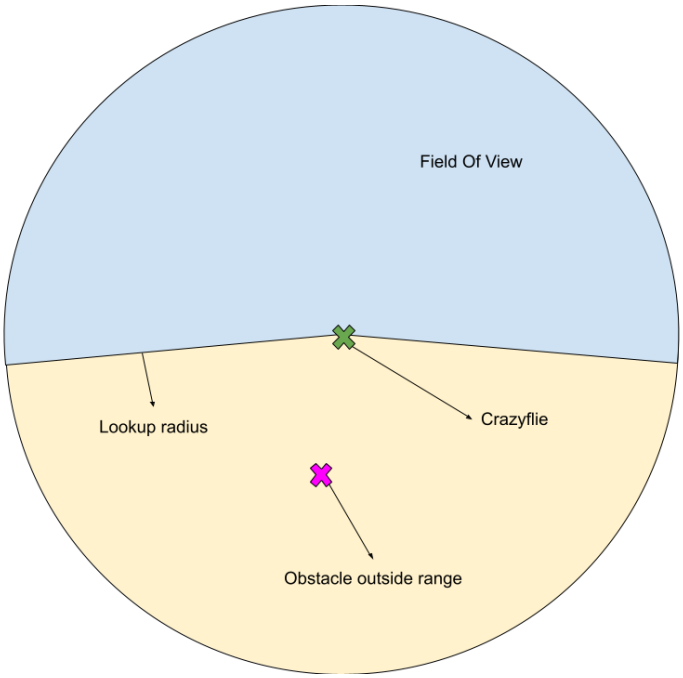


Figure 3.7: The Crazyflie's range (blue), which limits the increase in computational complexity as the number of obstacles increases.

supports collision-free guarantees satisfied by the uncertainty-aware BVC and the probabilistic reciprocal velocity obstacle approach.

Inter Crazyflie encounters can be classified into three types, (1) In-range, where both Crazyflies are in each other's range, (2) Out-range, where both Crazyflies outside their range, and (3) Leader-follower, where one Crazyflie follows the other such that it sees the other Crazyflie, but not vice versa. Crazyflies in the In-range encounter use the algorithm to modify their Voronoi cell's volumes to maintain safety. The Voronoi cells of Crazyflies in the Out-range encounter are not directly impacted by each other; hence, the cells' volumes are not modified with respect to each other, and the Crazyflies remain collision-free with each other. Finally, in

the Leader-follower encounter the Voronoi cell is unequally influenced by the two Crazyflies, where only the follower Crazyflie takes the responsibility of avoiding collisions with the leader. The leader Crazyflie does not directly contribute to the collision-free guarantee. Based on each pair's encounter type, collision avoidance constraints are activated to enable the collision-free guarantee presented in Chapters 3.1.3 and 3.3.

Chapter 4: Experimental Analysis

The Chapter presents the experimental protocols and related analysis of the results via trials performed in the integrated physics simulator. This integrated simulator utilizes the Crazyswarm API, allowing multiple Crazyflie instances to traverse the shared space concurrently. The algorithm's analysis is performed by computing novel metrics based on the simulation logs. The experiments performed on the multiple Crazyflie system aim to answer the Research Questions **R1** by utilizing the Hypotheses **H1**, **H2**, and **H3** as:

R1 Can small-sized UAVs with LPS-induced-positioning noise generate navigation commands within the system's update rate while avoiding dynamic obstacle peers?

H1. The optimization-based modified BVC algorithms will generate paths so that each Crazyflie reaches its goal within a desired navigation time.

H2. The Crazyflies when using the modified BVC algorithms will have a mean percentage of Safe or Partial overlap count for more than 80% of each trial.

H3. The optimization solver incorporated into the Modified BVC algorithms always generates a waypoint within the system's update rate (0.002 seconds).

The baseline algorithm developed for the Crazyflies utilizes an optical motion capture positioning system that can report the peer obstacle positions with precision.

LPS induces substantial noise in the Crazyflie's state estimates. Both the modified BVC algorithm enables navigation command generation from this noisy positioning data. Additionally, the modified BVC algorithm with reciprocal velocity obstacle constraints predicts collision with its peers by utilizing their noisy state estimation data obtained from Peer to Peer communication. Answering the Research Question **R1** will provide insights into the collision avoidance ability of the algorithms under positioning uncertainty that are discussed in Chapter 3.

The experiment's independent and dependent variables, the metric design for the analysis of the collision avoidance algorithms' ability for multiple UAV scenarios, the system's constants, and the experiment's methodology employed for the system's analysis are described in this Chapter.

4.1 Independent Variables

Varying the algorithm type and the number of Crazyflies can help compare their collision avoidance capability and scalabilty in the presence of dynamically moving obstacles. This comparative analysis also provides insights into the system's performance against LPS-induced positioning uncertainty; hence, the core experimental independent variables include the type of collision avoidance algorithm employed by the Crazyflies and the number of Crazyflies simultaneously flying in a trial.

Crazyflies act as dynamic obstacles to each other. The independent variables are summarised in Table 4.1. The algorithm type includes (1) the baseline

Table 4.1: Core experimental independent variables utilized for the system's analysis.

Independent Variables	Values
Algorithm Type	Baseline, MBVC, MBVC_Vel
Number of UAVs	4, 9, 15, 25

BVC algorithm (Baseline), (2) the modified BVC algorithm with only acceleration and uncertainty-aware buffers (MBVC), and (3) the modified BVC algorithm (MBVC_Vel). Only the MBVC_Vel considers the obstacles within a specific range as presented in Figure 3.7. The experiments involve from 4 to 25 agents distributed across a cuboid testbed with a square base of 5 meters on each side and a height of 5 meters.

4.2 Dependent Variables

The dependent variables presented in the Chapter were utilized in the analysis to verify the project's hypotheses. These variables include the *number of collisions*, the *completion rate*, the *overlap category count*, the *navigation time category*, the *optimal direct path*, the *actual path*, the *net distance deviation*, the *number of actual path intersections*, the number of *collision avoidance procedures*, and the *waypoint computation time*.

A collision occurs when the distance between any two Crazyflies is less than $2 * r_{UAV}$. After a collision, the Crazyflies lose their stability and fall on the simulation's ground at an altitude of 0 meters. The connection between the Crazyflie and the

base station is still maintained after the fall and it remains on the ground until the trial ceases. This information is utilized to compute the *number of collisions*.

The experimental constant $t_{NavigationMode}$ in Table 4.2 specifies a deadline for the Crazyflies in their Navigation mode to reach their goal. Once the deadline is reached, the Crazyswarm API disables the Navigation mode and the Crazyflies hover in the air. The *completion rate* for a trial is defined as the percentage of Crazyflies that are within 0.75 meters distance from their goal for more than 95% of the last 0.25 seconds before the deadline expired. The Crazyflies have a maximum allowable speed of 2 meters per second and they can travel up to 0.5 meters within a duration of 0.25 seconds. This information is used as a reference to define and generate the *completion rate* of each trial. The Crazyflie can approach its goal and oscillate around it until the end of the Navigation Mode; hence, a tolerance distance of 0.3 meters from its goal is chosen as the second reference to detect whether the Crazyflies have oscillatory trajectories, where they remain within 0.75 meters but fail to reach within 0.3 meters from their goal. The percentage of time the Crazyflie remains inside this distance tolerance is also decreased from 95% to 75% to improve the analysis.

The inter Crazyflie distances for all possible Crazyflie-Crazyflie combinations at a simulation time step are used to generate the *overlap count*. These distances are classified into four categories including, (1) Safe when the distance is > 0.75 meters, (2) Partial when the distance is within 0.75 and 0.5 meters, (3) Unsafe when the distance is between 0.5 and 0.25 meters, and (4) Collision when the distance is < 0.25 meters.

The specified distance ranges used in the classification are inspired by the preliminary results collected on the HMTLab Swarm Testbed [119]. The estimates' standard deviation is increased to 0.3 meters due to non-of-line-sight reflections from obstacles; hence, an assumption is made that each Crazyflie within 0.5 meters can possibly collide. The upper bound of the Partial category is derived from the fact that if both UAVs are at a distance 0f 0.75 meters from each other and the positioning noise has a standard deviation of up to 0.2 meters along each axis, then there is less than 2% chance that the Crazyflies will be classified as colliding. Further, UAVs have a physical volume of radius 0.125 meters. Assuming one of the UAV's location perfectly coincides with its predicted state, and the other is $0.2 * 3$ meters away from its estimated pose, there is a slight possibility that the UAVs within $0.6 + 0.125$ meters are in collision; hence, 0.75 meters was chosen to allow a larger reference defining Safe category.

After classifying the distances, a count for each Crazyflie was computed by summing up the number of times a Crazyflie was in a category across each time step of the trial. Later, *overlap category count* is computed by summing together all the Crazyflies' counts within a trial for each category separately.

