LOW-LATENCY AIRBORNE COLLISION DETECTION AND AVOIDANCE SYSTEM FOR UNMANNED AIRCRAFT SYSTEMS IN A VARYING ENVIRONMENT

by

Hongru Li

A thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Canada

ABSTRACT

Detect-and-avoid (DAA) systems equipped with non-cooperative sensors for unmanned aircraft systems (UAS) beyond visual line-of-sight (BVLOS) operation is of interest to many researchers and industries today. Planning trajectories for UAS DAA in real-time requires fast and efficient trajectory generation algorithms.

This thesis presents an airborne radar based non-cooperative DAA system that aims to detect moving obstacles such as aircraft and birds within the collision avoidance threshold and generate the desired trajectory as a resolution advisory for a pilot and UAS control commands.

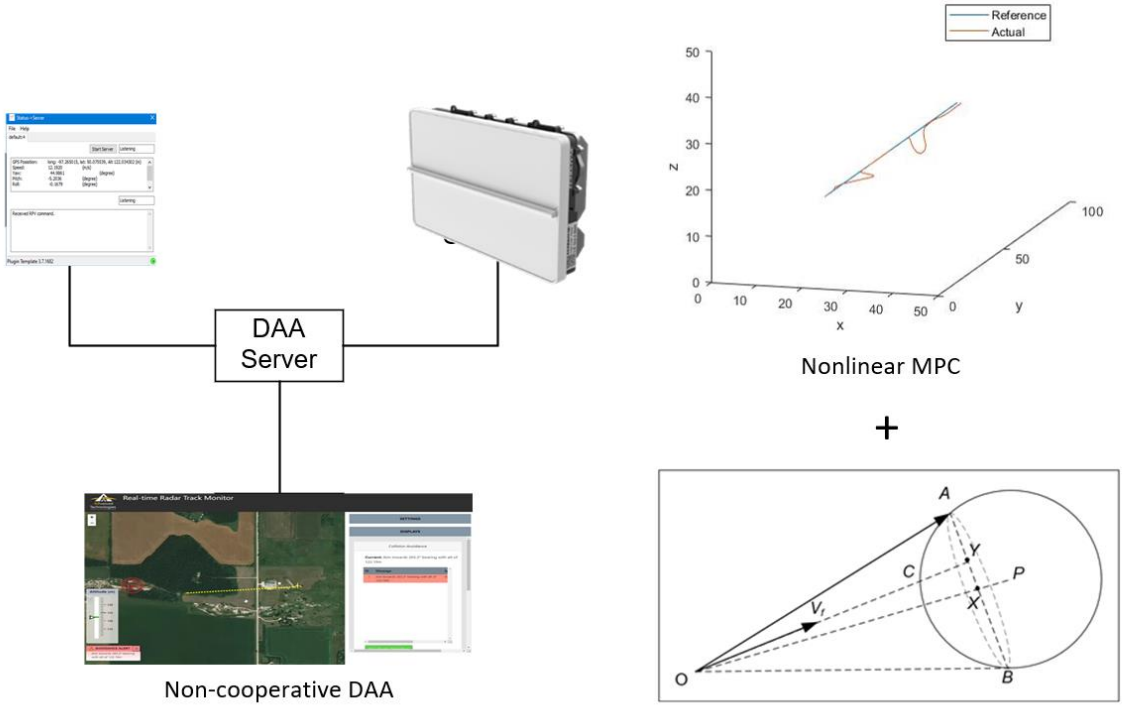
For collision detection, coordinate system transformation with quaternions is applied to process and display the obstacles detected by the airborne radar in the world frame as well as to construct a simulated airborne radar for the software-in-the-loop (SWIL) DAA system simulation.

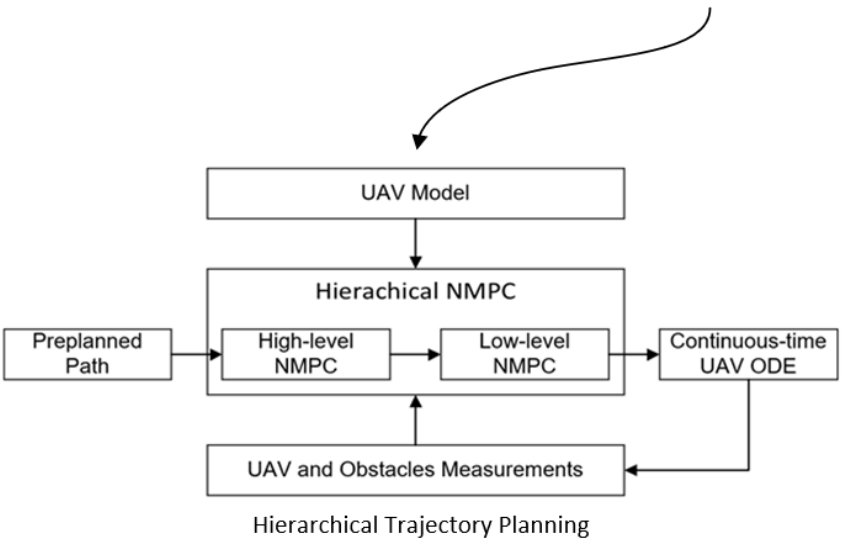
First, a reactive collision cone approach is presented for collision avoidance. In particular, software-in-the-loop simulation with test vectors from the minimum operational performance standards (MOPS) for DAA systems are used for the validation and verification of the system. Then, a non-linear model predictive control (NMPC) that utilizes the differential flatness of the UAV is presented to generate dynamically feasible collision-free trajectories in real-time. In order to overcome the limited performance of this NMPC approach due to the non-convexity of the solution space and high computational complexity, a hierarchical architecture is designed and presented. The efficiency of this approach is achieved by making use of: (i) the idea of global and local planners for the decomposition of trajectory planning and tracking, (ii) the fast analytic collision cone technique for path planning, and (iii) efficient generation of trajectories with minimum snaps as initial guesses for a low-level motion planner.

This new DAA system is distributed over a local area network (LAN), which incorporates various design patterns and principles, including multi-threaded object orientation, command query segregation, finite state machine, reader-writer locks, and heartbeat event to address: (i)

future demand for on-board and on-cloud computing, (ii) expandability to other unmanned aircraft traffic management (UTM) services, and (iii) system operation and monitoring.

VISUAL ABSTRACT





Hierarchical Trajectory Planning

ACKNOWLEDGMENTS

I would like to thank to my advisor, Professor Witold Kinsner, for his support and guidance. This journey is more than just the thesis, and it would never be the same without his humanity and wisdom.

Special thanks go to my friends Yan Wang and Siobhan Reid with whom I shared many research discussions and collaborations throughout the years.

Thanks also go to all previous and current members of the Cognitive Systems Group: Vinh Vu, Maryam Ghanbari, Kathryn Marcynuk, and Jesus Terrazas Gonzalez with whom I learned, discussed, and shared ideas.

I would also like to thank the current and former members of Aurora Aerial Inc. and AIRvanced Technologies Inc. who supported and collaborated in various parts of this thesis, including Alan Tay, Hongbo Zhao, Skylar Greenslade, Brenn Palma, Jiqing Dai, and Kaiyue Lu.

I would like to thank all my friends for bringing me joy and encouragement during the pandemic.

It is my honour to thank Dr. Robert McLeod and Dr. Nariman Sepehri for providing me with valuable comments on model predictive control and real-time computing and for their time and effort when reading this thesis.

Above all, I would like to thank my wife, Zhou Zhou, for all her support and love throughout the journey. Thank you to my parents Xi and Xiaohua for their encouragement and selfless support at each stage of my life. I owned you everything.

The research was funded in part by Mitacs and AIRvanced Technologies Inc. through Mitacs Accelerate program.

TABLE OF CONTENTS

DEFINITIONS

Above ground level	The distance that an object is above the underlying ground surface.
Airborne radar	A small-sized radar sensor that can be installed on aircraft.
Air traffic management	An aviation term encompasses all systems that assist aircraft to depart from an aerodrome, transit airspace, and land at a destination aerodrome.
Beyond visual line-of-sight	A beyond visual line-of-sight operation in which the remote crew losses direct visual contact with the aircraft to manage its flight and meet separation and collision avoidance responsibilities.
Collision avoidance	Extreme maneuvers of unmanned aircraft just prior to the closest point of approach to prevent collision in cases where safe separation is lost [1, p. 1].
Collision detection	A computational problem of detecting the intersection of two or more objects. In this thesis, it refers to the detection of potential collisions between the own aircraft and intruders.
Control horizon	The control horizon is the section of the time horizon where manipulated variable moves are allowed.
Cooperative sensor	A cooperative sensor that is on board an aircraft allows interrogation and/or broadcast of information.
Datalink	In aviation, a datalink is the hardware system that provides a wireless node-to-node data transfer. It is commonly used for communication between the aircraft and the ground control station.
Detect-and-avoid	The capability to see, sense or detect conflicting traffic or other hazards and take the appropriate action to comply with the applicable rules of flight.
Discretization steps	The number of the time steps for the optimal control problem. In this thesis, the MATLAB non-linear model predictive control (MPC) controller automatically discretizes the continuous time model using the built-in implicit trapezoidal rule with a sample time of 1 second. Since the presented MPC simulation is a batch

	program, a computation of the optimal control action at each simulation time step is proposed. The MPC sample time is set to be slightly larger than the maximum latency of itself to achieve a consistent sample time of the simulation and that of MPC. The optimal choice of the discretization steps, or equivalently the sample time is, a balance of performance and computation cost.
Dynamic system	A dynamic system is a system or process in which motion occurs, or includes active forces, as opposed to static conditions with no motion.
Dynamical system	A dynamic system is a dynamic system that exhibits cycle stability and the system can become chaotic with period doubling [2].
Emulation	An emulation is the imitation of the operation of a real-world process or system over time at both the software level and the hardware level.
Equivalent level of safety	A level of safety is equal to the standards for which equivalency is being sought.
Ground control station	The station at which the remote pilot manages the flight of an unmanned aircraft. A station usually consists of a computer and a hardware system for wireless communication.
Hard real time	When a system is hard real-time, any missed deadlines are considered to be a system failure [3].
Jacobian matrix	The Jacobian matrix of a vector-valued function of several variables is the matrix of all its first-order partial derivatives.
Low latency	Latency is a time delay between the cause and the effect of some physical change in the system being observed. In this thesis, low latency in environment perception, collision detection, and collision avoidance prevents the vehicle from traveling a significant distance.
Model predictive control	A control scheme that involves repeatedly solving a constrained optimization problem, using predictions of future costs, disturbance, and constraints over a moving time horizon to choose the control action [4].
Motion planning	Motion planning, also known as trajectory planning, is a computational problem to finding a sequence of a state and control constrained reference signal that moves the object from the source to the destination.

Non-cooperative sensor	A non-cooperative sensor that is on board an aircraft detect targets autonomously.
Non-linear model	The state-space representation describes the physical system with nonlinear relationships. In this thesis, the non-linearity of the quadrotor model is brought by the heading and velocity vectors relative to the body frame.
Optimal control	The process of determining control and state trajectories for a dynamic system over a period of time to minimize a performance index.
Prediction horizon	The prediction horizon extends past the control horizon to predict the final controlled variable outcomes but without manipulated variable movement.
Quaternion	Quaternion is a four-dimensional vector space that is a mathematical notation for representing spatial orientations and rotations of elements in three-dimensional space. The quaternion for coordinate system transformation has an advantage in size compared to the rotation matrix based on the Euler's rotation theorem.
Real time system	A real-time system can respond to external and internal events faster than the time constraints imposed on the system [3]. In this thesis, the reference time constraints are manned aircraft pilot's "see-and-avoid" criteria as well as the definition of collision avoidance by the Federal Aviation Administration (FAA).
Remotely piloted aircraft	An aircraft where the flying pilot is not on board the aircraft.
Resolution advisory	A resolution advisory given to the flight crew is either a type of climb or descent maneuver to provide separation from all threats, or a maneuver restriction intended to maintain existing separation.
Simulation	A simulation is the imitation of the operation of a real-world process or system over time at the software level.
Sliding view	The region of perception of a non-cooperative airborne sensor that is moving with the aircraft.
Soft real time	When a system is soft real-time, some delays in deadlines are acceptable [3].
Software-in-the-loop	In software-in-the-loop testing, the compiled source code is incorporated into the mathematical simulation that contains the models of the physical system. In this thesis, the proprietary

	software-in-the-loop simulation is a simulation of the autopilot where the autopilot code is linked to a flight simulator and the entire system runs on a single computer.
Unmanned aircraft system	An aircraft and its associated elements which are operated with no pilot on board.
Visual line-of-sight	A visual line-of-sight operation in which the remote crew maintains direct visual contact with the aircraft to manage its flight and meet separation and collision avoidance responsibilities.

LIST OF ABBREVIATIONS

ACAS	*Airborne Collision Avoidance System X*
AGL	*above ground level*
APF	*artificial potential field*
API	*application programming interface*
ATM	*air traffic management*
AWS	*Amazon Web Services*
BVLOS	*beyond visual line-of-sight*
CIC	*computer in control*
DAA	*detect and avoid*
DARPA	*Defense Advanced Research Projects Agency*
EC2	*Amazon Elastic Compute Cloud*
EKF	*extended Kalman filter*
ELOS	*equivalent level of safety*
ENU	*east, north, up*
FAA	*Federal Aviation Administration*
FCC	*Federal Communications Commission*
FOV	*field of view*
FPGA	*field programmable gate arrays*
FSM	*finite state machine*
GCS	*ground control station*
GUI	*graphical user interface*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
LAN	*local area network*
LIDAR	*light detection and ranging*
MAC	*mid-air collision*
MCP	*monotonic concession protocol*

MESA	*multi-role electronically scanned array*
MIMO	*multiple-input multiple-output*
MOPS	*minimum operational performance standards*
MPC	*model predictive control*
NAS	*National Airspace System*
NASA	*National Aeronautics and Space Administration*
NMPC	*non-linear model predictive control*
ODE	*ordinary differential equation*
QP	*quadratic programming*
RA	*resolution advisory*
RPAS	*remotely piloted aircraft system*
RPV	*remotely piloted vehicle*
RRT	*rapid exploring random tree*
RRT*	*rectified rapid exploring random tree*
RTCA	*Radio Technical Commission for Aeronautics*
SAA	*sense and avoid*
SDK	*software development kit*
SQP	*sequential quadratic programming*
SWIL	*software-in-the-loop*
TCAS	*Traffic Collision Avoidance System*
TCP	*Transmission Control Protocol*
TOCA	*time of closest approach*
UAS	*unmanned aircraft system*
sUAS	*small, unmanned aircraft system*
UAV	*unmanned aerial vehicle*
UTM	*unmanned aircraft system traffic management*
VFR	*visual flight rules*

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS

Notation: Scalars are denoted by plain text, italics. Vectors and matrices are denoted by bold text

θ	Pitch angles
σ	UAV outputs
φ	Roll angles
ψ	Yaw angles
e	Slack variable
J	Cost function for MPC
J_{feul}	Cost function term for optimal fuel consumption
J_{ref}	Cost function term for optimal reference trajectory tracking
J_{slack}	Cost function term for minimum slack variable
$\overrightarrow{\boldsymbol{OA}}$	Desired tangent of the cone
$\overrightarrow{\boldsymbol{OC}}$	Original trajectory of the UAV relative to the obstacle
$\overrightarrow{\boldsymbol{OP}}$	Position the obstacle relative to the UAV
$\overrightarrow{\boldsymbol{OX}}$	Position of the center of the base of the collision cone relative to the UAV
$\overrightarrow{\boldsymbol{OY}}$	Position of the intersection between original trajectory and the cone base
$\boldsymbol{p}$	Coordinates of the detected object returned from the airborne radar
$\boldsymbol{p}'$	ENU coordinates of the detected object returned from the airborne radar

$\boldsymbol{q_O}$	Quaternion representation of the transformation from the ENU coordinate system to the local Cartesian system
$\boldsymbol{q_R}$	Quaternion representation of the UAV rotation with reference to the local Cartesian coordinate system
$\boldsymbol{q_x, q_y, q_z}$	Quaternion representations of the UAV axis rotation with reference to the local Cartesian coordinate system
r_o	Radius of the sphere of the obstacle
$\boldsymbol{r_{obs,i}}$	Center of mass position of the obstacle i
$\boldsymbol{r_{obs}^B}$	Center of mass position of the obstacle in body frame
s	Sensitivity of slack variable
$\boldsymbol{T}$	Transformation matrix from the ENU coordinate system to the local Cartesian system
$\boldsymbol{u}$	UAV control inputs
$\boldsymbol{V_f}$	Original velocity vector of the UAV with relative to the obstacle
$\boldsymbol{V_{desired}^r}$	Desired relative velocity of UAV
$\boldsymbol{V_{desired}^w}$	Desired absolute velocity of UAV
$\boldsymbol{V_{Obs}}$	Velocity of the obstacle in world frame
$\boldsymbol{v_{obs}^B}$	Obstacle velocity in body frame
w_1, w_2, w_3, w_4	Weights for cost function terms
$\boldsymbol{X}$	UAV state vector

I. INTRODUCTION

1.1 Problem Statement

This thesis aims to address two problems in today's detect-and-avoid (DAA) system development:

1. A non-cooperative sensor based collision detection and avoidance system for beyond visual line-of-sight (BVLOS) unmanned aircraft system (UAS) operation; and
2. Real-time collision avoidance for DAA system.

1.1.1 Motivation

Over the past decades, UAS technologies and regulations have evolved quickly. BVLOS operation is critical to the commercial viability of the UAS as it provides services such as cargo delivery [5], urban mobility, and inspection [6].

For conventional piloted aircraft, the eyesight of a human pilot is still a primary method to avoid mid-air collision (MAC), even with the presence of transponders and radar systems [7]. For UAS, which is remotely piloted, it does not have this safety feature, resulting in an increased risk of MAC. Therefore, safe use of BVLOS UAS requires DAA technologies; otherwise, it requires visual observer(s) along the entire flight path to assist the pilot by providing detect-and-avoid functions [8].

In recent years, there have been many emerging DAA approaches for UAS in literature or development. However, most known emerging approaches, such as Airborne Collision Avoidance System X (ACAS X), also rely exclusively on cooperative sensors [9].

1.1.2 Problem Definition

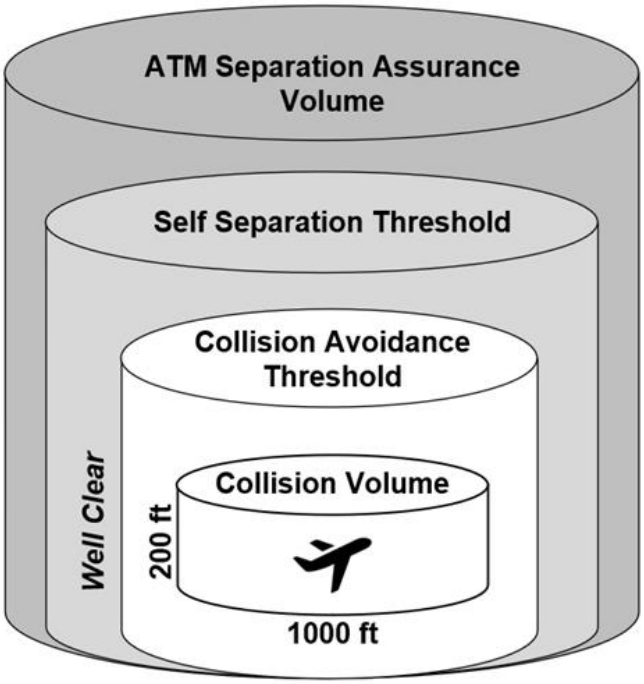


Fig. 1.1. SAA airspace volumes.

In general, a DAA (also referred to as Sense and Avoid or SAA) system can be developed based on either cooperative and/or non-cooperative sensors. It is intended to provide the UAS with the ability to self-separate from (i.e., remain well clear of) and avoid collisions with other aircraft or flying objects [10].

The definition of real-time collision avoidance in this thesis is adopted from the concepts brought by the Federal Aviation Administration (FAA) – a transportation agency of the U. S. government. In 2009, the FAA sponsored workshops to develop concepts for routine integration of UAS into the National Airspace System (NAS) [7] so that UAS can utilize existing controlled airspace facilities and services for DAA. The workshop concluded that DAA consists of two functional components [1]:

1. Self-Separation: “Function that reduces the probability of a collision by ensuring aircraft remaining well clear of each other, thereby assuring safe separation.” [1, p. 1]
2. Collision Avoidance: “Extreme maneuvers just prior to the closest point of approach to prevent collision in cases where safe separation is lost.” [1, p. 1]

As shown in Fig. 1.1, the two concepts are illustrated by a set of airspace volumes surrounding the unmanned aircraft: (i) a collision volume, (ii) a collision avoidance threshold (also referred to as well-clear volume), (iii) a self-separation threshold, and (iv) an air traffic management (ATM) separation assurance volume [11]. Therefore, the basic requirement of a non-cooperative DAA function is that the intruder has to be detected at a range larger than the collision volume and the ownership aircraft has sufficient time to avoid the collision.

DAA system equipped with non-cooperative sensors for UAS is an open-ended problem, and it is insufficiently studied. Considering the maximum altitude of 400 ft or 121 m Above Ground Level (AGL) for UAS operation in Canada [8] and corresponding air traffic complexity, cooperative sensors such as transponders or UAS Traffic Management services are not capable of detecting obstacles such as unregistered small Unmanned Aerial Systems (sUAS) and birds. Therefore, non-cooperative DAA is required for UAS BVLOS flight safety. It exploits autonomous sensors installed onboard, such as radars and cameras, to estimate the position and velocity of an intruder [12]. Conventional collision avoidance is achieved by both airborne collision avoidance systems such as Traffic Collision Avoidance System (TCAS), which rely exclusively on cooperative sensors [12] and the pilot's "see and avoid" operation [13]. Future DAA system approaches, such as Airborne Collision Avoidance System X (ACAS X), also rely exclusively on cooperative sensors as well [9].

1.1.3 Proposed Solution

The problem of non-cooperative DAA system development is decomposed and proposed to be solved in the following perspectives, as shown in Table 1.1.

TABLE 1.1. Decomposed Problems and Corresponding Solutions

Subproblem	**Proposed Solution**
Non-cooperative sensor selection	Utilization of the equivalent level of safety (ELOS) of "see and avoid"
Real-time interpretation of sensor detections	Quaternion-based radar data interpretation and generation
Software-in-the-loop (SWIL) simulation	
Integration of other UAS traffic management (UTM) services	Multi-threaded object orientation and command query segregation
DAA system operation and monitoring	Finite state machine and heartbeat event
Future on-board/on-cloud computation requirements	Distributed system of systems over Local Area Network (LAN)

The proposed solution of real-time trajectory generation for the DAA system in this thesis consists of three development stages.

In the first stage, a reactive collision cone approach is adopted and integrated into the presented DAA system, which considers the characteristics of the airborne radar sensor. This approach also helps to validate and verify the presented DAA system.

In the second stage, a motion planning approach, non-linear model predictive control (NMPC), is proposed to generate dynamically feasible trajectories. The concept of model predictive control (MPC) is consistent with the sliding view of the airborne radar. The differential flatness of the UAV is used to build the non-linear model.

In the third stage, a hierarchical architecture is proposed to combine the approaches in the past two stages to generate a dynamically feasible trajectory with greater prediction and control horizon in real-time computation.

1.2 Thesis Formulation

1.2.1 Thesis Statement

This thesis aims to (i) design and implement a non-cooperative sensor based DAA system for BVLOS UAS operation, and (ii) present a hierarchical MPC architecture for real-time collision avoidance trajectory planning for the DAA system.

1.2.2 Thesis Objectives

The thesis has three primary objectives:

1. Design a DAA system that meets the system requirements in Table 1.1;
2. Identify implementation measures for the performance of the real-time collision avoidance trajectory planning, including:
   a. Identify and design common BVLOS UAS flight scenarios;
   b. Implement the collision cone based approach and non-linear MPC approach; and
   c. Experiment two approaches with designed flight scenarios to establish the performance criteria.
3. Design a hierarchical MPC architecture, including:
   a. Design a hierarchical framework to incorporate concepts like path planning, hierarchical MPC, and collision cone approach; and
   b. Tune and test the solution performance in the previously designed flight scenarios.

1.2.3 Research Questions

The thesis presents several research questions regarding non-cooperative DAA and real-time collision avoidance approaches. The following list of questions highlights some of the major topics and describes which ones are addressed in this thesis.

1. What are essential measures of a non-cooperative DAA system?
   a. What are essential measures of collision detection?
   b. What are essential measures of collision avoidance?

Comments: The detection and avoidance capabilities of a non-cooperative DAA system are supposed to comply with the see-and-avoid requirement and navigational awareness. Thus, one of the minimum levels of requirements for real-time detection and measurement capability is to comply with the equivalent level of safety (ELOS) of manned aircraft pilots "see and avoid".

2. Why is collision cone based approach proposed for reactive collision avoidance?
   a. What are the essential properties of this approach?
   b. What are the primary advantages of this approach?

   Comments: Among various reactive collision avoidance approaches, the collision cone approach has fully utilized the characteristics of the selected airborne radar and BVLOS operation in the formation of the collision geometry and the computation of the collision-free trajectory, resulting in real-time computation.
3. Why is NMPC based approach proposed?

   Comments: Compared with sampling-based planners and interpolating curve planners, the obstacles and flight operation constraints can be taken into account easily, and a prediction of future system behavior can be provided. The adoption of differential flatness of UAV also makes this approach more computationally efficient.
4. Why is a hierarchical solution designed?
   a. What characterize the performance of the hierarchical solution?

   Comments: Both the MPC approach and the collision cone approach have major limitations resulting in sub-optimal solutions. For the collision cone approach, it is not aware of the UAV dynamics, and the generated trajectory is actually a path with desired constant motion state. For the NMPC approach, its high computational cost limits the size of the prediction and control horizon. The performance of the hierarchical solution is evaluated based on the limitations and the features of the two previously presented approaches in the designed experiments.

   b. What are the primary advantages of the hierarchical solution?

   Comments: The combined solution decouples collision avoidance and motion tracking by incorporating the concept of global and local planners. The high-level NMPC generates minimum snap trajectories from the path generated by the collision cone approach. The low-

level NMPC is in charge of path tracking and collision avoidance in real-time. It is able to overcome the limitations and keep the features of the two previously presented approaches.

1.3 Thesis Organization

This thesis presents a non-cooperative DAA system and a hierarchical real-time collision avoidance architecture for DAA system.

Chapter 2 provides an overview of non-cooperative sensors and real-time collision avoidance approaches.

Chapter 3 presents the DAA system architecture design and implementation except for its real-time collision avoidance module, which is presented in Ch. 4.

Chapter 4 presents the stages of development of the real-time hierarchical collision avoidance architecture for the presented DAA system.

1.4 Associated Publications and Reports

Chapter 3 and Section 4.2 together present the design and implementation of an airborne radar based DAA system, and the associated published paper is [14].

Section 4.3 presents the design and implementation of a hierarchical solution for real-time collision avoidance in a varying environment, and the associated paper (submitted for review) is [15].

The associated technical report (in preparation) for the software developed in this thesis is [16].

II. LITERATURE REVIEW

2.1 Non-cooperative Sensors

Common sensing approaches for non-cooperative collision avoidance are vision, light detection and ranging (lidar, also LiDAR, or LIDAR), and radio detection and ranging (radar). The selection of sensors should consider the following criteria: (i) real-time detection and measurement capability, (ii) operational environment, and (iii) aviation regulations.

According to U.S. UAS operational approval for UAS flights within the National Airspace System (NAS), compliance with the see-and-avoid requirement and navigational awareness are primary concerns in UAS operational approvals [17]. Thus, one of the minimum levels of requirements for real-time detection and measurement capability is to comply with the equivalent level of safety (ELOS) of manned aircraft pilots "see and avoid". Table 2.1 shows the summarized key regulations and performance for collision avoidance of high-speed manned aircraft [18], [19].

TABLE 2.1. "See-and-avoid" ELOS.

Performance Parameter	ELOS
Missed Distance	500 feet (152 m)
Field of Regard Az, El	± 60 °, ± 10 °
Visual Flight Rules (VFR) Detection Range	1.84 miles (2961 m)
Aircraft Recognition and Collision Course Awareness	6.1 s
Collision Avoidance Maneuver	6.4 s

Camera-based solutions have been mainly studied in recent years because of the greatly increased performance of computers and the advancement of machine learning techniques. Many

recent UAS collision avoidance approaches in literature adopt cameras together with machine learning techniques [20]–[23], but most of the flight scenarios are in-door and low-speed flights with obstacles within dozens of meters. Vision-based sensors are not sufficient for the DAA function described in Sec. 1.1.2 because of the following reasons:

1. It is hard to detect and identify flying objects at a large distance from the own aircraft. Accurate object detection and classification algorithms for flying objects are required to identify the model of a flying aircraft and determine its size. The resolution requirements and the availability of large datasets for training are big problems.
2. Since the distance error increases non-linearly for given stereo sub-pixel disparity errors [24], in a flight scenario where obstacles are hundreds of meters away from the own aircraft, the distance and velocity estimations by vision are unusable.

TABLE 2.2. Main Specifications of Three Considered Radar Sensors [12].

	Echodyne EchoFlight™ DAA MESA Radar	IMST sR-1200e™	Aerotenna μSharp Patch™
Field of View	≥120° Az x 80° El	65° Az x 24° El	50° Az x 30° El
Scan/Update Rate	≈ 1 Hz for 120° Az x 40° El	10 Hz - 200 Hz	90 Hz
Detection Range	3400 m (max range) (>750 m for small UAS)	307 m (max range)	120 m (max range)
Relative Position Measurements	Range, azimuth, elevation	Range, azimuth (based on phase difference)	Range
Sensing Resolution	±1° Azimuth, ±3° Elevation	≤ 0.6 m (range), 6.25 m/s (velocity), 2°-3° Angle	22 cm (range)
Operating frequency	24.45 GHz – 24.65 GHz (multi-channel)	24.00 GHz - 24.25 GHz	24.00 GHz
Weight	730 g – 817 g	280 g	43 g
Size (cm)	18.7 x 12.1 x 4.0	9.8 x 8.7 x 0.42	7.6 x 5.4 x 1.3
Power (operating)	< 35 W	4.5 W	1.25 W

For lidar, it has been adopted widely in autonomous driving and mapping applications, but it is not allowed to be equipped on UAS. According to Canadian Aviation Regulations, the projection of directed bright light sources at an aircraft is prohibited because it may create a hazard to aviation or cause damage to an aircraft or injury to persons on board the aircraft [8].

For airborne radar, it has been adopted in TCAS to estimate the range, angle, and closing velocity of traffic relative to the own aircraft. Current challenges for utilizing airborne radar in UAS are given to the limited onboard power availability and the low radar cross section for targets of interest, and only a few radar systems are commercially available for UAS, as shown in Table 2.2 [12].

2.2 Real-time Collision Avoidance State-of-the-Art

Firstly, the real-time collision avoidance approach from manned aircraft is insufficient for UAS BVLOS operation. TCAS issues resolution advisory (RA) in real-time when the intruder is within the flight envelope equivalent to the well-clear volume in Fig. 1.1. RAs are a set of climbing or descending instructions, and Fig. 2.1 shows an example of increase descent RA for an encounter where the own aircraft needs to increase the descent rate. When a RA is issued, an immediate response from pilots is expected to mitigate the risk of collision. Considering the complex environment below 121 m AGL, a more complex solution space should be considered.

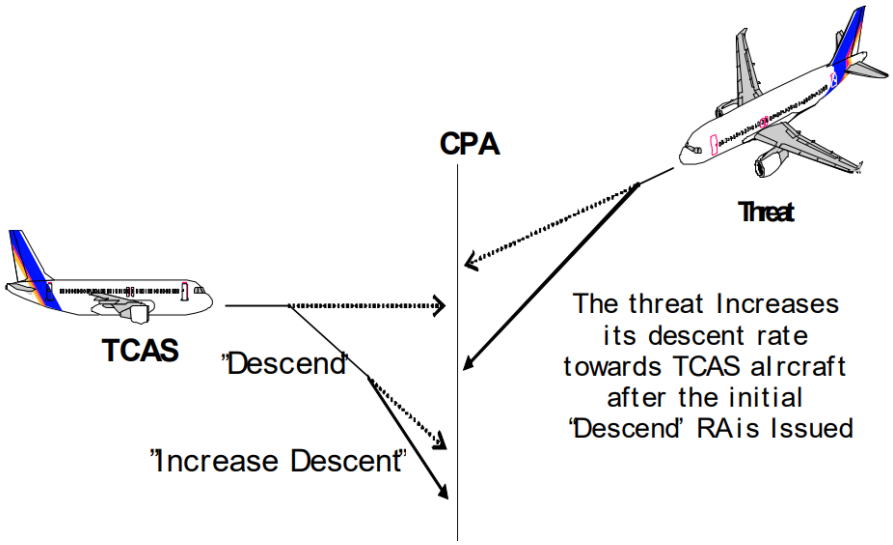


Fig. 2.1. Increase rate RA [25].