The Navigation mode in each trial lasts for $t_{NavigationMode} = 20$ seconds. Navigation mode for each Crazyflie can be classified into (1) *Navigation Time* when the Crazyflie has no peers in its range, and (2) *Avoidance Time* when the Crazyflie has at least one peer obstacle in its range. The *navigation time category* utilizes the r_{lookup} range for obstacle filtering to calculate the total time each Crazyflie spent in each of the navigation time categories. The *Avoidance time* can be fur-

ther classified into (1) *Collision Time* when the Crazyflie is in Collision, (2) *Hover Time* when the Crazyflie is avoiding obstacles by hovering in the space, and (3) *Non-Collision Time* when the Crazyflie is not in the first two *Avoidance Time*'s categories. The simulation logs contain the Crazyflies' positions and timestamps across each trial. Currently, this data set does not include whether the Crazyflies defaulted to hovering due to a lack of waypoint generation within an update cycle; hence, the *Avoidance Time*'s subcategories were not included in the analysis. Additionally, the *number of avoidance procedures* is defined as the number of times the Crazyflie is in the *Avoidance Time* category while avoiding at least one of its peers present within a range of r_{lookup}.

Further, the *optimal direct path* of a Crazyflie is the Euclidean distance between its start and goal locations. This path is a straight line path traveled by the Crazyflie in the absence of its peers. The *actual path* is the total distance traveled by the Crazyflie. This path is calculated by summing the Euclidean distances between each consecutive Crazyflie's positions sampled across each trial. The area of the polygon formed consecutive actual path points and their projection on the optimal direct path generates the *net deviation* between each consecutive projection, the area between *actual path* and the *optimal direct path*. The area of the polygon is calculated using the *Shapely* library and the sum of all the resulting polygonal areas results in the *net deviation*.

The number of *potential path intersections* represents the sum of all the *optimal direct path*s intersections generated for all possible Crazyflie-Crazyflie combinations in a trial. The distance between optimal direct paths also provides information

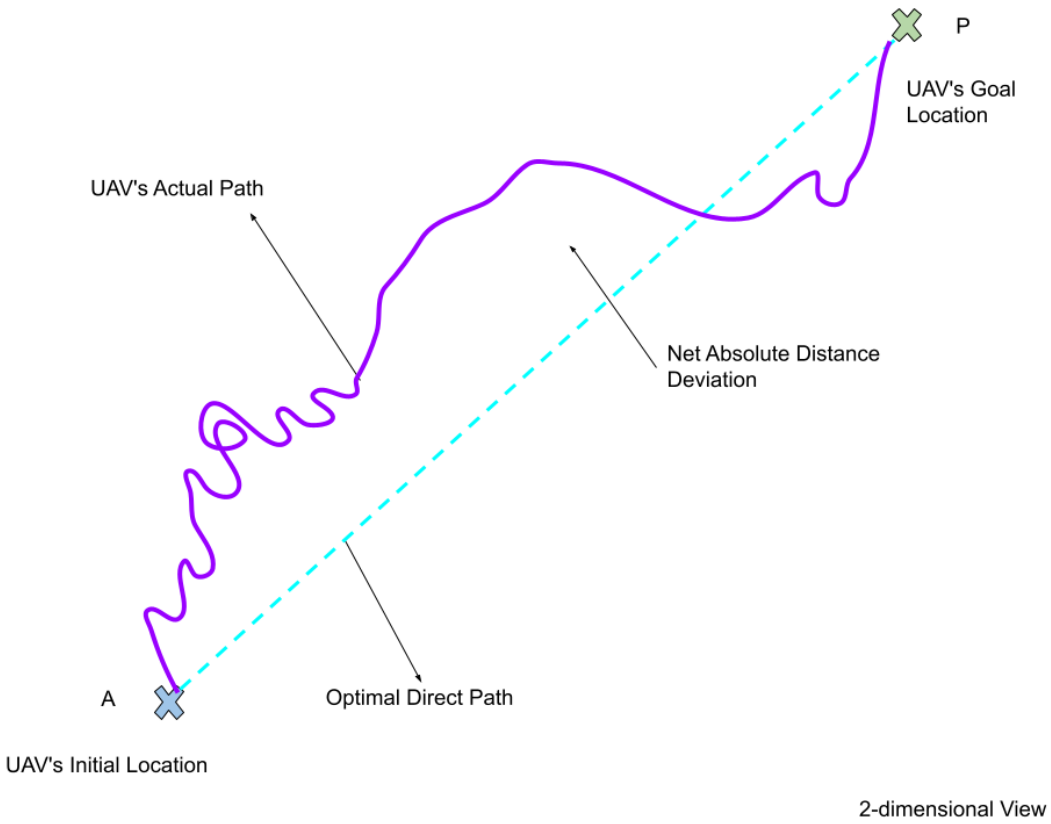


Figure 4.1: 2-dimensional view of the area between curves, the *optimal direct path* (Cyan) and the *actual path* (Purple) followed by the Crazyflie. This area is the *net deviation* variable.

on the experimental setup design. This distance variable provides information regarding the potential collisions that influence the Crazyflie's deviation from its optimal direct path. This variable is the final distance-based measure used in the system's analysis.

The number of peer obstacles in the Crazyflie's range utilized to generate the buffered Voronoi cell boundaries to generate a waypoint is referred to as *collision avoidance procedures*. Increasing the number of Crazyflies is expected to increase

the *collision avoidance procedures* for each algorithm utilized in the analysis. The *waypoint computation time* variable is the time required by the algorithm to compute a waypoint at each simulation time step for each Crazyflie. Each Crazyflie in a trial has approximately $10,000$ samples of this variable that are aggregated for an independent variable combination for comparative analysis. The system's update occurs at every 0.002 second and the *waypoint computation time* samples below this value denote the algorithm's computational efficiency.

4.3 Experimental Methodology

This chapter provides the experimental findings, which aim to showcase the performance of the collision avoidance algorithms for multiple Crazyflie scenarios by computing the statistical measures corresponding to the Variables in Chapter 4.2

Each trial begins with the Crazyflies taking off (Takeoff mode). After reaching a certain altitude, each Crazyflie switches into the Navigation mode. All Crazyflies in this mode traverse along their paths to their respective goal locations within a predefined duration of time $t_{NavigationMode}$. This time parameter is an essential input to the Crazyswarm API developed by Bitcraze to support Crazyflie's low-level processes onboard. Upon receiving the goal location, the collision avoidance submodule generates a waypoint, which is sent to the physics simulator containing Crazyflie's controller and the dynamic model. The feedback from the dynamic model is used to simulate EKF data for the collision avoidance module in order to generate the waypoint for the next simulation timestamp. After $t_{NavigationMode}$

seconds have elapsed, all the Crazyflies disable their Navigation modes and switch to Land mode. The simulation halts once all the Crazyflies have landed.

Preliminary testing indicated that the minimum spacing between Crazyflies' start and goal locations in the Navigation mode is sufficient to avoid collisions during the Takeoff and Land modes; hence, the simulation data from the Takeoff and Land modes are filtered out to solely focus on the Navigation mode's performance that lasts for $t_{collisionAvoid}$ resulting in $10,000$ samples per Crazyflie in each trial. The sample logs contain the simulation's timestamp, waypoint computation time, and the Crazyflie's current, and goal locations.

The experimental setup relies on the constant parameters listed in Table 4.2. The Crazyflies' initial location at the start of the Simulation is distributed across the vertices of a grid with a minimum spacing $start_{spacing}$. During takeoff, each Crazyflie attains an altitude h_{start}. The goal locations (x, y) are generated using a Poisson disc sampler, which generates coordinate pairs with a minimum spacing $goal_{spacing}$ [121]. The goal location's z coordinate is set to an altitude h_{goal} randomized by adding a Gaussian noise of standard deviation $goal_{\sigma}$. Minimum spacing between the start and goal location ensures that any two Crazyflies' start or goal locations do not lead to a collision. The locations are chosen within a sphere of a 5 meters radius.

A Crazyflie's bounding sphere's radius r_{UAV} enclosing it completely is provided by the Crazyswarm platform [115]. Crazyflies have a stable operational range of acceleration acc_{max} and velocity $velocity_{max}$ that determines the stability of the onboard controller. Additionally, necessary algorithmic parameters described in

Table 4.2: System parameters' values utilized to set up the simulation environment and implement the collision avoidance algorithms.

System Parameter	Value	Description
acc_{max}	0.5 $\frac{meters}{seconds^2}$	Crazyflie's maximum acceleration.
$velocity_{max}$	2 $\frac{meters}{seconds}$	Crazyflie's maximum velocity.
δ	0.01	Crazyflie's probability of collision.
σ_{ax}	0.20 meters	Covariance matrix's variance component corresponding to the axis ax.
r_{UAV}	0.125 meters	Crazyflie's bounding sphere's radius.
Δ	0.05 seconds	Reciprocal velocity obstacle' planning horizon larger than the simulation time step.
$t_{NavigationMode}$	20 seconds	Navigation mode's time duration required to integrate Crazyswarm API.
$start_{spacing}$	$4 * r_{UAV}$	Initial location's minimum spacing.
$goal_{spacing}$	$4 * r_{UAV}$	Goal location's minimum spacing.
h_{start}	1 meters	Start location's z coordinate.
h_{goal}	3 meters	Goal location's z coordinate.
$goal_{\sigma}$	0.5 meters	Standard deviation of the noise introduced in h_{goal} for altitude randomization.
dt	0.002 seconds	Simulation time step.
r_{lookup}	1 meters	Crazyflie's range for obstacle detection.

Chapters 3.1, 3.2, and 3.3, which are employed in the experiments are also listed in Table 4.2.

4.4 Experimental Results

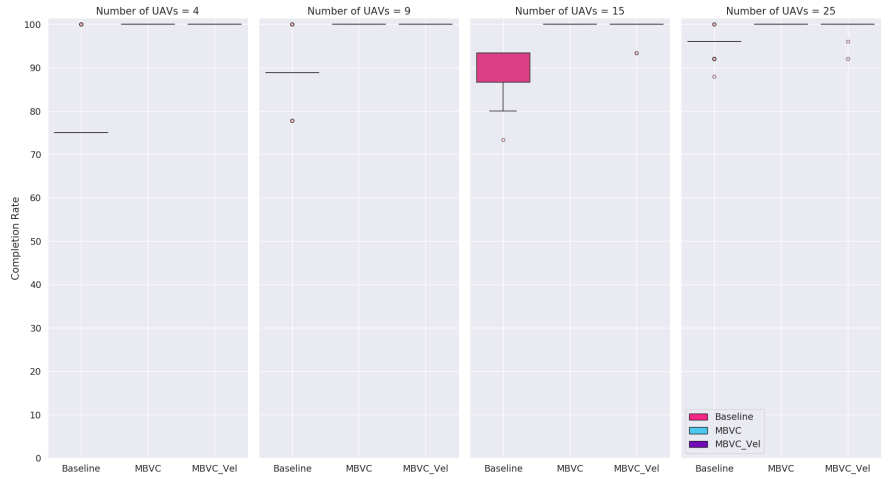
The three algorithms, MBVC, MBVC_Vel, and Baseline's collision avoidance ability were compared using the dependent variables presented in Chapter 4.2. These

variables are computed using the Crazyflie's locations, the inter-Crazyflie distances, and the computation time across each trial. This analysis enables the comparison of these algorithms across the number of UAVs in an integrated physics simulator with LPS.

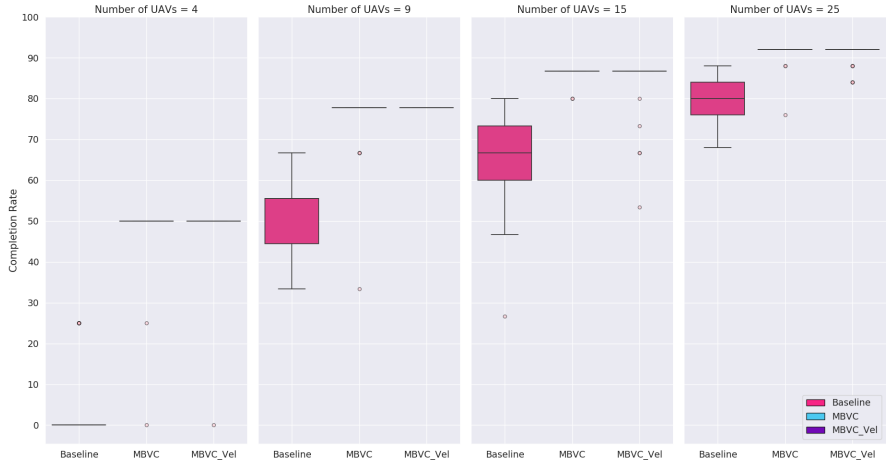
4.4.1 Mission Success-based Results

The *completion rate* represents the percentage of the Crazyflies reaching within 0.30 and 0.75 meters from their goal during the last 0.25 seconds for more than 95% or 75% of the duration. The modified BVC algorithms perform close to a 100% *completion rate* and outperform the Baseline as shown in Figure 4.2a, 4.2b and 4.2c. The Baseline's performance is relatively better for a relaxed distance tolerance of 0.75 meters and 75% of the duration as shown in Figure 4.2a and 4.2c respectively. Generally, as the number of UAVs increased the *completion rate* increased. The opposite trend of *completion rate* was expected, where a smaller number of UAVs resulted in larger *completion rate* values.

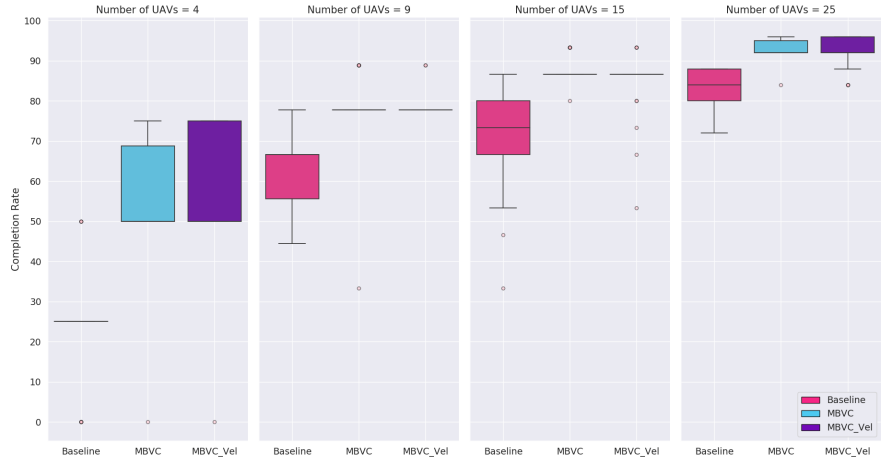
The MBVC algorithm for 25 agents has an approximate completion rate of 95% which suggests that 2 out of 25 agents fail to reach their goal. Similarly for all independent variable combinations up to 4 agents fail to reach their goals while maintaining the completion rate rule. This depicts that the performance of the algorithms is pretty consistent with an increasing number of agents and the MBVC_Vel performs better than the Baseline as seen in Figure 4.4.



(a) Tolerance: <0.75 meters, 95%



(b) Tolerance: <0.3 meters, 95%



(c) Tolerance: <0.3 meters, 75%

Figure 4.2: The *completion rate* for each algorithm and a number of Crazyflies combination, where (a) <0.75 meters, (b) <0.3 meters for more than 95% and (c) <0.3 meters for more than 75% of the last 0.25 seconds of the Navigation mode. Pink markers outside the range represent the outliers.

The design of the reciprocal velocity obstacle aims to share the collision avoidance ability among agents. This assumption was tested by measuring all possible inter-Crazyflie distances during each trial. An increasing number of Crazyflies result in increasing Crazyflie-Crazyflie combinations in Table 4.3; hence, the *overlap count* percentages corresponding to the *overlap count* in Table 4.4 are used in the analysis. A larger Partial *overlap count* percentage was expected for the MBVC_Vel algorithm corresponding to a near 100% *completion rate*; however, it was not observed. Conversely, the inter-Crazyflie distances remain in the Safe category for the majority of cases. Additionally, The increase in the number of Crazyflies increases the mean percentage of the Safe *overlap count*. The percentage values in other categories decrease with an increasing number of UAVs. The results also indicate that the Baseline algorithm maintains a comparable performance as the MBVC algorithms. Additional data were collected to understand the improving Safe% with an increasing number of agents.

Table 4.3: Each *overlap count* per type for each algorithm and the number of Crazyflies. The last column corresponds to the total inter-Crazyflie overlap count per trial, which increases with an increasing number of Crazyflies.

Algorithm	Number of UAVs	Safe (>0.75)		Partial (0.5<x <0.75)		Unsafe (0.25<x <0.5)		Collision (<0.25)	
		Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Baseline	4	1,487.80	89.51	650.08	48.2	316.96	43.21	47.16	12
	9	11,339.60	361.65	2,455.96	214.61	1,061.00	138.37	155.44	24
	15	36,585.96	678.58	4,858.76	428.43	2,043.40	218.37	296.88	47.7
	25	110,448.92	1,126.63	9,947.20	732.01	4,108.64	355.67	595.24	66.69
MBVC	4	1,497.68	97.07	645.2	54.66	311.52	43.33	47.6	13.12
	9	11,297.04	322.91	2,483.88	207.69	1,071.60	111.11	159.48	20.92
	15	36,096.44	663.27	5,180.40	422.8	2,188.32	211.69	319.84	43.2
	25	110,871.80	1,187.44	9,716.12	795.58	3,957.52	351.44	554.56	49.23
MBVC_Vel	4	1,491.32	95.89	657.4	65.24	307.48	41.32	45.8	8.56
	9	11,084.28	351.15	2,612.44	202.14	1,148.88	132.09	166.4	26.06
	15	36,400.96	716.72	4,967.48	467.82	2,113.00	220.97	303.56	38.09
	25	110,441.60	1,108.75	9,963.72	723.72	4,098.56	345.53	596.12	59.72

Table 4.4: The percentage of each type's *overlap count* per total number of Crazyflie-Crazyflie overlap counts for each algorithm by the number of Crazyflies.