In general, there are two types of collision avoidance approaches, the reactive approach, and the planning approach. The reactive approach uses surrounding information from sensors and the UAS and reacts based on the information. It can generate control commands rapidly but may fall to a local minimum. For the planning approach, an optimal path or trajectory is computed when the environment information is updated. Thus, this approach requires one of the strategies: searching the environment, optimizing the reference path, or interpolating the path between waypoints. This approach is more computation-heavy as compared to reactive approaches.

One major category of the reactive approach is the geometric methods which rely on the analysis of geometric attributes [26]. Many works of literature adopt geometric methods for on-board collision avoidance [27]–[31]. In most of the literature, a novel collision cone based geometric approach [32] is applied to utilize the geometric attributes of the own aircraft and the obstacles. This collision cone approach was first proposed in 1998. In [27], the collision cone approach is represented in the body frame and used directly for fixed wing UAV control. Most recently, this idea has been used in scenarios where the uncertainties of the obstacles are considered by generating and evaluating each collision cone based trajectory generated based on a Gaussian Mixture Model [31]. Another major category of the reactive approach is the artificial potential field (APF) method [33] which uses the concept of potential field that the own vehicles are attracted to the desired positions and repulsed by the detected obstacles. Khatib first proposed the idea in 1985, and it has been widely used until today. Other than the common local minima problems for the reactive approach, the force-field method has other limitations such as non-reachable goals near obstacles. Therefore, new techniques are also incorporated. In [34], the APF method is used to guide the expansion of rapid exploring random tree (RRT) for path planning (planning approaches are detailly reviewed in the later paragraphs). In a recent study [35], the idea of the curl-free vector is incorporated. The direction of the curl-free vector is determined by the velocity of UAVs and obstacles, allowing the own UAVs to avoid collision in a dynamic environment.

Planning approaches have been a popular subject of study for autonomous vehicles for the last decades. In the study of autonomous vehicle, path planning and motion planning have fundamental differences in performance and stability. In general, path planning produces a sequence of positions from the start to the goal, whereas motion planning produces a discrete-

time sequence of state and control signals. In the scenario of UAS DAA described in this thesis, computation of a dynamically feasible time-parameterized trajectory is desired.

One category of motion planning is the graph searching planners such as A-Star (A*) algorithm. The basic idea of this category of motion planning is to traverse a state space to get from starting point to the goal and returns an optimal or suboptimal sequence of states (also referred to as nodes). This state space is either represented as an occupancy grid or lattice that describes the environment. A-Star algorithm is a fast path searching algorithm as it is a heuristic extension of Dijkstra's algorithm. Its major drawback is its space complexity due to storing all generated nodes in memory. Therefore, it is usually used for motion planning in a small-sized known environment [36]. Several applications have used A* algorithm as a basis for improvement. In [37], a hybrid A* algorithm that associates a continuous state with each cell is used in parking lot navigation for Junior (Stanford University automated vehicle) in the DARPA Urban Challenge.

Sampling-based planners, another category of motion planning approaches, are commonly used in order to solve the time constraints of those deterministic searching algorithms. Rapidly-Exploring Random Tree (RRT) is a commonly used sampling-based path planner in the autonomous vehicle by conducting random searches and continuously optimizing the tree-structured trajectories [38], [39]. Since the resulting path is not optimal and not curvature continuous, a rectified RRT (RRT*) was introduced to allow the paths to converge to optimal trajectories by adding a steering procedure [40]. However, RRT* is computationally heavier than RRT, and applications are limited to relatively low-speed flight within a relatively small space [41].

Due to the limited number of nodes generated by sampling-based planners, interpolating curve planners are commonly used to smoothen the generated path. Different path smoothing and curve generation techniques such as clothoid curves, polynomial curves, Bézier curves, and spline curves are used [42]. In [43], polynomial curve based trajectories are generated for lane-changing scenarios where fourth and fifth-degree polynomials for longitudinal constraints and fifth-degree polynomials for the lateral constraints are applied. In [44]–[46], the piecewise polynomial curves (splines) are generated and jointly optimized in quadratic programming (QP)

to generate a polynomial trajectory with minimum snaps for indoor, high-speed quadrotor flight. Although spline approaches make real-time trajectory planning in a large map feasible, it tends to violate the constraints as the optimization process is to ensure the continuity between each polynomial curve segment. Therefore, extra computation power is spent on collision check, re-split segments, and re-optimization in [45].

Lastly, model predictive control (MPC) is a popular approach to trajectory generation and tracking problems. Compared with sampling-based planners and interpolating curve planners, the obstacles and flight operation constraints can be taken into account easily, and a prediction of future system behavior can be provided. However, the MPC approach is usually computationally expensive, especially as the function optimization takes place at each motion state. Therefore, many applications are reduced to low-speed flights, especially when obstacles constraints are taken into account [47], [48].

2.3 Decentralized Collision Avoidance

For decentralized DAA, in which the own aircraft generates independent resolution advisories or trajectories, this may lead to inefficient or unsuccessful separation in a multi-agent system. In other words, the aircraft is among other aircraft equipped with the same DAA system. This problem is not sufficiently studied in planner-based DAA literature. One common approach is the formation control in which multi-agents are kept in the formation, such as movement with a leader [49] and coordinated movement between agents [50]. Classic monotonic concession protocol (MCP) based on game theory is another approach that was studied for the self-separation of manned aircraft [51]. The MCP captures the incremental negotiation process between two aircraft, and the agents incrementally make proposals and counter-proposals of progressively less value to themselves until an equilibrium of risk is met. In [52], the limit cycle technique is used to specify the direction of the trajectory to be clockwise or anticlockwise, bypassing the circular obstacle.

2.4 Summary of Chapter 2

This chapter highlighted some fundamental principles behind non-cooperative DAA systems and real-time collision avoidance. Simultaneously, the requirement analysis and the review of non-cooperative sensors for the DAA system in this chapter narrow the range of sensor selection. Chapter 3 focuses on the DAA system developed from the selected non-cooperative sensor.

A review of the existing real-time collision avoidance algorithms is also provided in this chapter. These concepts are used for real-time collision avoidance trajectory generation in Ch. 4.

III. AIRBORNE RADAR BASED OBSTACLE DETECTION SYSTEM

3.1 Overview

This chapter presents the design and implementation of the DAA system used in this thesis. The hardware used for the DAA system is presented first. The airborne radar data processing and interpretation are then presented. Lastly, the DAA system and component designs are presented.

3.2 Hardware

3.2.1 UAS

The sUAS used is AAT-1200 provided by Aurora Aerial Inc [53]. It is a UAV with a maximum payload of 14 kg. The UAS used a Micropilot autopilot system. One noticeable feature of this UAV is a failsafe auto-deployment parachute mounted on top of the UAV as shown in Fig. 3.1.

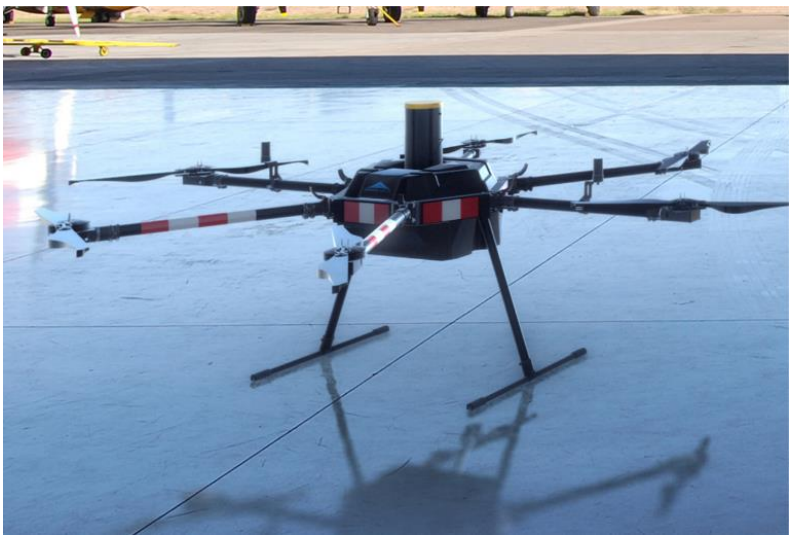


Fig. 3.1. AAT-1200 drone [53].

One common operation scenario for such type of UAS is a 30 min flight following a preplanned path at a cruise speed of 10 - 20 m/s for payload delivery.

3.2.2 Airborne Radar

Fig. 3.2. EchoFlight™ MESA radar [54].

After consideration of common BVLOS scenarios as well as “see-and-avoid” ELOS described in Sec. 2.1, Echodyne EchoFlight™ DAA MESA radar [54] is purchased and used in this thesis for obstacle detection. In terms of legal and safety concerns for radar usage in civil applications, it has received a Grant of Equipment Authorization from the U.S. Federal Communications Commission (FCC). It is safe to operate by following the FCC Requirements for Radio Frequency Exposure.

The selected airborne radar has a default operating frequency centered at 24.55 GHz, and its embedded field-programmable gate array (FPGA) processes each Range-Velocity (RV) map collected and is capable of providing information on detected tracks with a maximum throughput of 25 kB/s [54]. Each track data packet contains the location and velocity of the detected and tracked objects in a three-dimensional Cartesian coordinate system with reference to the face of the radar antenna, and the minimum tracking range varies from 750 m to 2 km depending on the size of the intruder [54].

Considering the field of view of the EchoFlight™ radar as described in Table 2.2, the mounting location of this 700g airborne radar is shown in Fig. 3.3. The mounting is designed and implemented by AIRvanced Technology Inc.

The control and data transmission are through Gigabit Ethernet.

Fig. 3.3. Airborne radar mounted on AAT-1200 during hovering test.

Fig. 3.4. A picture of radar detection flight test setup. The AAT-1200 equipped with radar is placed on top of roof. The airborne radar is expected to detect another UAV which will be taking off on the runway.

3.2.3 Ground Control Station

The ground control station (GCS) is a part of the provided AAT-1200 UAS. It is a laptop running Micropilot HORIZON^{mp}, the ground control software.

Multiple DAA system programs developed and presented in this thesis also run on this GCS interfacing with the airborne radar and the HORIZON^{mp}.

3.2.4 Airborne Radar Datalink

The datalink module selection and implementation are collective work done by AIRvanced Technology Inc. and me.

In order to achieve real-time data transmission between airborne radar and the DAA system on GCS, a low latency multiple-input and multiple-output (MIMO) data link is selected and configured to build a wireless ethernet communication between the airborne radar and the GCS. It is also used to support the wireless communication between the autopilot and the HORIZON^{mp}.

3.3 Airborne Data Interpretation

3.3.1 Overview

The radar detection outputs and UAV attitude are in coordinate systems different from the geographic coordinate system. In order to visualize the object detected by the airborne radar in a map for collision advisory and avoidance, a transformation of the coordinates from the local Cartesian coordinate system to the East, North, Up (ENU) coordinate system is needed.

The quaternion-based transformation of the coordinate system enables the integration of the non-cooperative sensor into UAS as well as airborne radar simulation in a simulated environment.

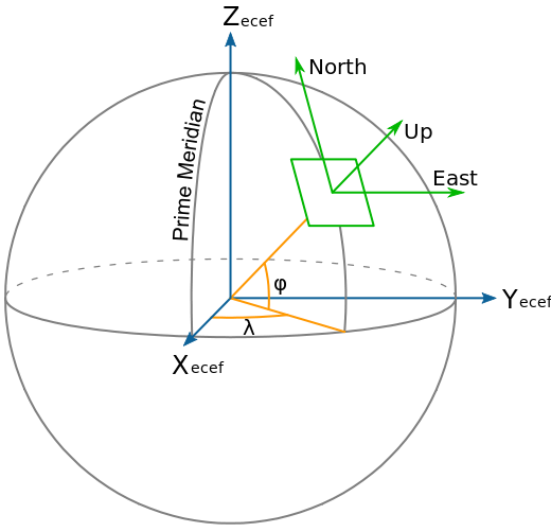


Fig. 3.5. ENU Longitude Latitude relationships [55].

3.3.2 Coordinate System

As shown in Fig. 3.6, the position and velocity of the detected objects are measured in a local Cartesian coordinate system with reference to the radar antenna. The detected obstacle position and velocity representation in the body frame is

$$r_{obs}^{B} = [x, y, z]^{T} \tag{3.1}$$

$$v_{obs}^{B} = [v_x, v_y, v_z]^{T} \tag{3.2}$$

The representation is divided into sidewards, upward, and forward positions or velocities with reference to the radar antenna, respectively.

Assuming the radar mounted on the UAV is level and facing forward with respect to the UAV, it has the same orientation as the UAV. According to the telemetry data from the autopilot, in the local Cartesian coordinate system, the yaw is defined as left-handed rotations of the y-axis, the pitch is defined as the left-handed rotation of the x-axis, and roll is defined as the right-handed rotation of the z-axis.

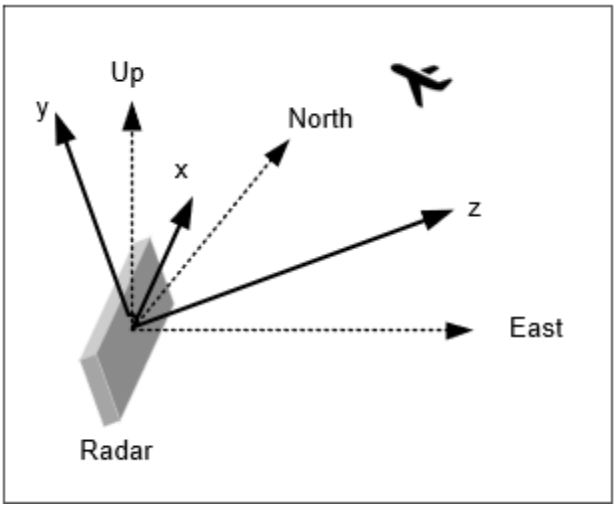


Fig. 3.6. Illustration of radar's Cartesian coordinate system with reference to ENU coordinate system.

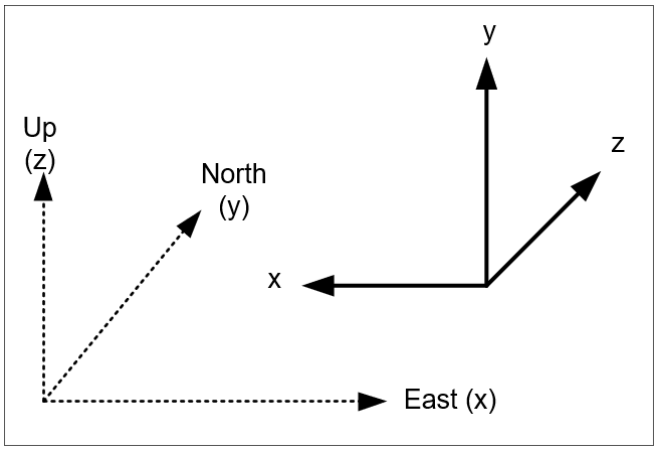


Fig. 3.7. Definition of yaw, pitch, and roll relative to ENU coordinate system. When yaw, pitch, and roll are set to zero degrees, the UAV is level and pointing to true north.

3.3.3 Quaternion

Before the coordinate system transformation for airborne radar data interpretation is presented, quaternions are first introduced. Only properties that are related to coordinate system transformation in Sec. 3.3.4 are introduced in this section.

Quaternions allow the description of the aircraft orientation in all possible attitudes with no inherent geometrical singularity compared to Euler angles. In addition, quaternion-based computation is suitable for real-time applications because no trigonometric relations exist in the

quaternion-based computation of rotations and transformation. Trigonometric computation is only required for the formulation of the quaternions in the presented system.

Generally, a quaternion q consists of a scalar part and a vector part

$$q = [s, v], s \in R, v \in R^3 \tag{3.3}$$

A unit quaternion represents a unique rotation in space, whereas the magnitude represents scaling in space which is not used in this application. The unit quaternion below represents a right-handed rotation of θ about an arbitrary unit vector $\hat{\mathrm{k}}$

$$q = [\cos(\theta/2), \sin(\theta/2)\hat{\mathrm{k}}] \tag{3.4}$$

Since quaternion is four-dimensional, the Hamilton product is defined for quaternion multiplication. The multiplication of basis elements is defined below

$$ij = k, ji = -k, \tag{3.5}$$
$$jk = i, kj = -i, \tag{3.6}$$
$$ki = j, ik = -j, \tag{3.7}$$
$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.8}$$

The Hamilton product of two quaternions is determined by the multiplication of basis elements with distributive law. It can be easily verified that Hamilton product is:

1. Associative; and
2. Not commutative except for multiplication by a scalar quaternion in form of $[\mathrm{s}, 0]$.

By utilizing Eq. 3.4, rotation of a vector p by θ about the vector $\hat{\mathrm{k}}$ can be expressed as

$$p' = qpq^{-1} \tag{3.9}$$

where Hamilton product is used, $q^{-1} = [s, -v]$ is the conjugate of q.