Algorithm	Number of UAVs	Safe (>0.75)		Partial (0.5 <x <0.75)		Unsafe (0.25 <x <0.5)		Collision (<0.25)	
		Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Baseline	4	59.46	3.58	25.98	1.93	12.67	1.73	1.88	0.48
	9	75.54	2.41	16.36	1.43	7.07	0.92	1.04	0.16
	15	83.56	1.55	11.10	0.98	4.67	0.50	0.68	0.11
	25	88.29	0.90	7.95	0.59	3.28	0.28	0.48	0.05
MBVC	4	59.86	3.88	25.79	2.18	12.45	1.73	1.90	0.52
	9	75.25	2.15	16.55	1.38	7.14	0.74	1.06	0.14
	15	82.44	1.51	11.83	0.97	5.00	0.48	0.73	0.10
	25	88.63	0.95	7.77	0.64	3.16	0.28	0.44	0.04
MBVC_Vel	4	59.61	3.83	26.27	2.61	12.29	1.65	1.83	0.34
	9	73.84	2.34	17.40	1.35	7.65	0.88	1.11	0.17
	15	83.14	1.64	11.35	1.07	4.83	0.50	0.69	0.09
	25	88.28	0.89	7.96	0.58	3.28	0.28	0.48	0.05

Increasing the number of UAVs in the space results in a congested environment with less volume per Crazyflie to navigate freely without a collision. An increase in *number of collisions* is seen in Table 4.5 with an increasing number of agents. The MBVC algorithm performs worse than the other two algorithms with a maximum of 7 number of collisions, followed by the Baseline and MBVC_Vel for 25 Crazyflies. Even though *number of collisions* increase with an increasing number of Crazyflies, a corresponding increase in Safe *overlap count* percentages as shown in Table 4.4 suggests that the Crazyflies have enough volume around them to navigate freely improving their *completion rate*; however, this improvement was not seen.

Algorithm	Number of UAVs	Median	Max
Baseline	4	0	2
	9	0	1
	15	1	3
	25	1	**5**
MBVC	4	0	0
	9	0	2
	15	0	2
	25	1	**7**
MBVC_Vel	4	0	1
	9	0	0
	15	0	**4**
	25	1	**4**

Table 4.5: The *number of collisions* per trial for each algorithm by the number of Crazyflies. The Min value was ignored in the table because it is 0 for all the independent variables. The values in bold represent the maximum *number of collisions*.

The distance between optimal direct paths was measured to understand the experimental design and understand the increasing *overlap count*'s Safe% with an

increasing number of agents. This distance variable in Table 4.6 shows that the random generator assigned the goals such that the distance between the optimal direct paths increased with an increasing number of agents. This increase in the distance resulted in a larger free space around the UAV's optimal direct paths which resulted in a lower distance deviation to avoid collisions and indirectly increased the Safe%. The decrease in the lower distance deviation can be seen in Table 4.7, where the *net deviation* decreases with the increasing agents in the space.

Algorithm	Number of UAVs	Distance		
		Median	Min	Max
Baseline	4	0.42	0.00	3.39
	9	0.60	0.00	3.71
	15	0.74	0.00	4.29
	25	**0.88**	0.00	**4.65**
MBVC	4	0.49	0.01	3.70
	9	0.65	0.00	3.71
	15	0.71	0.00	4.12
	25	**0.92**	0.00	**4.71**
MBVC_Vel	4	0.56	0.00	3.17
	9	0.55	0.00	3.53
	15	0.72	0.00	4.11
	25	**0.86**	0.00	**4.71**

Table 4.6: The distance between the optimal direct paths per trial for each algorithm by the number of Crazyflies. The Min value corresponds to the potential collisions between the paths. The values in bold represent the maximum distance between optimal direct paths.

Collision avoidance between the Crazyflies is expected to increase the *actual path* in comparison to the *optimal direct path*. While this trend shows the *actual path* is longer in Table 4.7, there is no observable difference across the independent variable combinations. The Poisson disc sampling used to generate goal locations generates values such that the *optimal path* is independent of the number of UAVs. The *net deviation*, the area between the two paths decreased with increasing number of UAVs. This smaller area between the curves can indicate that the Crazyflies oscillate in the 3-dimensional space which results in a larger actual path lengths and decrease in the area between the two curves. These oscillations in the *actual path* can result in degraded *completion rate* for the Baseline algorithm; however, the *net deviation* do not indicate observable differences across the algorithms. Therefore, these path-based metrics do not provide insights into the degraded *completion rate* for the Baseline compared to the modified BVC algorithms; however, the decreasing *net deviation* depends on the experimental design and the path between the optimal direct paths.

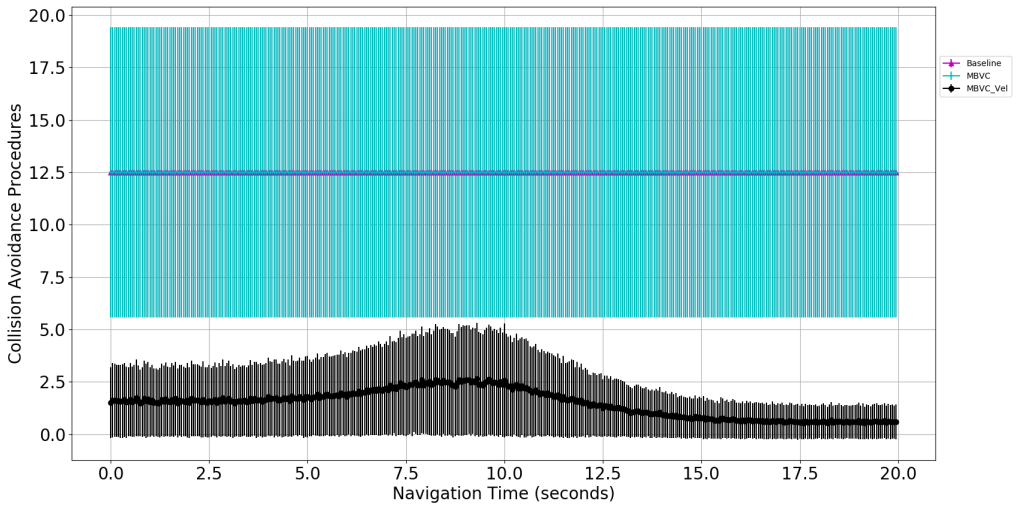
Table 4.7: The *total actual path*, *net deviation*, and *optimal direct path* per trial for each algorithm by the number of Crazyflies.

Algorithm	Number of UAVs	Actual Path			Net Deviation			Optimal Path		
		Median	Min	Max	Median	Min	Max	Median	Min	Max
Baseline	4	28.29	19.61	34.56	4.11	2.38	7.15	2.39	1.36	3.80
	9	28.65	23.18	35.69	6.16	3.67	10.52	2.51	1.09	3.90
	15	28.43	23.73	35.67	7.21	3.97	15.87	2.67	1.41	4.01
	25	28.18	22.94	35.43	2.44	1.01	7.10	2.85	1.04	4.57
MBVC	4	28.24	23.17	34.14	4.24	2.66	7.63	2.39	1.53	3.40
	9	28.37	23.32	33.76	6.16	3.85	15.12	2.50	1.27	3.95
	15	28.28	22.70	33.47	7.24	3.31	13.61	2.66	1.27	4.03
	25	28.26	22.04	35.30	2.37	1.11	7.32	2.80	1.10	4.66
MBVC_Vel	4	28.37	24.12	36.60	4.19	2.78	7.82	2.38	1.21	3.37
	9	28.43	22.96	35.57	6.05	3.88	11.73	2.47	1.27	3.74
	15	28.49	23.23	34.73	7.33	3.68	15.33	2.60	1.24	4.08
	25	28.36	23.83	33.49	2.42	1.11	6.78	2.80	1.23	4.56

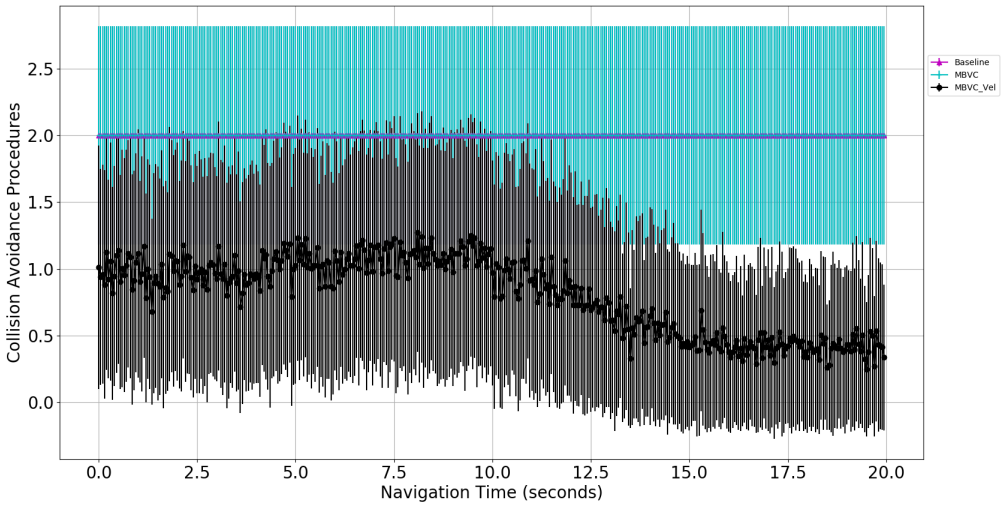
An increasing number of Crazyflies is expected to increase the total *avoidance time* and the total *collision avoidance procedure* due to the decrease in free space per Crazyflie in a given volume. The *navigation time* for a Crazyflie represents a time where it is not avoiding its peers within its range. This *navigation time* decreases with an increasing number of agents, whereas the *avoidance time* has an opposite trend as shown in 4.8. A similar increase in the total *collision avoidance procedure* was observed with an increase in the *avoidance time* in Figure 4.3; however, no differentiating results were seen between the algorithms. The number of *collision avoidance procedures* in Figure 4.3a and 4.3b suggest that the Baseline and MBVC algorithms with no range filters consider all the UAV in a given space; hence, the MBVC_Vel has a lower *collision avoidance procedures* where the Crazyflies are only actively avoiding collisions within r_{lookup} range.

Table 4.8: The total *navigation time* and the *avoidance time* of the Crazyflies per trial for each algorithm and the number of Crazyflies.

Algorithm	Number of UAVs	Avoidance Time			Navigation Time		
		Median	Min	Max	Median	Min	Max
Baseline	4	14.51	10.61	16.79	5.49	3.21	9.39
	9	16.26	6.08	19.12	3.74	0.88	13.92
	15	16.8	5.26	19.41	3.2	0.59	14.74
	25	17.78	5.16	19.71	2.22	0.29	14.84
MBVC	4	14.23	10.29	17.66	5.77	2.34	9.71
	9	16.52	10.22	19.17	3.48	0.83	9.78
	15	17.4	11.68	19.44	2.6	0.56	8.32
	25	17.67	8.38	19.7	2.33	0.3	11.62
MBVC_Vel	4	14.63	11.36	17.01	5.37	2.99	8.64
	9	16.73	11.38	19.45	3.27	0.55	8.62
	15	17.07	8.05	19.4	2.93	0.6	11.95
	25	17.66	7.83	19.67	2.34	0.33	12.17



(a) Number of UAVs = 25

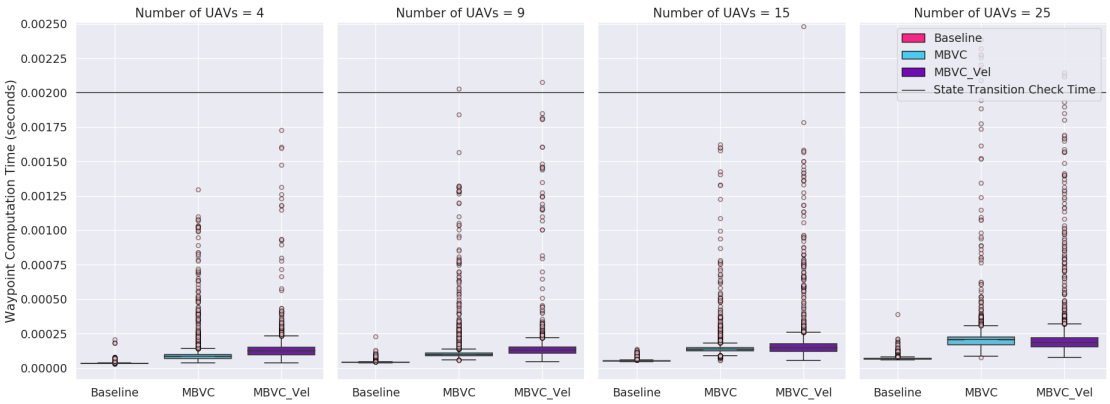


(b) Number of UAVs = 4

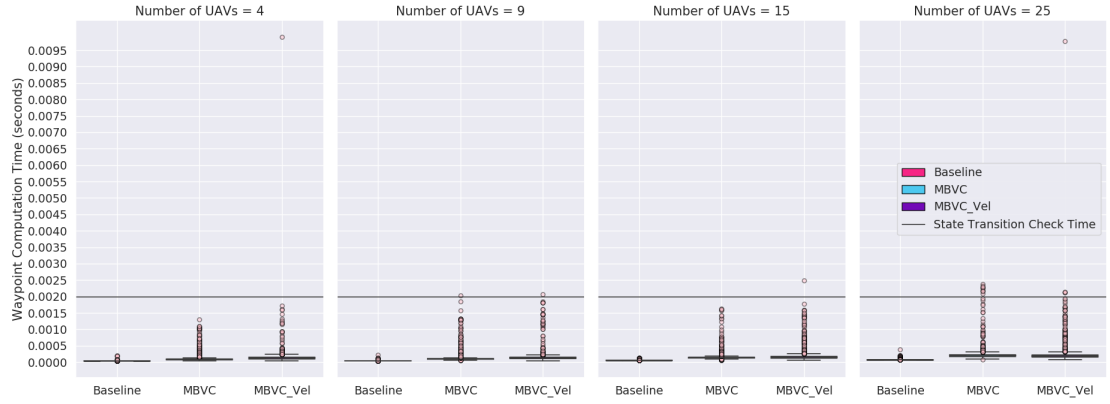
Figure 4.3: The *collision avoidance procedures* for each algorithm for (a) 25 UAVs and (b) 4 UAVs provided in the line plot with error bars. The Baseline and MBVC consider all the agents for the *collision avoidance procedures* computation and have overlapping mean and error bar lines.

4.4.2 Time Complexity-based Results

The number of BVC constraints increases with an increase in the number of UAVs; hence, the *waypoint computation time* metric was incorporated to determine the time complexity of the algorithms with increasing agents. All three algorithms generate a waypoint within 0.002 seconds for the vast majority of trials as shown in Figure 4.4. A few outliers are present for the optimization-based algorithms, MBVC and MBVC_Vel. An increasing number of UAVs is expected to increase the overall waypoint computation time for both optimization-based modified BVC algorithms; however, the range filter-based MBVC_Vel is expected to be efficient when compared to the MBVC as seen for 25 Crazyflies. The Baseline design allows for a steady performance with an increasing number of UAVs.



(a) After removing outliers.



(b) Including all the outliers.

Figure 4.4: The *waypoint computation time* for each algorithm and a number of Crazyflies is provided in the box and whisker plots. Pink markers outside the range represent the outliers. The reference line (Black) represents Crazyflie's update rate.

4.5 Discussion

UAVs act as dynamic obstacles to each other in this project that seeks to explore whether the multiple small-sized UAVs' collision avoidance capabilities are affected by the positioning noise introduced by LPS. The algorithms' performance is analyzed using distance-based metrics to assess the capability of timely navigation command generation with increasing UAVs.

The *completion rate* indicated that trials with MBVC and MBVC_Vel performed better in the presence of LPS-induced noise. This improvement demonstrated that the UAVs reached their goal with a tighter tolerance within the deadline; however, the remaining metrics did not provide any insights as to why this was the case. The hypothesis **H1** expected that the optimization-based modified BVC will generate effective paths for collision avoidance and this was supported; however, a comparative improvement in the performance was not found. The performance of all three algorithms remains consistent with increasing agents and up to 4 agents failing to satisfy the completion rate rule while reaching their goals. Computing *collision avoidance procedures* suggested that the MBVC_Vel algorithm has a lower value than the other algorithms, which helps this algorithm improves its priority to reach the goal resulting in a higher completion rate for MBVC_Vel.

The mean percentage of Partial and Safe *overlap count* related to the inter-UAV coordination contradicted **H2** when employing the MBVC or MBVC_Vel algorithms, where the combined Safe and Partial *overlap count* will always ex-

hibit a higher value than the Baseline and be greater than 80%. While MBVC and MBVC_Vel counts were higher than 80%, there was no observable difference between the Baseline and these algorithms. The count categories results did not reflect an association towards the slight successful *completion rate*. The distance between the optimal direct paths provided insights into the limitations of the experimental design. The random generator for goal position assignment resulted in scenarios such that with an increasing number of agents, the distance between the optimal direct paths increased. This increase resulted in a larger inter-Crazyflie distance and a lower deviation from their optimal direct path to reach their goal locations.