The composite rotation of p by q_0 and q_1 can be expressed as

$$p' = (q_0 q_1)p(q_0 q_1)^{-1} \tag{3.10}$$

The relationship between quaternion and rotation matrix is obtained by expanding p' in Eq. 3.9 into the components of $v = [x, y, z]$

$$p' = (s^2 - v \cdot v)p + 2s(v \times p) + 2(v \cdot p)v \tag{3.11}$$

$$p' = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - sz) & 2(xz + sy) \\ 2(xy + sz) & 1 - 2(x^2 + z^2) & 2(yz - sx) \\ 2(xz - sy) & 2(yz + sx) & 1 - 2(x^2 + y^2) \end{bmatrix} p \tag{3.12}$$

3.3.4 Transformation of Airborne Radar Data

As shown in Fig. 3.6, other than rotations of the local Cartesian coordinate system, there is also a difference in orientations between the local Cartesian coordinate system and the ENU coordinate system. The transformation matrix and quaternion representation from the ENU coordinate system to the local Cartesian system are shown, respectively

$$T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.13}$$

$$q_O = \begin{bmatrix} 0 \\ 0 \\ 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \tag{3.14}$$

Note that T is an involutory matrix, and we have $T = T^{-1}$ and $q_O = {q_O}^{-1}$ which simplifies the inverse computation for radar simulator in Sec. 3.4.3.

According to the NASA document describing the quaternion-based calculations, there are twelve permutations of possible 3-axis rotation sequences to compute the quaternion of the coordinate system transformation as a function of the Euler angles [56]. In a recent study, the selection of rotation sequence has effects on execution time and accuracy in aircraft attitude control simulation with variable step size [57]. The axis rotation sequence of the z-axis, x-axis and y-axis [56] is used, which has relatively low execution time and high accuracy in the simulation environment described in [57].

The quaternion of the UAV rotation, q_R, with reference to the local Cartesian coordinate system is then represented as

$$q_R = q_z q_x q_y \tag{3.15}$$

$$= \begin{bmatrix} \cos\left(\frac{\varphi}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\varphi}{2}\right) \end{bmatrix} \begin{bmatrix} \cos\left(-\frac{\theta}{2}\right) \\ \sin\left(-\frac{\theta}{2}\right) \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \cos\left(-\frac{\psi}{2}\right) \\ 0 \\ \sin\left(-\frac{\psi}{2}\right) \\ 0 \end{bmatrix} \tag{3.16}$$

$$= \begin{bmatrix} -sin\left(\frac{\psi}{2}\right) * sin\left(\frac{\theta}{2}\right) * sin\left(\frac{\varphi}{2}\right) - cos\left(\frac{\psi}{2}\right) * cos\left(\frac{\theta}{2}\right) * cos\left(\frac{\varphi}{2}\right) \\ -sin\left(\frac{\psi}{2}\right) * cos\left(\frac{\theta}{2}\right) * sin\left(\frac{\varphi}{2}\right) - cos\left(\frac{\psi}{2}\right) * sin\left(\frac{\theta}{2}\right) * cos\left(\frac{\varphi}{2}\right) \\ sin\left(\frac{\psi}{2}\right) * sin\left(\frac{\theta}{2}\right) * cos\left(\frac{\varphi}{2}\right) - cos\left(\frac{\psi}{2}\right) * cos\left(\frac{\theta}{2}\right) * sin\left(\frac{\varphi}{2}\right) \\ sin\left(\frac{\psi}{2}\right) * cos\left(\frac{\theta}{2}\right) * cos\left(\frac{\varphi}{2}\right) - cos\left(\frac{\psi}{2}\right) * sin\left(\frac{\theta}{2}\right) * sin\left(\frac{\varphi}{2}\right) \end{bmatrix} \tag{3.17}$$

Here, ψ, $\theta$and φ are yaw, pitch, and roll angles of the UAV respectively; q_x, q_y and q_z are the quaternion representation of axis rotation according to the uncommon yaw, pitch, and roll definition illustrated in Fig. 3.7.

Therefore, the quaternion representing the orientation and rotation of the detected object in the ENU coordinate system is expressed as

$$q = q_R q_O \tag{3.18}$$

The resulting ENU coordinates of the object relative to the current drone location, p', equals to

$$p' = qpq^{-1} \tag{3.19}$$

where

$$p = \begin{bmatrix} 0 \\ x \\ y \\ z \end{bmatrix} \tag{3.20}$$

Here, x, y and z are the coordinates of the detected object returned from the airborne radar.

3.4 Software

3.4.1 System Architecture

The presented software designs and algorithms in this thesis are work done by me. I am also the lead and major contributor to the implementation of the presented prototype version of the DAA system.

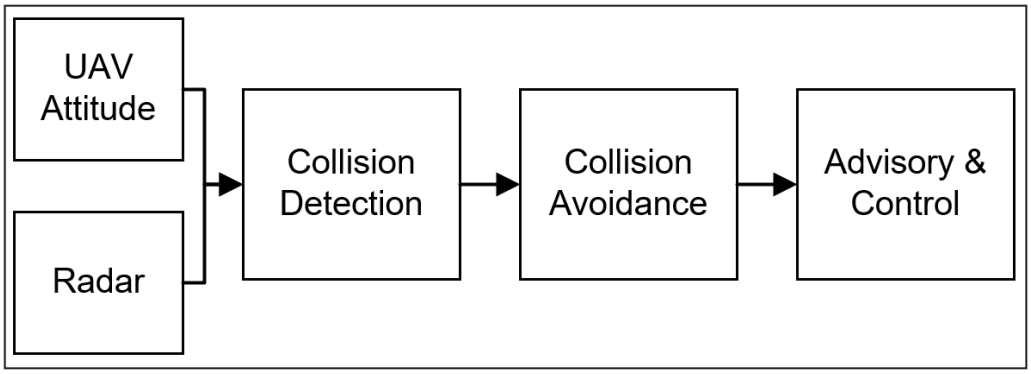


Fig. 3.8. The block diagram demonstrates the function modules for the DAA system.

As shown in Fig. 3.8, the Collision Detection module process the position and velocity information of the obstacles using UAV attitude information. It then identifies any obstacles that are of immediate threats to the UAS.

The Collision Avoidance module generates the desired trajectory using the collision cone based reactive collision avoidance approach to avoid the most immediate threat. This collision cone approach is presented in Sec. 4.2.

Lastly, the Control and Advisory module provides a graphical presentation of the desired trajectories and corresponding parameters to support both collision risk mitigation for pilots and Computer in Control (CIC) maneuvers in Remotely Piloted Vehicle (RPV) mode, a mode that controls the autopilot by inputting the desired altitude, velocity and heading.

Other than achieving the presented block diagram for the airborne radar based collision detection and avoidance as shown in Fig. 3.8, the following awarenesses are also considered: (i)

integration of other air traffic services, (ii) SWIL simulation environment for system testing, and (iii) distributed DAA system for future on-board and on-cloud computation requirements.

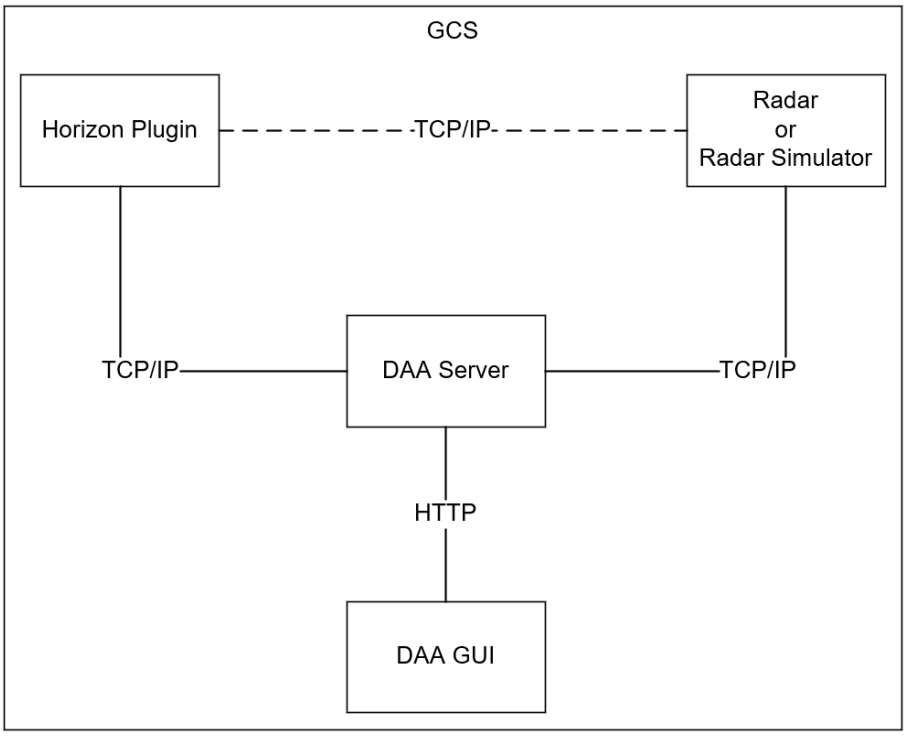


Fig. 3.9. Network diagram of DAA system components.

Therefore, the DAA system is designed and developed as a system of systems that consists of the following software and programs as shown in Fig. 3.9: a GCS plugin, an airborne radar or radar simulator, a DAA server, and a DAA GUI. Each component is described in the later sections.

The GCS plugin constantly accesses and transmits UAV attitudes. The radar simulator is developed for flight scenario construction in a SWIL simulation environment. The DAA server plays a crucial role in data perception, collision detection, and avoidance trajectory generation. The DAA GUI is designed and developed for DAA system operation and visualization of collision avoidance information.

In addition, in the scenario of SWIL simulation, a Transmission Control Protocol/Internet Protocol (TCP/IP) connection is established between the GCS plugin and the radar simulator. In the scenario of a flight test, there is no TCP/IP connection between the airborne radar equipped

and the horizon plugin. The DAA server is unable to distinguish the difference between the radar simulator and the actual radar in any of the scenarios.

3.4.2 Autopilot System Plugin

As shown in Fig. 3.10, the role of the autopilot system plugin is to perform the two major missions simultaneously:

1. Accesses UAV attitudes and transmits to requested components; and
2. Accepts commands from DAA server and controls UAV in Remotely Piloted Vehicle (RPV) mode

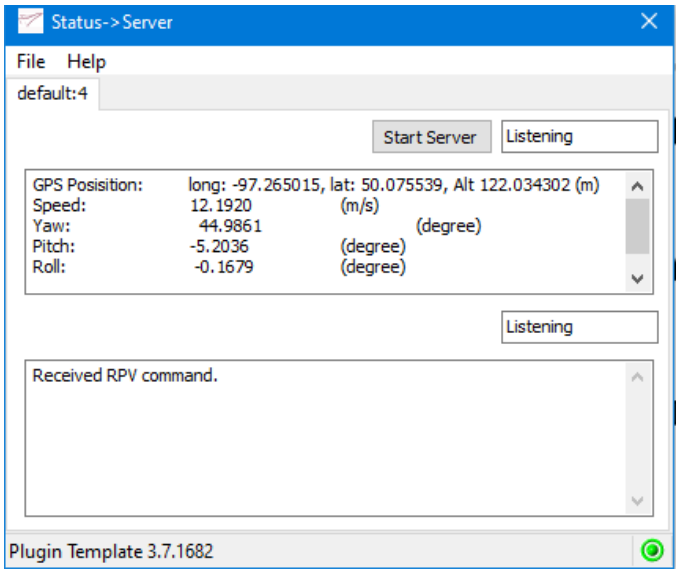


Fig. 3.10. GUI of the Horizon plugin running besides HORIZON^{mp}.

It is developed by using Micropilot Xtender™ software development kit (SDK). In terms of UAV attitude access and control, the plugin uses the provided application programming interface (API) to communicate with the UAV object, an abstraction of the actual UAV. Since the telemetry data update frequency is set to 30 Hz in this AAT-1200 with negligible data access latency over a consistent localhost TCP/IP connection, it is much faster than the radar detection rate of 10 Hz.

Therefore, the first data transmission task is achieved by broadcast. A persistent client socket is generated per successful connection request and pushed to a queue. The UAV attitude information is sent to every client in the queue at a frequency of 30 Hz.

The second command and control task is achieved by a persistent server socket with a different port number which is dedicated to command and control of UAV.

3.4.3 Radar Simulator

A simulated radar sensor is designed and built to be a part of the DAA system simulation environment. A brief structure chart of the simulated radar sensor is shown in Fig. 3.11, which illustrates the data flow of radar data generation. The radar interface is the top-level module that interfaces with and transmits simulated airborne data to the presented DAA system. The real-time aircraft attitude data from the autopilot system is received by the drone data access module. The object data in the world coordinates are generated by the object data generation module based on predefined linear motion parameters and the system timestamps. Both object data and aircraft attitude are passed to the data point computation module to compute the simulated data points coming out of the airborne radar.

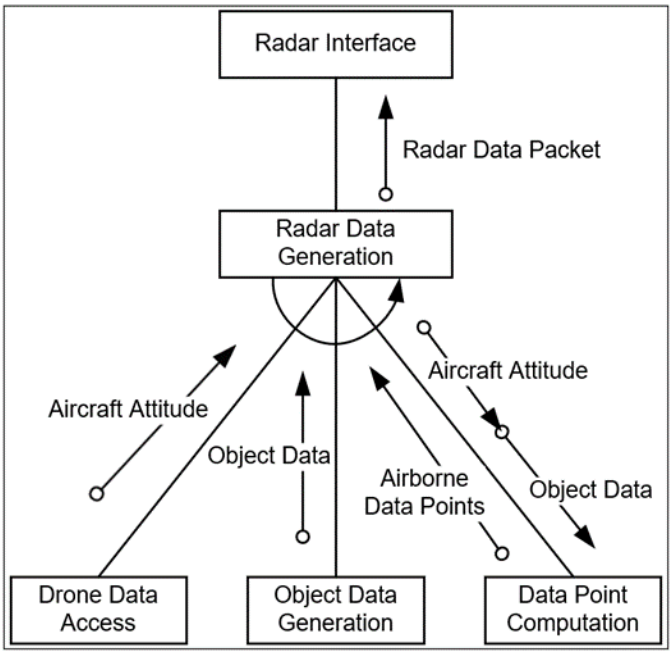


Fig. 3.11. The structure chart of the simulated radar sensor.

In order to simulate data points coming out of the airborne radar, an inversed transformation takes place. The transformation converts the simulated obstacles in ENU coordinate system into the local Cartesian coordinate system, which is

$$p' = q^{-1}pq \tag{3.21}$$

where

$$p = \begin{bmatrix} 0 \\ east \\ north \\ up \end{bmatrix} \tag{3.22}$$

With Eq. 3.21 and predefined location, velocity, and heading information of the obstacles in the world coordinates, moving obstacles with constant velocity and heading in the local Cartesian coordinate system are generated for airborne radar data simulation.

3.4.4 DAA GUI

The DAA GUI design is a collective work done by AIRvanced Technology Inc. and me, and its implementation is mainly contributed by AIRvanced Technology Inc. It is briefly presented in this thesis only to help the reader understand the presented DAA system.