The range-based obstacle filter aimed to improve the computational complexity of the MBVC_Vel algorithm compared to MBVC by introducing the range r_{lookup}. Hypothesis **H3** was developed based on the assumption that this range filtration will consistently allow the MBVC_Vel to generate a waypoint within 0.002 seconds. This small computation time facilitates the detection of real-time changes and updates in the space while ensuring safety with minimal delay. Generally, this hypothesis was supported; however, there were cases where the filtration design resulted in an outlier. An increasing number of UAVs is expected to increase the overall waypoint computation time for both optimization-based modified BVC algorithms; however, the range filter-based MBVC_Vel is seen to be efficient compared to the MBVC for 25 Crazyflies.

The modified BVC algorithms generated navigation commands in a timely manner; however, the expected substantial improvements over the Baseline were not

observed. Despite the absence of a noticeable improvement in the distance-based metrics, the Crazyflies with the modified BVC algorithms were more successful in reaching their goals with a tight distance tolerance. Additionally, the integrated physics simulator also enabled the Crazyflie's controllers with the Crazyswarm API to contribute to the development of small-sized UAVs.

Chapter 5: Conclusion

This project developed and integrated a modified BVC for the Crazyflie quadcopters and validated the algorithm in a simulated Crazyswarm platform, which incorporates the gym-pybullet-drones physics simulator. The first contribution of this project is the algorithm formulation as an optimization problem subjected to uncertainty-aware and acceleration buffer constraints to improve the algorithm's resilience against positioning uncertainty. The Crazyflies' baseline algorithm was developed for a motion capture positioning system, whereas the modified algorithm incorporates the standard deviation of the positioning noise from LPS to enhance the Crazyflies' collision avoidance ability. Furthermore, reciprocal velocity obstacle constraints introduced in the algorithm estimate the peer obstacles' velocity to enhance collision avoidance with dynamic obstacles. These design modifications improve the low-level autonomy of the Crazyflies and their extension to multiple quadcopter applications in indoor environments with LPS enabling a probabilistic collision-free guarantee promised by the BVC design. A range filter for obstacle detection is also introduced in the algorithm to limit the number of constraints as the number of Crazyflies increase. This design modification ensures efficiency in time complexity and generates a waypoint before each system update rate. The Crazyflie's existing state machine's update rate is 500 Hz and the waypoint is generated within 0.002 seconds for more than 99% of the simulation instances.

Distance-based metrics were introduced for analyzing the collision avoidance capabilities for multiple quadcopter scenarios. This contribution provides a comprehensive evaluation of the system's performance by utilizing the inter-UAV distance categorizations and the distance deviation from the *optimal direct path* introduced in Chapter 4.2. The incorporation of reciprocal velocity constraints was to improve the collision avoidance responsibility associated with the measure of distance-based metrics; however, the findings for this measure did not demonstrate the observable improvement in the modified BVC algorithms. Crazyflies equipped with the baseline algorithm had slightly lower success in reaching the goal location with a tight distance tolerance of 0.3 meters; however, this slightly successful measure for the modified BVC algorithms did not reflect an association towards the inter-UAV distance categorization throughout the trials.

The last contribution was the enhancement of the existing Crazyflie simulator to enable comprehensive testing and validation for the multiple quadcopter scenario. The baseline algorithm was initially developed and tested on a Crazyswarm platform that bypasses the Crazyflie's controller and second-order dynamic motion. This lack of a comprehensive and lightweight physics simulator for Crazyflies necessities a testing platform that addresses these gaps and can easily interact with the existing Crazyswarm API to support the development of Crazyflies. The results generated across the trials primarily showed that the modified BVC maintained its performance irrespective of the number of Crazyflies in the shared space. Overall, this project contributed to the development of the modified BVC algorithm and validated its performance in custom simulation worlds using distance-based

metrics introduced for multiple quadcopter system evaluations.

5.1 Future Work

This project implemented a modified buffered Voronoi and reciprocal velocity-based collision avoidance algorithm for Crazyflies on a physics simulator, but there are several avenues for future work that can enhance the Crazyflie system's capabilities. The next step will involve evaluating this algorithm on real Crazyflies by overcoming hardware limitations (e.g., delayed information exchange due to Peer to Peer communication design, excessive LPS Z directional noise near static obstacles due to non-line-of-sight reflections). Additional experiments improvising the velocity assignment of the Crazyflies moving towards their waypoint, the addition of static obstacles apart from the testbed bounding boundary, and the introduction communication gap in Peer to Peer information exchange can provide valuable insight into the algorithm's application to real-world scenarios.

The current state of collision avoidance on the Crazyflies does not allow deadlock resolution. The current algorithm assumes that the noise present in the localization module does not result in a deadlock configuration. Incorporating a deadlock resolution will improve the algorithm's reliability to traverse to its goal location with an increasing number of Crazyflies in a given space. Furthermore, the algorithm can be expanded to incorporate the yaw angle into the waypoint generation process. Incorporating the attitude controller instead of always keeping the yaw angle at 0 can allow Crazyflies to navigate in the 3-dimensional space and

handle more complex maneuvering scenarios. Further calculations specific to the Navigation mode's Avoidance Time categories including the Hover Time and the Collision Time combined with the help of an improved simulation log data structure can provide information to differentiate the oscillations in the quadcopter's trajectory from hovering.

Lastly, the implemented algorithm can act as a foundation for training reinforcement learning models for Crazyflie's that are gaining popularity in the field of small-sized quadcopter research. More complex control strategies can be learned from their environment for multiple Crazyflie applications using the modified buffered Voronoi cell approach as a baseline. Addressing these aspects can improve the autonomous capabilities of small-sized quadcopters and enable their use in a wider range of real-world applications.

Bibliography

[1] D. Coldewey. "DroneSeed's $36M A round makes it a one-stop shop for post-wildfire reforestation." TechCrunch, Ed. (Sep. 2021), [Online]. Available: https://techcrunch.com/2021/09/29/droneseeds-36m-a-round-makes-it-a-one-stop-shop-for-post-wildfire-reforestation/ (visited on 09/11/2022).

[2] S. Cagle. "'Dragon' drones: The flame throwers fighting wildfires with fire." T. Gaurdian, Ed. (Sep. 2019), [Online]. Available: https://www.theguardian.com/us-news/2019/sep/03/wildfires-drones-controlled-prescribed-burns (visited on 09/11/2022).

[3] J. G. A. Barbedo, "A review on the use of unmanned aerial vehicles and imaging sensors for monitoring and assessing plant stresses," *Drones*, vol. 3, no. 2, p. 40, 2019.

[4] D. A. Edward G. Keating John Kerman and D. Mosher, "Usage patterns and costs of unmanned aerial systems," Congressional Budget Office, 2021. [Online]. Available: https://www.cbo.gov/publication/57260 (visited on 03/03/2023).

[5] K. Chan, U Nirmal, and W. Cheaw, "Progress on drone technology and their applications: A comprehensive review," in *Conference Proceedings*, AIP Publishing LLC, vol. 2030, 2018, p. 020 308.

[6] Bitcraze, *Crazyflie 2.1*. [Online]. Available: `http://bitcraze.io/products/crazyflie-2-1/` (visited on 12/19/2022).

[7] R. da Silva Tchilian, U. F. Moreno, and M. Netto, "Assisted teleoperation for a human-swarm interaction system," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 602–607, 2020.

[8] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (uavs): Practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, pp. 1–29, 2023.

[9] I. Fadelli, *A new approach to improve robot navigation in crowded environments*, Mar. 2023. [Online]. Available: `https://techxplore.com/news/2023-02-approach-robot-crowded-environments.html` (visited on 12/09/2022).

[10] Z. Goraj, *Current trends in aircraft design (7th EASN): Aircraft Engineering and Aerospace Technology*. Emerald Publishing Limited, 2019.

[11] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[12] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *International Conference on Methods and Models in Automation and Robotics*, IEEE, vol. 22, 2017, pp. 37–42.

[13] M. Macchini, T. Havy, A. Weber, F. Schiano, and D. Floreano, "Hand-worn haptic interface for drone teleoperation," in *International Conference on Robotics and Automation*, IEEE, 2020, pp. 10 212–10 218.

[14] R. Hadidi, B. Asgari, S. Jijina, A. Amyette, N. Shoghi, and H. Kim, "Quantifying the design-space tradeoffs in autonomous drones," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, vol. 26, Association for Computing Machinery, 2021, pp. 661–673.

[15] J. E. Noronha, "Development of a swarm control platform for educational and research applications," Ph.D. dissertation, Iowa State University, Iowa, United States, 2016.

[16] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[17] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *Communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[18] M. A. Lopez, M. Baddeley, W. T. Lunardi, A. Pandey, and J.-P. Giacalone, *Towards secure wireless mesh networks for uav swarm connectivity: Current threats, research, and opportunities*, 2021. arXiv: 2108.13154 [cs.NI].

[19] K. Richardsson, Barbara, Marcus, *et al.*, *Category: Loco positioning*, 2021. [Online]. Available: https://www.bitcraze.io/category/loco-positioning/ (visited on 12/19/2022).

[20] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on various inertial measurement unit (imu) sensor applications," *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[21] K. Saadeddin, M. F. Abdel-Hafez, and M. A. Jarrah, "Estimating vehicle state by gps/imu fusion with vehicle dynamics," *Journal of Intelligent & Robotic Systems*, vol. 74, pp. 147–172, 2014.

[22] G. Balamurugan, J Valarmathi, and V. Naidu, "Survey on uav navigation in gps denied environments," in *International conference on signal processing, communication, power and embedded system*, IEEE, 2016, pp. 198–204.

[23] N. Kumari, R. Kulkarni, M. R. Ahmed, and N. Kumar, "Use of kalman filter and its variants in state estimation: A review," *Artificial Intelligence for a Sustainable Industry 4.0*, pp. 213–230, 2021.

[24] E. Ackerman, *The secret to small drone obstacle avoidance is to just crash into stuff*, 2016. [Online]. Available: https://spectrum.ieee.org/the-secret-to-small-drone-obstacle-avoidance-is-to-just-crash-into-stuff (visited on 03/03/2023).

[25] E. D'Amato, I. Notaro, and M. Mattei, "Reactive collision avoidance using essential visibility graphs," in *International Conference on Control, Decision and Information Technologies*, IEEE, vol. 6, 2019, pp. 522–527.

[26] Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, and V. Kumar, "Robust aerial robot swarms without collision avoidance," *Robotics and Automation Letters*, vol. 3, no. 1, pp. 596–603, 2017.

[27] J. Burgués, V. Hernández, A. J. Lilienthal, and S. Marco, "Smelling nano aerial vehicle for gas source localization and mapping," *Sensors*, vol. 19, no. 3, p. 478, Jan. 2019.

[28] S. Feng, Y. Zhang, S. E. Li, Z. Cao, H. X. Liu, and L. Li, "String stability for vehicular platoon control: Definitions and analysis methods," *Annual Reviews in Control*, vol. 47, pp. 81–97, 2019.

[29] D. Palossi, F. Conti, and L. Benini, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs," in *International Conference on Distributed Computing in Sensor Systems*, IEEE, vol. 15, 2019, pp. 604–611.

[30] Z. Ullah, Z. Xu, L. Zhang, L. Zhang, and W. Ullah, "Rl and ann based modular path planning controller for resource-constrained robots in the indoor complex dynamic environment," *IEEE Access*, vol. 6, pp. 74 557–74 568, 2018.

[31] *Crazyswarm discussion #644*. [Online]. Available: `https://github.com/USC-ACTLab/crazyswarm/discussions/644` (visited on 12/19/2022).

[32] G. Gov, "Gps standard positioning service (sps) performance standard," *GPS. Gov Website.*, vol. 4, pp. 43–58, 2015. (visited on 01/04/2023).

[33] A. Theiss, D. C. Yen, and C.-Y. Ku, "Global positioning systems: An analysis of applications, current development and future implementations," *Computer Standards & Interfaces*, vol. 27, no. 2, pp. 89–100, 2005.

[34] A. Famili, A. Stavrou, H. Wang, and J.-M. J. Park, "Pilot: High-precision indoor localization for autonomous drones," *Transactions on Vehicular Technology*, pp. 1–15, 2022.

[35] Y. Li, M. Scanavino, E. Capello, F. Dabbene, G. Guglieri, and A. Vilardi, "A novel distributed architecture for uav indoor navigation," *Transportation research procedia*, vol. 35, pp. 13–22, 2018.

[36] M. Pérez, D Gualda, J Vicente, J. Villadangos, and J Ureña, "Review of uav positioning in indoor environments and new proposal based on us measurements," in *Central Europe Workshop Proceedings*, vol. 2498, 2019, pp. 267–274.

[37] Y. Lee, J. Yoon, H. Yang, C. Kim, and D. Lee, "Camera-gps-imu sensor fusion for autonomous flying," in *International Conference on Ubiquitous and Future Networks*, IEEE, vol. 8, 2016, pp. 85–88.

[38] C. Wang, T. Wang, J. Liang, Y. Chen, Y. Zhang, and C. Wang, "Monocular visual slam for small uavs in gps-denied environments," in *International Conference on Robotics and Biomimetics*, IEEE, 2012, pp. 896–901.

[39] O. Amidi, T. Kanade, and R. Miller, "Vision-based autonomous helicopter research at carnegie mellon robotics institute (1991-1998)," 2000.

[40] I. Pavlidis and P. Symosek, "The imaging issue in an automatic face/disguise detection system," in *Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications (Cat. No. PR00640)*, IEEE, 2000, pp. 15–24.

[41] H. Muller, V. Niculescu, T. Polonelli, M. Magno, and L. Benini, "Robust and efficient depth-based obstacle avoidance for autonomous miniaturized uavs," 2022. arXiv: 2208.12624 [cs.RO].

[42] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, *et al.*, "A scalable framework for real-time multi-robot, multi-human collision avoidance," in *International conference on robotics and automation*, IEEE, 2019, pp. 936–943.

[43] K. Loayza, P. Lucas, and E. Peláez, "A centralized control of movements using a collision avoidance algorithm for a swarm of autonomous agents," in *Second Ecuador Technical Chapters Meeting*, IEEE, 2017, pp. 1–6.

[44] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[45] H. Zhu, B. Brito, and J. Alonso-Mora, "Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells," *Autonomous Robots*, vol. 46, no. 2, pp. 401–420, 2022.

[46] M. Field, D. Stirling, F. Naghdy, and Z. Pan, "Motion capture in robotics review," in *International conference on control and automation*, IEEE, 2009, pp. 1697–1702.

[47] P. Nogueira, "Motion capture fundamentals," in *Doctoral Symposium in Informatics Engineering*, vol. 303, 2011.

[48] Wikipedia, *Motion capture — Wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Motion\%20capture&oldid=1139895845`, 2023. (visited on 12/19/2022).

[49] A Masiero, F Fissore, R Antonello, A Cenedese, and A Vettore, "A comparison of uwb and motion capture uav indoor positioning," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 1695–1699, 2019.

[50] A. Masiero, F. Fissore, and A. Vettore, "A low cost uwb based solution for direct georeferencing uav photogrammetry," *Remote Sensing*, vol. 9, no. 5, p. 414, 2017.

[51] *Primex-22 - buy.* [Online]. Available: `https://optitrack.com/cameras/primex-22/buy.html` (visited on 03/03/2023).

[52] M. Kolakowski and V. Djaja-Josko, "Tdoa-twr based positioning algorithm for uwb localization system," in *International Conference on Microwave, Radar and Wireless Communications*, IEEE, vol. 21, 2016, pp. 1–4.

[53] M. Stocker, B. Großwindhager, C. A. Boano, and K. Römer, "Towards secure and scalable uwb-based positioning systems," in *International Conference on Mobile Ad Hoc and Sensor Systems*, IEEE, vol. 17, 2020, pp. 247–255.

[54] P. Grasso, M. S. Innocente, J. J. Tai, O. Haas, and A. M. Dizqah, "Analysis and accuracy improvement of uwb-tdoa-based indoor positioning system," *Sensors*, vol. 22, no. 23, p. 9136, 2022.

[55] H.-Y. Lin and J.-R. Zhan, "Gnss-denied uav indoor navigation with uwb incorporated visual inertial odometry," *Measurement*, vol. 206, p. 112 256, 2023.

[56] C. Chadehumbe and J. Sjöberg, "Autonomous flight of the micro drone crazyflie 2.1 through an obstacle course," Ph.D. dissertation, Uppsala Universitet, Uppsala, Sweden, 2020, p. 36.

[57] S. Chaumette, D. A. G. Jáuregui, S. Bottecchia, and N. Couture, "Issues of indoor control of a swarm of drones in the context of an opera directed by a soundpainter," in *International Workshop on Human-Drone Interaction*, vol. 1, 2019.

[58] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[59] E. Gat, R. P. Bonnasso, R. Murphy, *et al.*, "On three-layer architectures," *Artificial intelligence and mobile robots*, vol. 195, p. 210, 1998.

[60] R. C. Arkin and T. Balch, "Aura: Principles and practice in review," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 175–189, 1997.

[61] R. Hartley and F. Pipitone, "Experiments with the subsumption architecture," in *International Conference on Robotics and Automation*, IEEE, 1991, pp. 1652–1658.

[62] J. H. Connell *et al.*, "Sss: A hybrid architecture applied to robot navigation.," in *International Conference on Robotics and Automation*, IEEE, vol. 3, 1992, pp. 2719–2724.

[63] A Sankaranarayanan and I. Masuda, "A new algorithm for robot curve-following amidst unknown obstacles, and a generalization of maze-searching," in *International Conference on Robotics and Automation*, IEEE Computer Society, 1992, pp. 2487–2488.

[64] I. Kamon, E. Rimon, and E. Rivlin, "Tangentbug: A range-sensor-based navigation algorithm," *The International Journal of Robotics Research*, vol. 17, no. 9, pp. 934–953, 1998.

[65] M. Caccia, R. Bono, G. Bruzzone, and G. Veruggio, "Variable-configuration uuvs for marine science applications," *Robotics & Automation Magazine*, vol. 6, no. 2, pp. 22–32, 1999.