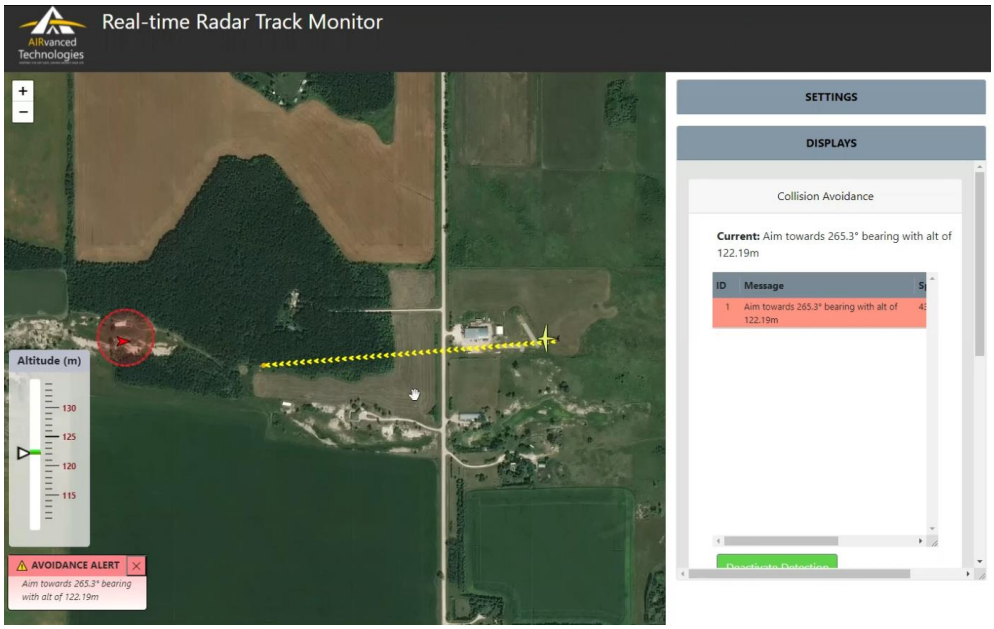


Fig. 3.12. A screenshot of the DAA GUI. The DAA system is running together with a SWIL simulated UAS and a radar simulator.

3.4.5 DAA Server

The DAA server is a multi-threaded program running on the GCS. It actually consists of the following major classes and modules, as shown in Fig. 3.13. In the interfacing module, there are three threads constantly running and interfacing with airborne radar and HORIZON^{mp} plugin. By looking at their names, two threads are in charge of receiving the UAV attitude information and transmitting UAV control commands; one radar thread is in charge of interfacing with radar using a given API. The DAA thread uses environment information to compute obstacle avoidance trajectories and corresponding control commands or advisories in real-time. The detailed algorithmic design is introduced in Sec. 4.2. The HTTP server thread responds to different types of requests sent from the DAA GUI for DAA visualization and UAS status monitoring. The system state control thread is in charge of controlling the phase of operation of the presented DAA system.

Lastly, the system status & data module that bi-directionally associates with all displayed classes and modules consists of variables and parameters updated by and required for all associated threads.

In consideration of DAA system operation as well as thread management in BVLOS operation, finite state machine (FSM) based system operation design is proposed as shown in Fig. 3.14 and Table 3.1. In each thread, it not only needs to handle the assigned tasks described in the previous paragraph but also, at the beginning of each iteration, it checks its state consistency, as shown in the pseudocode below as part of the system operation design.

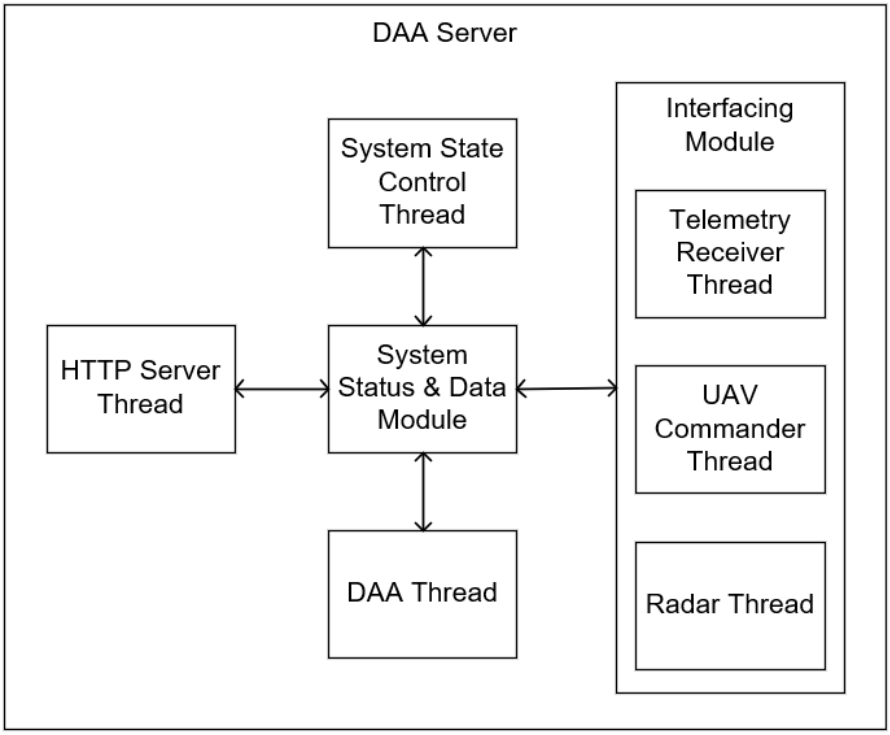


Fig. 3.13. A brief class diagram of the DAA server. For readability purposes, names displayed are not the exact class or module names used.

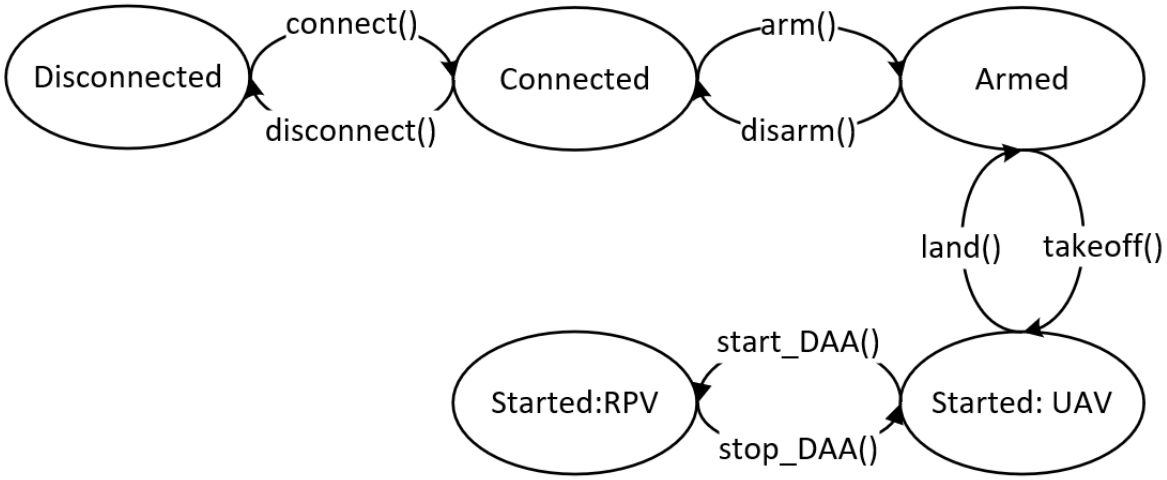


Fig. 3.14. System state transition of the DAA system.

Algorithm 1: System state transition and synchronization

```
while actual_state != expected_state
    try
        if actual_state > expected_state
            actual_state = transit_back (actual_state);
        else
            actual_state = transit_forward (actual_state);
    catch (error)
        throw(error)
end
threading.event(actual_state) *
```

* threading.event alerts to the system control thread that this thread is at desired states, or the state transition is successful, so that system control thread knows when to proceed the next system control action.

TABLE 3.1. Module states at every system state.

System State	Module	Module expected state
Disconnected	Radar Thread	disconnected
	Horizon Threads	disconnected
	DAA Thread	disarmed
	HTTP Server Thread	stopped
Connected	Radar Thread	connected
	Horizon Threads	connected
	DAA Thread	disarmed

	HTTP Server Thread	stopped
Armed	Radar Thread	started
	Horizon Threads	connected
	DAA Thread	armed
	HTTP Server Thread	started
Started: UAV Mode	Radar Thread	started
	Horizon Threads	connected
	DAA Thread	detecting
	HTTP Server Thread	started
Started: RPV Mode	Radar Thread	started
	Horizon Threads	connected
	DAA Thread	commanding
	HTTP Server Thread	started

In addition, synchronization is required to be part of the design of the presented DAA server by considering the following real-time read and write situation of the data stored in the system status & data module only for DAA computation:

1. HTTP server, DAA and system state control threads read the data at a constant frequency of 10Hz;
2. Every data element is only written by one thread in the interfacing module at a constant frequency of 10Hz or 30Hz; and
3. The write operations for some data elements (such as json variable access and assignment) are not atomic.

By considering the actual read and write requirements, readers–writer (RW) lock, a lightweight solution for this scenario, is used. This approach allows concurrent read operation by all non-interfacing modules, and write operation requires exclusive access.

3.5 Summary of Chapter 3

In this chapter, a DAA system that consists of airborne radar, distributed software systems running on GCS, and datalink is presented in the aspects of hardware used and software design. In addition, quaternion-based computation for airborne sensor data interpretation and generation is also presented. The design and development of the real-time collision avoidance module are presented in Ch. 4.

IV. REAL-TIME COLLISION AVOIDANCE

4.1 Overview

This chapter presents the approach taken or designed for real-time collision avoidance. In Sec. 4.2, the collision cone approach, which has been integrated into the presented DAA system, is presented. Then, in Sec. 4.3, the NMPC approach that formulates the problem as a nonlinear optimal control problem is presented. Lastly, a two-layered NMPC architecture that incorporates the feature of the previous two approaches is presented.

4.2 Collision Cone Approach

4.2.1 Motivation

A reactive approach for collision avoidance within the field of view of the airborne radar is an efficient starting point. This section presents a low-latency collision cone based approach for collision avoidance of dynamic obstacles.

4.2.2 Collision Geometry

The collision avoidance algorithm is activated and constantly executed when the measured time of closest approach and distance of closest approach of a tracked obstacle is less than the predefined threshold values.

Since the radius of collision volume (500ft) is about one hundred times larger than the radius of the UAV, the collision geometry between the own aircraft and the detected obstacle is represented as a collision geometry between a point and a sphere, as shown in Fig. 4.1.

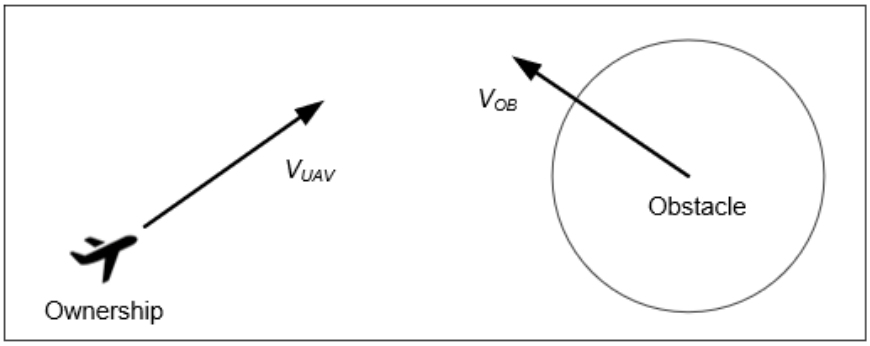


Fig. 4.1. Collision geometry between the UAV and the detected obstacle. The UAV is expected not to enter the sphere of the obstacle as the center.

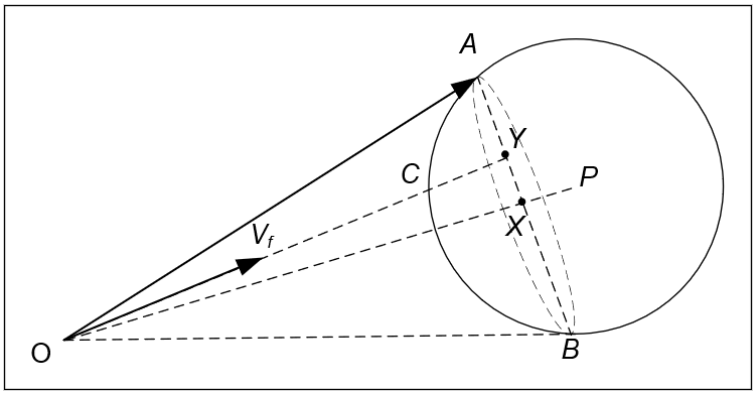


Fig. 4.2. The graphical illustration of the collision cone when the activation criteria are met. V_f is the velocity vector of the UAV relative to the obstacle, $(V_{UAV} - V_{Obs})$.

4.2.3 Algorithms

The core idea of this collision avoidance approach is to use the relative position of the obstacle with respect to the UAV and the velocity of the UAV with respect to the obstacle to constructing a collision cone to compute the desired trajectory $\overrightarrow{OA}$, in Fig. 4.2. $\overrightarrow{OA}$ has the following properties:

1. It is tangent to the sphere of the obstacle; and
2. It intersect with the ray $\overrightarrow{\mathrm{XY}}$, i.e. it is the tangent closest to the original trajectory $\overrightarrow{\mathrm{OC}}$.

The collision cone can be defined by $\overrightarrow{OX}$ and r, where X is the center of the base and r is the radius of the base and O is the apex. Using trigonometry, $\overrightarrow{OX}$ and r can be expressed as

$$\overrightarrow{OX} = \widehat{OP} * \frac{\|\overrightarrow{OP}\|^2 - {r_o}^2}{OP} \tag{4.1}$$

$$r = \sqrt[2]{(\|\overrightarrow{OP}\|^2 - {r_o}^2) - \|\overrightarrow{OX}\|^2} \tag{4.2}$$

where $\overrightarrow{OP}$ and r_o are given. $\overrightarrow{OP}$ is the relative position of the obstacle and r_o is the radius of the sphere of the obstacle.

The estimated time of the UAV reaching point Y, the intersection point at the base of the collision cone is given as

$$t = \frac{\|\overrightarrow{OX}\|}{\widehat{OX} \cdot V_f} \tag{4.3}$$

where point X is the center of the base of the collision cone and V_f is the velocity vector of the UAV with relative to the obstacle. Hence, the position of the intersection point, $\overrightarrow{OY}$ is given as

$$\overrightarrow{OY} = V_f * t \tag{4.4}$$

Therefore, the direction of the desired relative velocity vector is the unit vector of $\overrightarrow{OA}$, which can be expressed as

$$\overrightarrow{OA} = \overrightarrow{OX} + \mathrm{r} * \frac{\overrightarrow{OY} - \overrightarrow{OX}}{\|\overrightarrow{OY} - \overrightarrow{OX}\|} \tag{4.5}$$

The UAV can change either one or both its speed and its heading to adjust its velocity vector to be equal to $\overrightarrow{OA}$. The precise strategy to adopt would depend on the acceleration limits, kinematic constraints, and the time within which the UAV should pull its heading out of the collision cone [32]. In this paper, the cruise speed is assumed to be constant; hence the desired absolute velocity vector in the world frame, $V^w_{desired}$, can be calculated by solving the equation

$$\left|\widehat{OA} * |V^r_{desired}| + V_{Obs}\right| = |V^w_{desired}| \tag{4.6}$$

where $V^r_{desired}$ is the desired relative velocity vector and $V^w_{desired}$ is the desired absolute velocity vector.

4.2.4 Simulation Results and Discussion

A series of test vectors from the DO-365, Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems [58] are modified and used to test the collision avoidance advisory functionality of the presented DAA system in a software-in-the-loop (SWIL) simulation environment with simulated airborne radar as shown in Fig. 4.3.

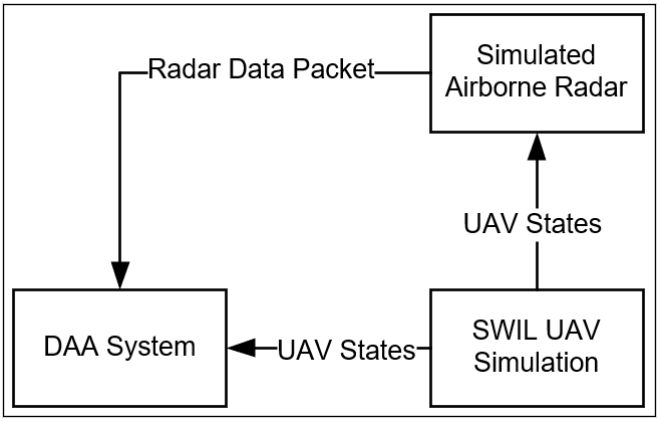


Fig. 4.3. A simplified data-flow diagram of the simulation environment

Because the modified MOPS tests are mainly conducted by Aurora Aerial Inc. [14], only a few typical test results conducted by me are displayed, as shown in Figs. 4.4 and 4.5.

As shown in Figs. 4.4 and 4.5, the desired trajectories are advised to the ground control station operator through alert messages as well as a graphical representation. In Fig. 4.5, with a head-on encounter with a moving obstacle, the estimated point of the closest approach is located between the own aircraft and the approaching obstacle, resulting in a shorter desired trajectory and greater desired turn.

Fig. 4.4. Collision avoidance advisories to a static obstacle.