[66] M. Wang, Z. Su, D. Tu, and X. Lu, "A hybrid algorithm based on artificial potential field and bug for path planning of mobile robot," in *International*

Conference on Measurement, Information and Control, IEEE, vol. 2, 2013, pp. 1393–1398.

[67] K. N. McGuire, G. C. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robotics and Autonomous Systems*, vol. 121, p. 103 261, 2019.

[68] Z. Liu, X. Wang, and K. Li, "Research on path planning of multi-rotor uav based on improved artificial potential field method," in *Materials science, Engineering and Chemistry Web of Conferences*, EDP Sciences, vol. 336, 2021, p. 07 006.

[69] D. Maravall, J. De Lope, and J. P. Fuentes, "Navigation and self-semantic location of drones in indoor environments by combining the visual bug algorithm and entropy-based vision," *Frontiers in neurorobotics*, vol. 11, p. 46, 2017.

[70] A. A. Masoud, "Kinodynamic motion planning," *Robotics & Automation Magazine*, vol. 17, no. 1, pp. 85–99, 2010.

[71] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *International Conference on Robotics and Automation*, IEEE, vol. 2, 1985, pp. 500–505.

[72] S. G. Loizou and K. J. Kyriakopoulos, "Navigation of multiple kinematically constrained robots," *Transactions on Robotics*, vol. 24, no. 1, pp. 221–231, 2008.

[73] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative uavs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18 382–18 390, 2017.

[74] L. Lifen, S. Ruoxin, L. Shuandao, and W. Jiang, "Path planning for uavs based on improved artificial potential field method through changing the repulsive potential function," in *Chinese Guidance, Navigation and Control Conference*, IEEE, 2016, pp. 2011–2015.

[75] E. Wu, Y. Sun, J. Huang, C. Zhang, and Z. Li, "Multi uav cluster control method based on virtual core in improved artificial potential field," *IEEE Access*, vol. 8, pp. 131 647–131 661, 2020.

[76] D. M. K. K. V. Rao, H. Habibi, J. L. Sanchez-Lopez, and H. Voos, *An integrated real-time uav trajectory optimization with potential field approach for dynamic collision avoidance*, 2023. arXiv: 2303.02043 [cs.RO].

[77] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, pp. 207–222, 2002.

[78] J. Ren, K. A. McIsaac, and R. V. Patel, "Modified newton's method applied to potential field-based navigation for nonholonomic robots in dynamic environments," *Robotica*, vol. 26, no. 1, pp. 117–127, 2008.

[79] A. Marif, W. Rahmaniar, M. A. M. Vera, A. A. Nuryono, R. Majdoubi, and A. Çakan, "Artificial potential field algorithm for obstacle avoidance in uav quadrotor for dynamic environment," in *International Conference on Communication, Networks and Satellite*, IEEE, 2021, pp. 184–189.

[80] O. Cetin, I. Zagli, and G. Yilmaz, "Establishing obstacle and collision free communication relay for uavs with artificial potential fields," *Journal of Intelligent & Robotic Systems*, vol. 69, pp. 361–372, 2013.

[81] R. M. J. A. Souza, G. V. Lima, A. S. Morais, L. C. Oliveira-Lopes, D. C. Ramos, and F. L. Tofoli, "Modified artificial potential field for the path planning of aircraft swarms in three-dimensional environments," *Sensors*, vol. 22, no. 4, p. 1558, 2022.

[82] Y. Du, X. Zhang, and Z. Nie, "A real-time collision avoidance strategy in dynamic airspace based on dynamic artificial potential field algorithm," *IEEE Access*, vol. 7, pp. 169 469–169 479, 2019.

[83] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, eabg5810, 2021.

[84] V. Niculescu, L. Lamberti, F. Conti, L. Benini, and D. Palossi, "Improving autonomous nano-drones performance via automated end-to-end optimization and deployment of dnns," *Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 4, pp. 548–562, 2021.

[85] D. Wang, W. Li, X. Liu, N. Li, and C. Zhang, "Uav environmental perception and autonomous obstacle avoidance: A deep learning and depth camera combined solution," *Computers and Electronics in Agriculture*, vol. 175, p. 105 523, 2020.

[86] A. Rodionova, Y. V. Pant, C. Kurtz, K. Jang, H. Abbas, and R. Mangharam, "Learning-and-flying: A learning-based, decentralized mission-aware uas collision avoidance scheme," *Transactions on Cyber-Physical Systems*, vol. 5, no. 4, pp. 1–26, 2021.

[87] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[88] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. de Croon, "Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone," *Journal of Field Robotics*, vol. 37, no. 4, pp. 667–692, 2020.

[89] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, *Uncertainty-aware reinforcement learning for collision avoidance*, 2017. arXiv: `1702.01182 [cs.LG]`.

[90] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *Robotics and automation letters*, vol. 5, no. 3, pp. 4249–4256, 2020.

[91] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[92] R. Ourari, K. Cui, A. Elshamanhory, and H. Koeppl, "Nearest-neighbor-based collision avoidance for quadrotors via reinforcement learning," in *International Conference on Robotics and Automation*, IEEE, 2022, pp. 293–300.

[93] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *International Conference on Intelligent Robots and Systems*, IEEE, 2021, pp. 7512–7519.

[94] M. Vitus, V. Pradeep, G. Hoffmann, S. Waslander, and C. Tomlin, "Tunnel-milp: Path planning with sequential convex polytopes," in *Guidance, navigation and control conference and exhibit*, AIAA, 2008, p. 7132.

[95] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International journal of robust and nonlinear control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[96] S. H. Arul and D. Manocha, "Dcad: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *Robotics and Automation Letters*, vol. 5, no. 2, pp. 1191–1198, 2020.

[97] J. Shin and H. J. Kim, "Nonlinear model predictive formation flight," *Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 5, pp. 1116–1125, 2009.

[98] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear mpc for collision avoidance and control of uavs with dynamic

obstacles," *Robotics and automation letters*, vol. 5, no. 4, pp. 6001–6008, 2020.

[99] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.

[100] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Systems*, vol. 7, no. 01, pp. 55–64, 2019.

[101] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, "Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty," in *International Conference on Intelligent Robots and Systems*, IEEE, 2017, pp. 1089–1096.

[102] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *International conference on robotics and automation*, IEEE, 2008, pp. 1928–1935.

[103] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," in *International Conference on Robotics and Automation (Cat. No. 01CH37164)*, IEEE, vol. 4, 2001, pp. 3716–3721.

[104] A. L. Smith and F. G. Harmon, "Uas collision avoidance algorithm based on an aggregate collision cone approach," *Journal of Aerospace Engineering*, vol. 24, no. 4, pp. 463–477, 2011.

[105] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[106] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *International Conference on Robotics and Automation*, IEEE, 2011, pp. 3475–3482.

[107] J. Gao, H. Zhang, L. Tan, and X. Ren, "Uav dynamic obstacle avoidance based on improved reciprocal velocity obstacle," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2216, 2022, p. 012 014.

[108] S. H. Arul and D. Manocha, "V-rvo: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *International Conference on Intelligent Robots and Systems*, IEEE, 2021, pp. 8097–8104.

[109] K. Motonaka and S. Miyoshi, "Obstacle avoidance using buffered voronoi cells based on local information from a laser range scanner," *Advanced Robotics*, pp. 1–14, 2022.

[110] H. Zhu and J. Alonso-Mora, "B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance," in *International symposium on multi-robot and multi-agent system*, IEEE, 2019, pp. 162–168.

[111] A. Pierson, W. Schwarting, S. Karaman, and D. Rus, "Weighted buffered voronoi cells for distributed semi-cooperative behavior," in *International conference on robotics and automation*, IEEE, 2020, pp. 5611–5617.

[112] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[113] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[114] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *International conference on robotics and automation*, IEEE, 2008, pp. 1928–1935.

[115] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *International Conference on Robotics and Automation*, IEEE, 2017, pp. 3299–3304.

[116] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[117] S. G. Johnson, *The NLopt nonlinear-optimization package*, `https://github.com/stevengj/nlopt`, 2007. (visited on 12/19/2022).

[118] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*, ser. Mathematics and Its Applications, S. Gomez and J.-P. Hennart, Eds., vol. 275, Springer, 1994, pp. 51–67.

[119] T. Snyder, "Implementation of distributed localized communications," B.S. thesis, Oregon State University, Oregon, United States of America, 2022.

[120] A. Taffanel, *Add functionality to allow the stm32 to send/receive p2p radio packets issue #29 bitcraze/crazyflie2-nrf-firmware*, 2019. [Online]. Available: `https://github.com/bitcraze/crazyflie2-nrf-firmware/issues/29` (visited on 03/03/2023).

[121] D. Ghosh and A. Vogt, "Sampling methods related to bernoulli and poisson sampling," in *Proceedings of the Joint Statistical Meetings*, American Statistical Association, 2002, pp. 3569–3570.