Fig. 4.5. Collision avoidance advisory to an approaching obstacle. The aiming point is between the obstacle and the own aircraft, and it is fixed throughout the flight. See video in DAA_Webpage_Demo.mp4 attached).

Considering the future compatibility to input from cooperative sensors, no field of view is applied to the simulated airborne radar so that the collision avoidance advisory is fully tested with the given test vectors.

The presented collision avoidance approach is able to provide expected alerts in real-time with execution time of less than 1ms to generate the desired trajectory.

4.2.5 Summary of Section 4.2

In this section, a collision cone based dynamic obstacle avoidance is presented theoretically and by simulation. The implementation is used in Sec. 4.4 for the planning of flight path used by the high-level MPC.

4.3 Nonlinear MPC Approach

4.3.1 Motivation

The previous reactive collision cone approach has the following major limitations:

1. It is unable to guarantee a dynamically feasible trajectory, i.e., there is a snap in the desired heading; and
2. The generated trajectory is sub-optimal in the FOV of the airborne.

Therefore, more advanced motion planning algorithms needs to be applied. By considering the existence of a "sliding and limited-size" map generated by airborne radar as well as the requirement of visualization of a dynamically feasible collision avoidance trajectory, MPC is then studied and applied.

Section 4.3 presents an optimal control formulation of the path planning problem using a simplified non-linear dynamic of the quadrotor to find the optimal trajectory with dynamic obstacles in real-time.

4.3.2 Dynamic Model

Instead of using a complete non-linear dynamic model of the UAV [44], [59], which reduced the problem to low speed, a simplified approach that incorporates the differential flatness of the quadrotor is adopted as the starting point [47].

Since the quadrotor dynamics with the four inputs is differentially flat [44], the states and inputs can be written as algebraic functions of four carefully selected outputs and derivatives [44]. The flat outputs are given by

$$\sigma = [x, y, z, \psi]^T \tag{4.7}$$

where $r = [x, y, z]^T$ represents the coordinates of the center of mass in the world frame and ψ is the heading.

The control inputs of the model are containted in the vector

$$u = [u_x, u_y, u_z, u_\psi]^T \tag{4.8}$$

The input consists of sideward, forward, upward, and angular velocity inputs. Then, the state vector becomes

$$X = [x, y, z, \psi, v_x, v_y, v_z, v_\psi]^T \tag{4.9}$$

The nonlinear model $\dot{X} = f(x, u)$ is defined as:

$$\dot{x} = v_x \cos(\psi) - v_y \sin(\psi) \tag{4.10}$$

$$\dot{y} = v_x \sin(\psi) + v_y \cos(\psi) \tag{4.11}$$

$$\dot{z} = v_z \tag{4.12}$$

$$\dot{\psi} = v_\psi \tag{4.13}$$

$$\dot{v_i} = \frac{-v_i + k_i u_i}{\tau_i}, \quad i \in \{x, y, z, \psi\} \tag{4.14}$$

where k_i is defined by the step response tangent method [60] and the values are shown in Table 4.1.

TABLE 4.1. First order model parameters of the DJI-M100 platform [47].

	k_i	τ_i(s)
x	1	0.8355
y	1	0.7701
z	1	0.5013
ψ	π/180	0.5142

For computation efficiency, the Jacobian matrices with X and u as input variables are calculated respectively

$$\left[\frac{\partial \dot{\mathrm{X}}}{\partial \mathrm{X}_1} \quad \cdots \quad \frac{\partial \dot{\mathrm{X}}}{\partial \mathrm{X}_8}\right] = \begin{bmatrix} 0 & 0 & 0 & -v_x sin(\psi) - v_y cos(\psi) & cos\,(\psi) & -sin(\psi) & 0 & 0 \\ 0 & 0 & 0 & v_x cos(\psi) - v_y sin(\psi) & sin\,(\psi) & cos(\psi) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1/\tau_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/\tau_y & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1/\tau_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/\tau_\psi \end{bmatrix} \tag{4.15}$$

$$\left[\frac{\partial \dot{X}}{\partial u_1} \quad \cdots \quad \frac{\partial \dot{X}}{\partial u_4}\right] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k_x/\tau_x & 0 & 0 & 0 \\ 0 & k_y/\tau_y & 0 & 0 \\ 0 & 0 & k_z/\tau_y & 0 \\ 0 & 0 & 0 & k_\psi/\tau_z \end{bmatrix} \tag{4.16}$$

4.3.3 Optimal Control Formulation

The goals of the trajectory generated by the NMPC are:

1. Optimal fuel consumption;
2. Tracking of the reference trajectory (preplanned path); and
3. Real-time dynamic obstacle avoidance.

Since the kinetic energy consumption is proportional to the velocity squared, the cost function term for optimal fuel consumption can be written as

$$J_{feul} = w_1 \sum_{i=0}^{P-1} \sum_{j=1}^{4} {u_{i,j}}^2 \tag{4.17}$$

For trajectory tracking, the variance of the predicted states and its reference states is used. The reference states are generated from the preplanned flight path. In general, an autonomous

flight plan consists of a list of waypoints, desired cruise speed, and desired altitude. Therefore, the reference trajectory used in this optimization problem is an interpolation of the autonomous flight plan. As a result, different weights are applied for the variance of the positions and the variance of the UAV attitude, as shown in the equation below

$$J_{ref} = w_2 \sum_{i=0}^{P-1} \sum_{j=1}^{3} (X_{i,j} - Ref_{i,j})^2 + w_3 \sum_{i=0}^{P-1} \sum_{j=5}^{8} (X_{i,j} - Ref_{i,j})^2 \quad (4.18)$$

where $X_{i,j}$ is the j[th] element in state vector X at time step i within the prediction horizon of size P.

For dynamic obstacle avoidance, the moving obstacles are formulated as below to comply with the airborne radar detection in Sec. 3.3

$$0 = [x, y, z, v_x, v_y, v_z, radius]^T \quad (4.19)$$

where radius is the radius of the obstacle sphere and $r_{obs} = [x, y, z]^T$ is the center of mass of the obstacle in the world frame. In this scenario, soft constraints are used since entering the sphere does not represent the occurrence of a collision, so that the planner is able to generate feasible solutions in tight situations by slightly violating the collision sphere. The core idea of this obstacle avoidance is to ensure at any time step i, we have

$$\frac{\|r_i - r_{obs,i}\|}{radius} + s * e \geq 1, i \in [1, P-1] \quad (4.20)$$

where e is known as the slack variable that relaxes the constraints and s is the sensitivity term used for possible multiple soft constraints in the problem. For computation efficiency and maximization of the growth rate of the slack variable, the collision avoidance constraint is written as

$$-\frac{(r_i - r_{obs,i})^2}{radius^2 - s*e} + 1 \leq 0, i \in [1, P-1] \quad (4.21)$$

Therefore, the corresponding cost term is

$$J_{slack} = w_4 * e^2 \quad (4.22)$$

As a result, the corresponding optimal control problem is

$$\min_{X,U} J = J_{feul} + J_{ref} + J_{slack} \tag{4.23}$$

subject to

$$-\frac{(r_i - r_{obs,i})^2}{radius^2 - s*e} + 1 \leq 0 \; for \; all \; i \; \in [0, P-1] \tag{4.24}$$

$$|u_i| \leq \; u_{max} \; for \; all \; i \; \in [0, P-1] \tag{4.25}$$

$$|\sigma_i| \leq \; \sigma_{max} \; for \; all \; i \; \in [0, P-1] \tag{4.26}$$

4.3.4 Flight Scenario

The considered general scenario is the last-mile delivery or inspection applications for a small UAS equipped with an airborne radar with a minimum detection range of 700 m and a field of view of 120˚ in azimuth.

The flight mission of the UAV is to fly through a preplanned path at a constant cruise speed. According to general UTM solutions [61], the preplanned path consists of straight flight paths. 10-20 m/s is also a common cruise speed range for UAS operation.

Considering the maximum altitude of 400 feet or 122 meters AGL for current RPAS operation in Canada, the targeted threats are mainly other small UAVs and birds. The radius of the collision volume is proposed to be 50 meters.

In summary, the simplified testing scenario is that a small UAV is to fly through a straight preplanned path at a maximum cruise speed of 10 m/s and a maximum angular velocity of 150 ˚/s. Intruders are converging from the side of the own aircraft at the same speed. A MAC will occur on the preplanned path if no actions are taken.

4.3.5 Simulation Results and Discussion

MATLAB Model Predictive Control Toolbox has been used to implement the presented NMPC planner. The architecture of the presented approach simulation is shown in Fig. 4.6. The simulation algorithm is a common simulation design as shown below

Algorithm 2: Simulation environment
Input: Obstacle list $\{O_i\}$; initial own aircraft states X_0 and outputs Y_0; reference trajectory Xref
for each i = 1 to 50s $Yest_i \leftarrow$ measure_UAV_output (noise ,Y_0) $Xest_i \leftarrow$ correct (EKF, Y_i) $Xref_i \leftarrow$ segment (Xref, i) u_i, $trajectory_i \leftarrow$ nlmpcmove ($Xest_i$, $Xref_i$) $Xest_{i+1} \leftarrow$ predict (EKF, u_i) X_{i+1}, $Y_{i+1} \leftarrow$ move (X_i, u_i) $\{O_{i+1}\} \leftarrow$ move ($\{O_{i+1}\}$) end

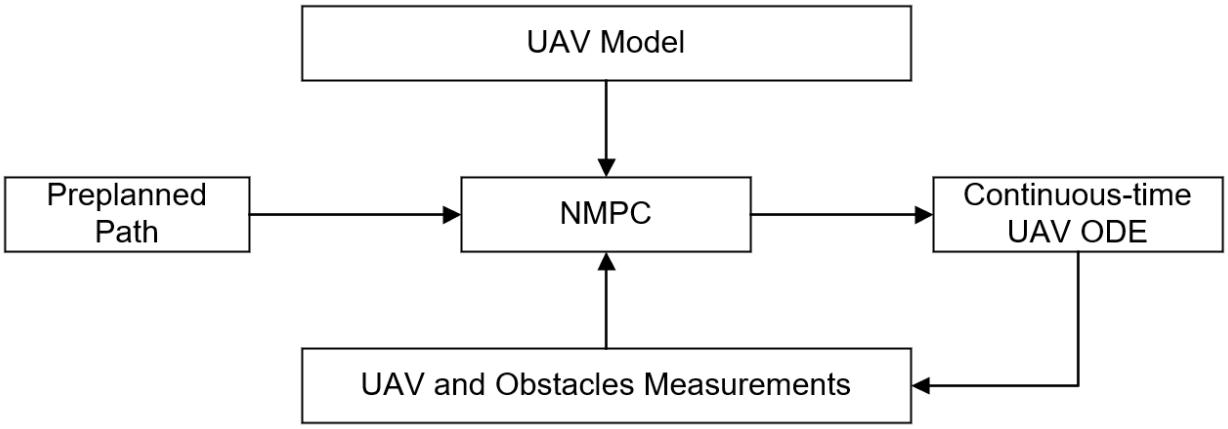


Fig. 4.6. The NMPC architecture.

Despite that, the simulation design is able to take into account the uncertainty of measurements, the uncertainty of measurement is not studied in the thesis, and all the measurements are the true

values obtained from the continuous-time ordinary differential equation (ODE) of the UAV model. The implementation parameters are shown in Table 4.2.

TABLE 4.2. NMPC Parameters and Weights.

Prediction horizon	20s
Control horizon	10s
Discretization steps	10
Optimization solver	fmincon with SQP algorithm
w_1	0.1
w_2	0
w_3	0.1
w_4	1e5
s	1

In a non-convex solution space, the performance of the NMPC depends on the level of the non-convexity of the collision scenario. In the two collision scenarios are designed and presented in one test, as shown in Figs. 4.7 and 4.8, the NMPC is able to avoid the obstacles in real-time with different levels of violation of the collision avoidance constraint.

As shown in Fig. 4.9, the second collision avoidance trajectory shows the solver is temporarily stuck at finding the optimal solution resulting in high computational cost and significant violation of the soft constraints supported by Figs. 4.12 and 4.13.

The advantage of this approach is clear. It is able to:

1. Generate dynamically feasible trajectory to ensure safer flight; and
2. Can avoid obstacles at low latency in many cases

The dynamically feasible trajectory generated by the previously presented NMPC is only an optimal solution to its local “sliding” map. Due to the limited prediction and control horizon, the

presented NMPC is unaware of detected obstacles at a distance during optimization, resulting in the following possible outcomes:

1. The slack variables are non-zero, i.e., there are violations of constraints; and/or
2. The generated trajectory is less optimal than that of the collision cone approach.

In addition, non-convex collision avoidance constraints for NMPC can result in inconsistent computational cost and possible violation of the soft constraints.

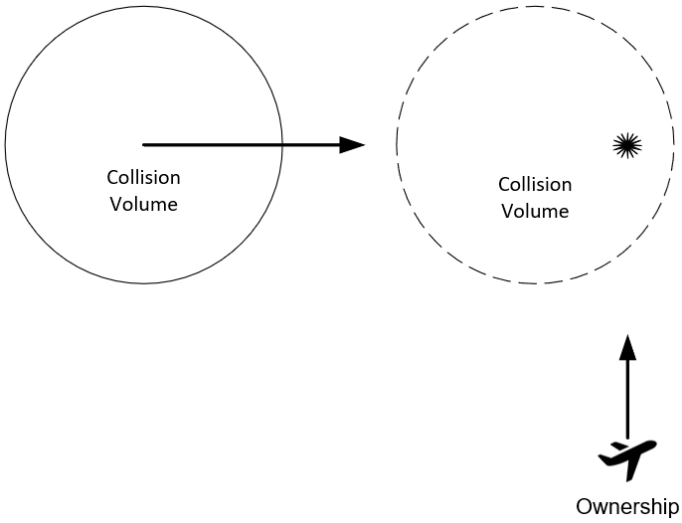


Fig. 4.7. Top view of the first collision at one side of the collision volume. The asterisk is the point at the time of the closest approach (TOCA) to the center of the collision volume, i.e. intruder.

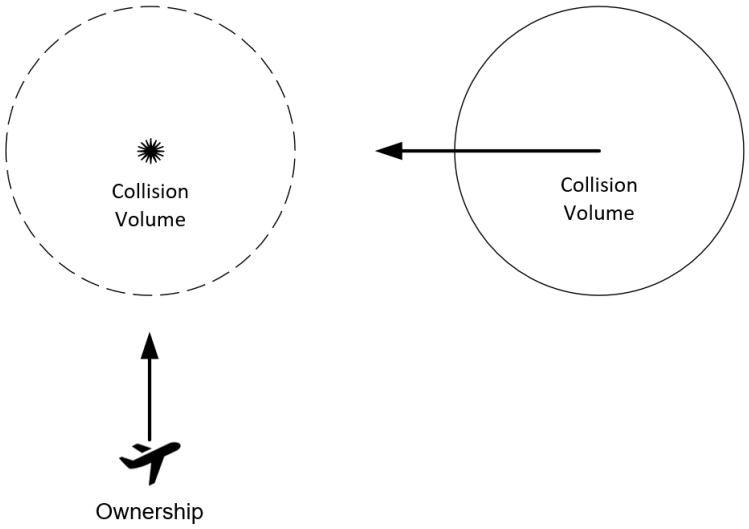


Fig. 4.8. Top view of the second collision at the center of the collision volume. The asterisk is the point at TOCA to the center of the collision volume, i.e. intruder.

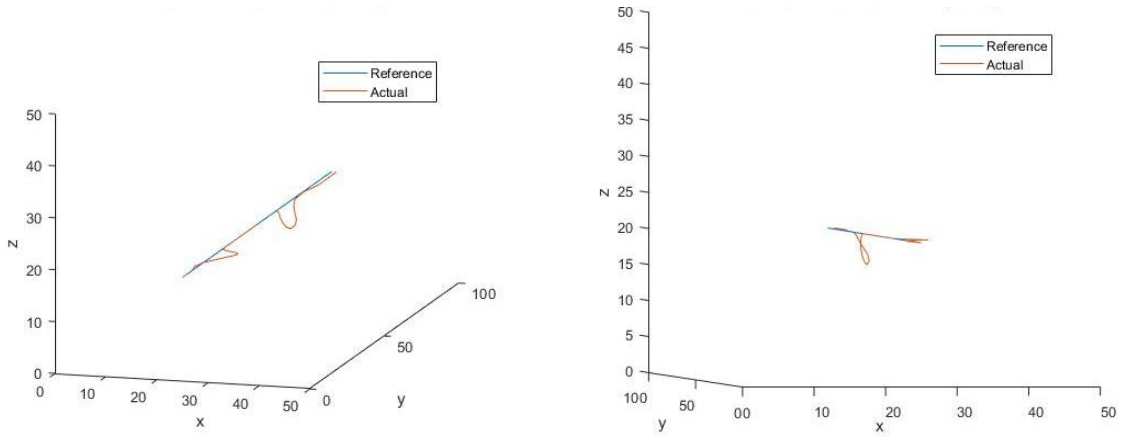


Fig. 4.9. The actual flight trajectory controlled by NMPC in different view angles.

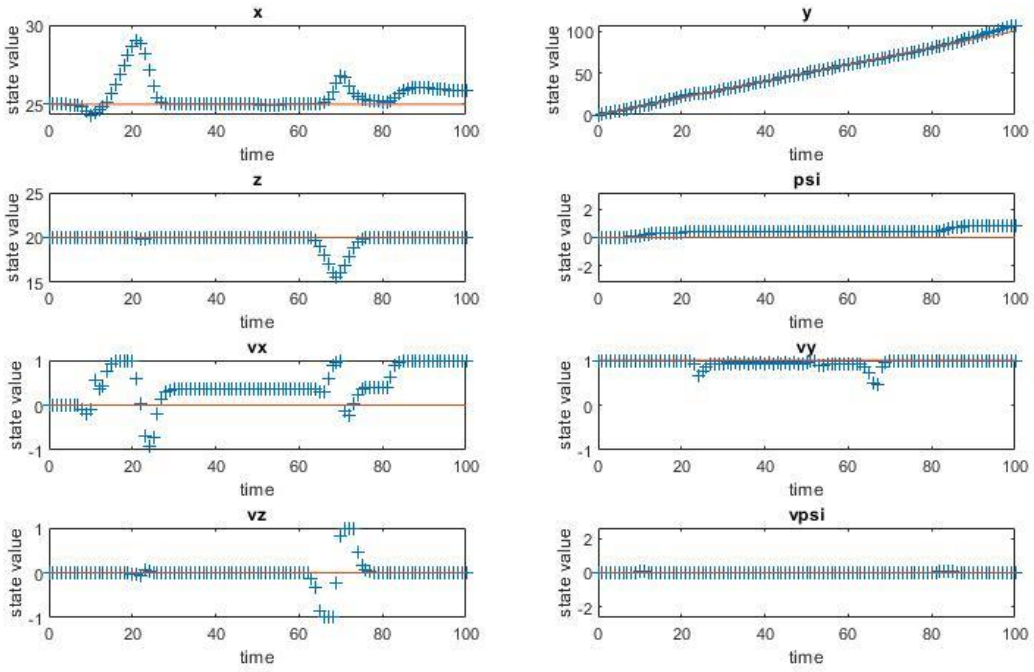


Fig. 4.10. Plots of all UAV states against time.

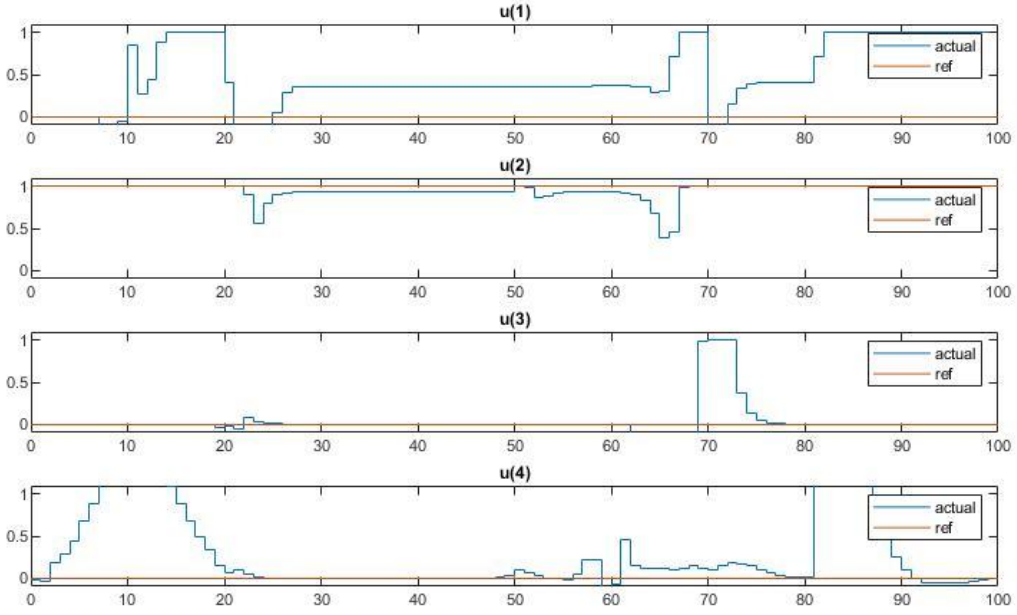


Fig. 4.11. Plots of inputs against time.

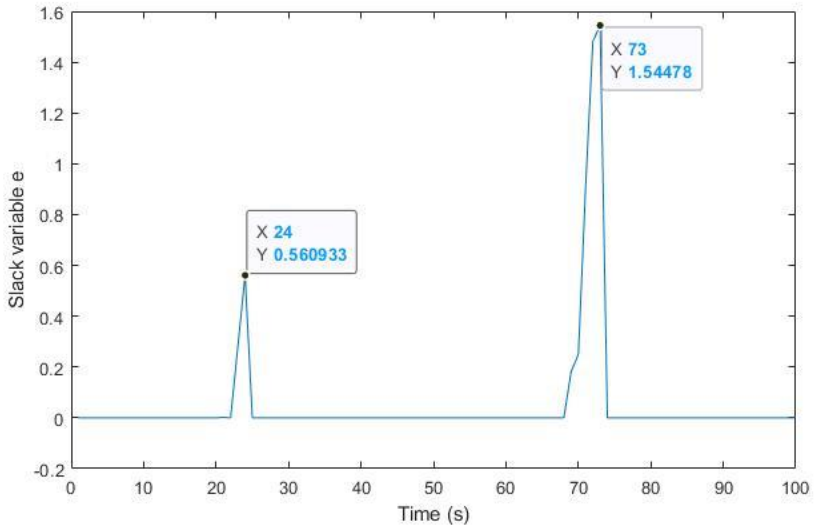


Fig. 4.12. Slack variable plot.

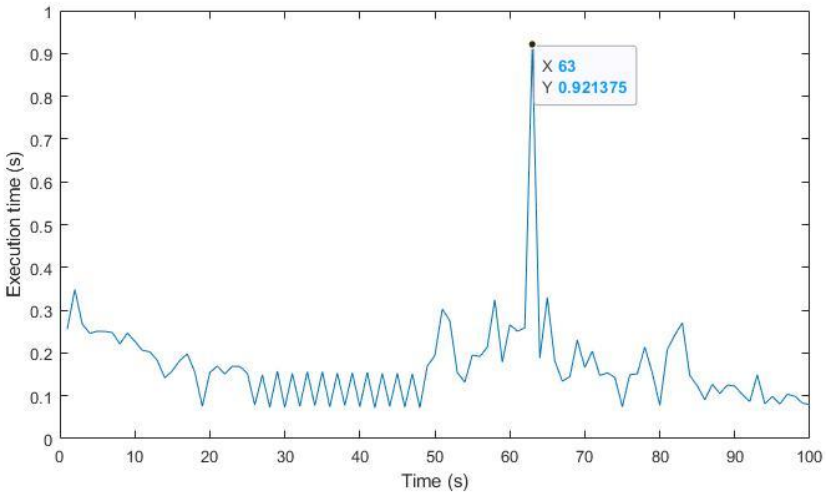


Fig. 4.13. Execution time plot.

4.3.6 Summary of Section 4.3

This section demonstrates a NMPC approach which consists of nonlinear model formulation, the optimal control problem formulation, and the simulation program used to test the performance of this approach. This implementation is modified and used in the hierarchical solution presented in Sec. 4.4.

4.4 Combined Solution: Hierarchical MPC Approach

4.4.1 Overview

In this section, the two-layered MPC approach is demonstrated theoretically and by simulation. This approach incorporates the idea of global and local planner[41] and the collision cone approach in order to solve the disadvantages of the previously presented approach in Sec. 4.3.

4.4.2 Optimization Scheme

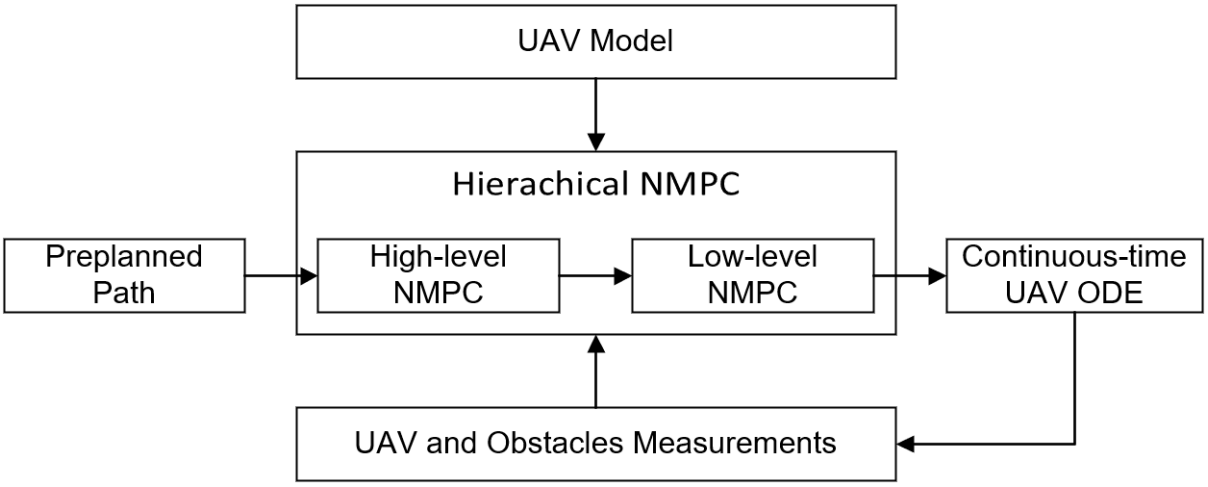


Fig. 4.14. The hierarchical NMPC architecture.

As shown in Fig. 4.14, there are two NMPCs used. The UAV model is the same as the UAV model presented in Sec. 4.3.2. The preplanned path is also generated from the same flight scenario as described in Sec. 4.3.4. For the presented non-cooperative sensor based DAA system, there is no global map available. The role of the high-level NMPC planner is to generate a trajectory in real-time to achieve, similar to that of a global planner: (i) free of collision within the map of detection, and (ii) optimal fuel consumption.

In order to achieve the objective of a collision-free trajectory, a reference path is generated by utilizing the output from collision cone approach algorithm. Firstly, the pseudocode below shows the generation of a sequence of straight trajectories.

Algorithm 3: Collision cone based path planning
Input: Obstacle list $\{O_i\}$; initial own aircraft states X_0
Output: Aiming points with timestamp $\{AP_i\}$
$X_i = X_0$ for each i = 1 to n $AP_i \leftarrow$ computeAP (O_i, X_i) $X_i \leftarrow$ updateState (AP_i) end Return $\{AP_i\}$

For each straight trajectory, the starting point is the aiming point of the previous trajectory. Then, the reference trajectory, Ref is linearly interpolated from the $\{AP_i\}$ except that the reference heading values in Ref are repeated copies from each straight trajectory. In other words, there are snaps in reference heading on the joints between straight trajectories.

Therefore, the high-level NMPC's objective is relatively straightforward: computing a dynamically feasible trajectory to smoothen the path with optimal fuel consumption. The cost function terms from Eq. 4.17 and 4.18 are used, and the optimal control problem for the high-level NMPC planner is therefore formulated as

$$\min_{X,U} J = J_{feul} + J_{ref} \tag{4.27}$$

subject to $$|u_i| \leq u_{max} \tag{4.28}$$

$$|u_i| \leq u_{max} \; for \; all \; i \in [0, P-1] \tag{4.29}$$

$$|\sigma_i| \leq \sigma_{max} \; for \; all \; i \in [0, P-1] \tag{4.30}$$

The role of this low-level NMPC planner is to achieve: (i) reference trajectory tracking, and (ii) real-time collision-free motion planning when the dynamic obstacle is at close range.

The trajectory tracking and dynamic obstacle avoidance formulations are similar to the NMPC presented in Sec. 4.3 but with different weight values tuned. For the optimal control problem is formulated as

$$\min_{X,U} J = J_{feul} + J_{ref} + J_{slack} \tag{4.31}$$

subject to $$-\frac{(r_i - r_{obs,i})^2}{radius^2 - s*e} + 1 \leq 0 \; for \; all \; i \in [0, P-1] \tag{4.32}$$

$$|u_i| \leq u_{max} \; for \; all \; i \in [0, P-1] \tag{4.33}$$

$$|\sigma_i| \leq \sigma_{max} \; for \; all \; i \in [0, P-1] \tag{4.34}$$

4.4.3 Simulation Results and Discussion

The proposed idea is to build a multi-threaded program that both planners are running in parallel at different frequencies. However, at the current stage, the performance of this approach is simulated and analyzed in a batch process. The sequence of computation is

1. Generation of trajectory with snaps in reference heading using collision cone approach;
2. Generation of a smoothened trajectory with high-level NMPC; and
3. Constant time step simulation of low-level NMPC with moving obstacles.

For step (3), the simulation process uses Algorithm 2 described in Sec. 4.3.5 as well.

MATLAB Model Predictive Control Toolbox has been used to implement the two presented NMPC planner. The implementation parameters are shown in Table 4.3 and 4.4.

TABLE 4.3. High-level MPC Parameters and Weights

Prediction horizon	30s
Control horizon	30s
Discretization steps	30
Optimization solver	fmincon with SQP algorithm
w_1	0.1
w_2	0.1
w_3	0

TABLE 4.4. Low-level MPC Parameters and Weights

Prediction horizon	20s
Control horizon	10s
Discretization steps	10
Optimization solver	fmincon with SQP algorithm
w_1	0.1
w_2	1
w_3	1
w_4	1e5
s	1

In order to evaluate the performance of the presented approach, the single NMPC approach presented in Sec. 4.3 is used as the reference approach.

The presented test scenario below is a worst-case one-to-one converging at a point collision scenario for this reference approach where the point of collision is exactly at the center of the collision sphere, where the non-convexity of the solution space is maximized.

Compared with the low-level NMPC only reference approach, the peak computational cost is much smaller, as shown in Fig. 4.15. There is no violation of the soft constraint, as shown in Fig. 4.16. In addition, the actual flight paths shown in Figs. 4.17 and 4.18 also support the observation that the new solution addresses the non-convexity issue due to the limited horizon of prediction and control in the reference approach.

The execution time for both high-level NMPC and low-level NMPC is sufficiently short for real-time operation. The computational efficiency of the high-level planner is achieved by the following reasons:

1. The path generation with collision cone techniques has the same time complexity as the collision cone approach presented in Sec. 4.2, which is hard real-time.
2. With the same horizon of prediction and control, the high-level planner has a much smaller time complexity than the low-level planner due to the lack of constraints.

This computational efficiency ensures the reference trajectory generated by the high-level planner is sufficient low-level planner. Since the airborne radar data update rate is 10Hz, both high-level and low-level NMPC planners are able to update their trajectory without missing any data packets transmitted from the radar.

Lastly, another test scenario presented below is the identical scenario with two moving obstacles presented in Sec. 4.3.5 in order to demonstrate the advantage of having a high-level NMPC for path planning. The UAV is efficiently avoiding the second moving obstacles with minimum fuel consumption in real-time as shown in Figs. 4.20, 4.21, 4.22, and 4.23.

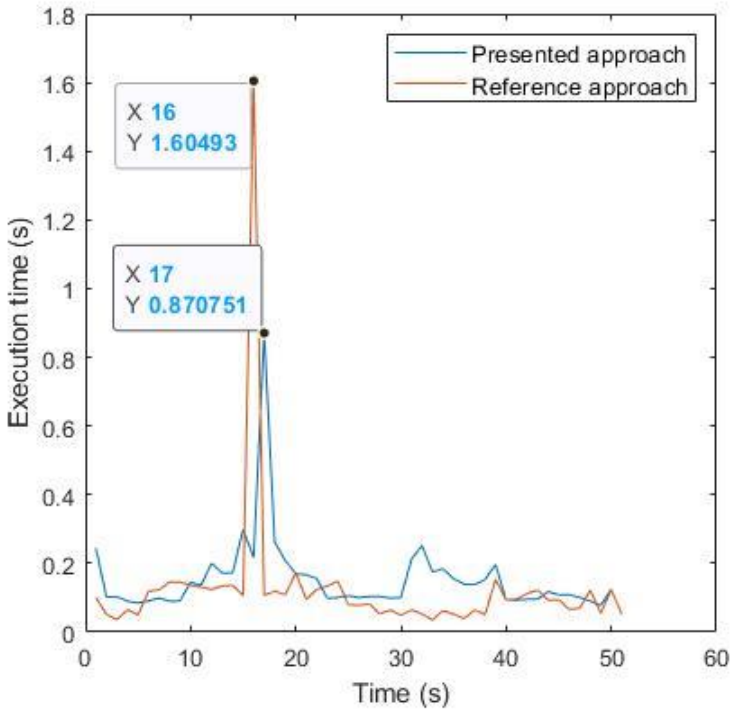


Fig. 4.15. Trajectory generation execution time of different approaches.

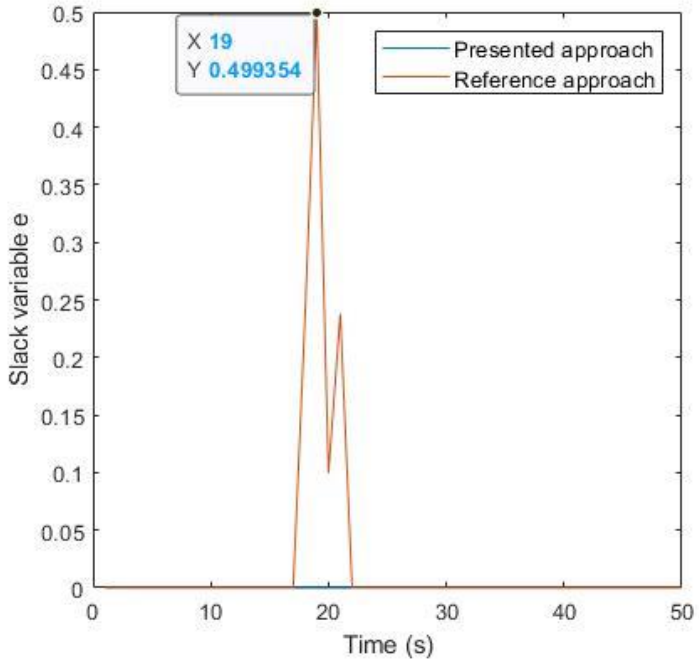


Fig. 4.16. Slack variable values of different approaches.

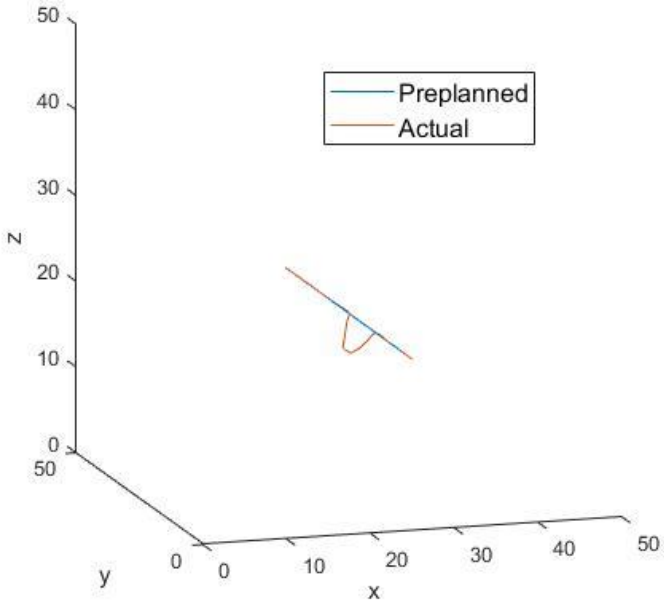


Fig. 4.17. The actual flight path generated by reference approach.

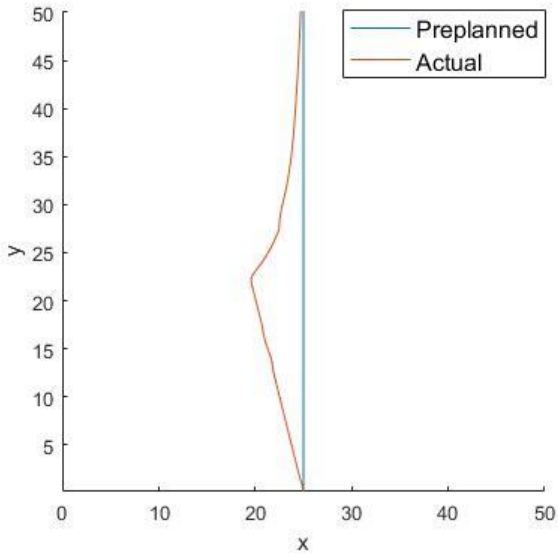


Fig. 4.18. The actual flight path generated by low-level NMPC with reference trajectory generated by high-level NMPC.

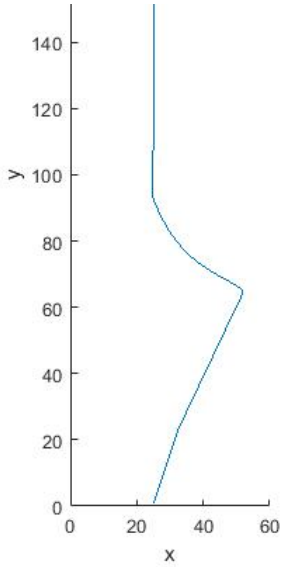


Fig. 4.19. The overall reference trajectory generated by the high-level NMPC.

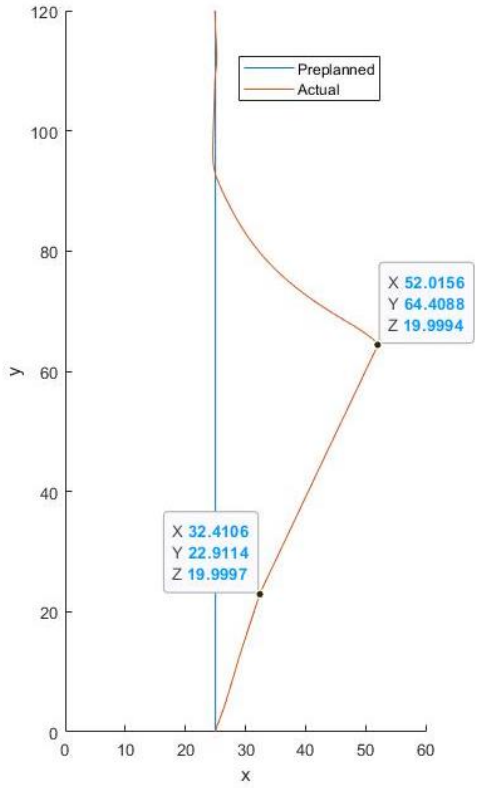


Fig. 4.20. The actual flight trajectory controlled by hierarchical MPC. The labeled two points are the computed aiming points of two moving obstacles.

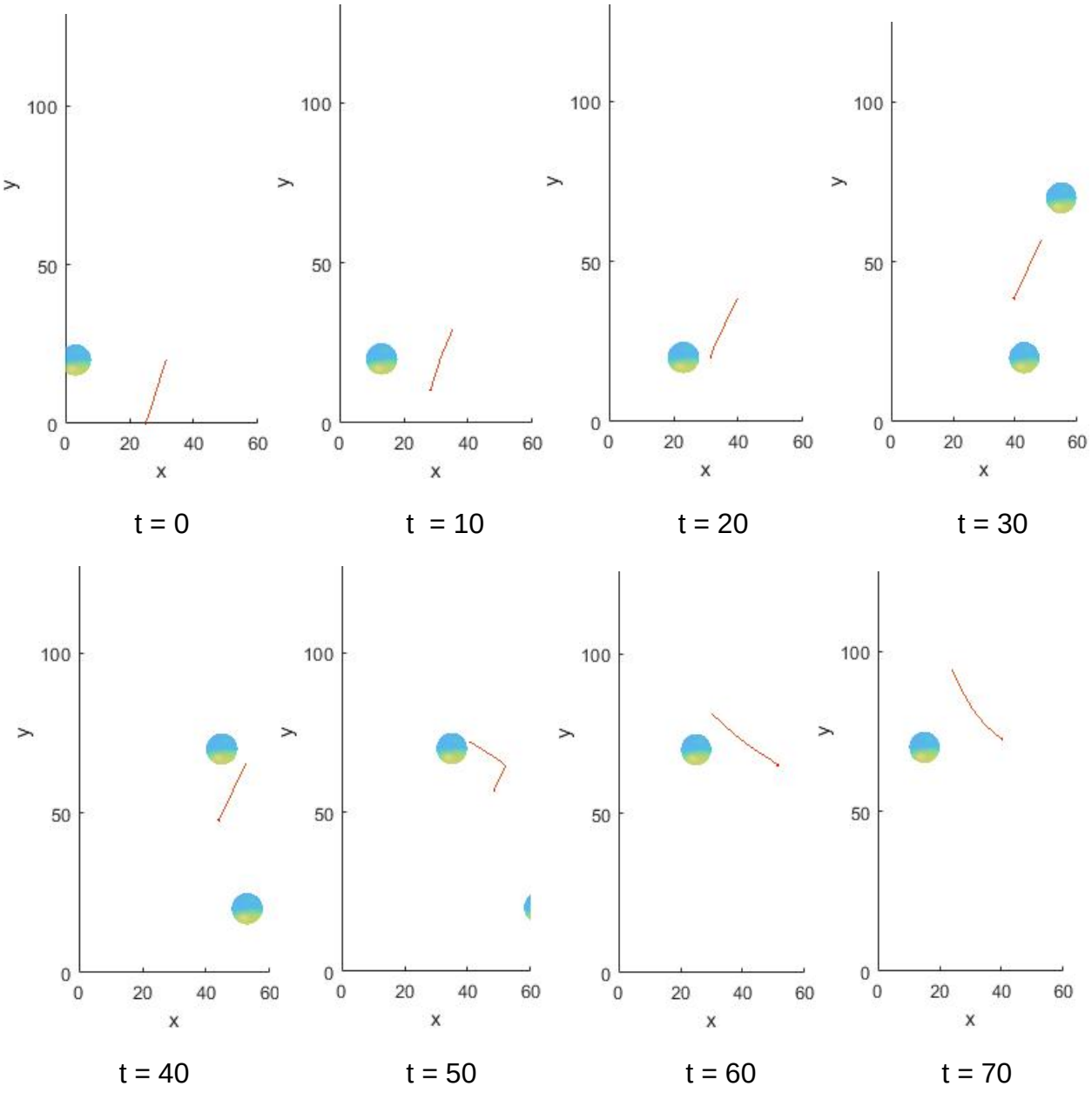


Fig. 4.21. Low-level NMPC trajectory predictions at different time steps.

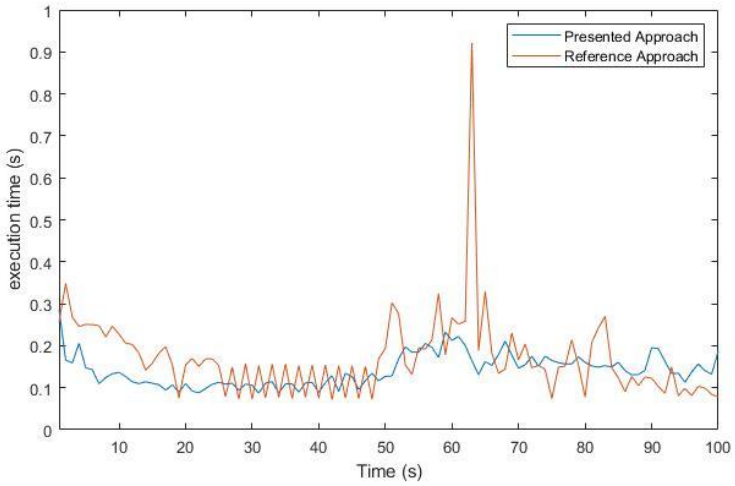


Fig. 4.22. Trajectory generation execution time of the presented approach in the scenario with two moving obstacles.

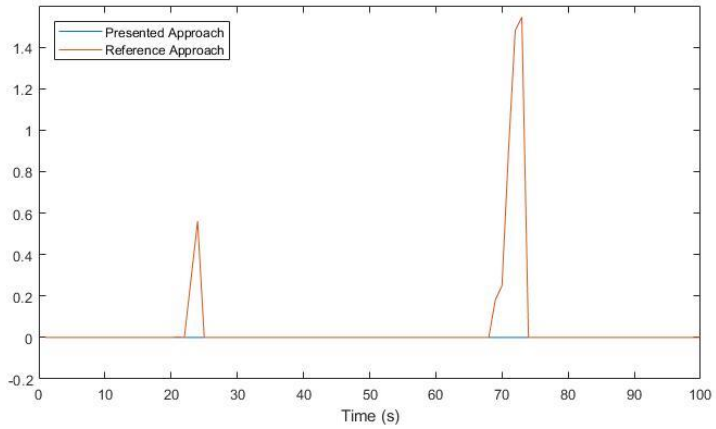


Fig. 4.23. Slack variable values of the presented approach in the scenario with two moving obstacles.

4.4.4 Summary of Section 4.4

In this section, the two-layered MPC approach for obstacle avoidance in airborne radar based DAA systems operating in a varying environment is demonstrated theoretically and by simulation. In identical flight scenarios, the presented approach can generate collision-free trajectories that overcome limitations in previous approaches, including execution time consistency and violation of the soft constraints.

V. CONCLUSIONS

5.1 Overview

This thesis presents a non-cooperative detect-and-avoid (DAA) system design for civil beyond visual line-of-sight (BVLOS) unmanned aircraft system (UAS) operations and a study of real-time collision avoidance in a varying environment. An airborne radar based non-cooperative DAA system is firstly presented in Ch. 3. Other than the system designs, quaternion-based computation for airborne radar data interpretation and generation is also presented. In Ch. 4, an iterative study of real-time collision avoidance trajectory generation is presented. In the first stage (Sec. 4.2), a reactive collision cone approach is adopted and integrated into the presented DAA system. The reactive algorithm is able to generate the desired path in real-time with no guarantee of dynamic feasibility. In the second stage (Sec. 4.3), a non-linear model predictive control (NMPC) with a differentially flat UAV model is presented. It generates dynamically feasible trajectories but the trajectory optimality and the execution time per iteration is unstable and dependent on the non-convexity of the solution space. In the last stage (Sec. 4.4), a two-layered NMPC architecture that incorporates the collision cone technique is developed for real-time collision avoidance trajectory generation. In the identical simulation scenarios, this new architecture is able to overcome the limitations observed in the previous two stages.

5.2 Thesis Conclusions

This thesis attempts to answer many interesting research questions about the non-cooperative DAA system and real-time collision avoidance as outlined in Sec. 1.2.3. This section links back the results and observations to the research questions to provide insight into the answers and potential future research. Also, this section concludes with the features of the presented DAA system that address the design problems raised in Sec. 1.1.2.

In the presented DAA system, the real-time collision detection capability achieved by the airborne radar performance and the quaternion-based collision detection computation satisfies the equivalent level of safety (ELOS) of “see and avoid” for manned aircraft.

The design and implementation of the radar simulator and the autopilot system software plugin enable the DAA system verification and validation in autopilot’s proprietary software-in-the-loop (SWIL) simulation environment.

By making use of finite state machine and thread monitoring with heartbeat event for system state synchronization and monitoring, the DAA system operation provides user-friendly interfaces and faulty recovery capabilities.

The distributed software system design, which follows object-oriented programming and the principal of command query segregation, establishes a platform that allows multiple access and control to the autopilot system for future UAS traffic management (UTM) services integration, such as flight path optimization based on weather information. In addition, such system design allows for future development for on-cloud or on-board computing to eliminate the limit on wireless datalink’s communication range. This can be achieved by migrating the interfacing module to the on-board computer and the rest threads to Amazon Web Services (AWS) Elastic Compute Cloud (EC2) units, and the datalink is then replaced by the 4G or 5G network.

For the study of real-time collision avoidance trajectory generation, three approaches are developed iteratively. It starts with the collision cone approach. One noticeable feature of this approach is the efficiency of the computation. Radar sensors transmit not only the position but also the velocity vector of the detection, which are fully utilized by matrix computation in this three-dimensional geometric collision avoidance approach. Then, an NMPC which utilizes the differential flatness of the unmanned aerial vehicle (UAV) is presented to show the computational intractability due to the non-convexity of the solution space and high computational complexity of the NMPC.

In an effort to overcome the limitation while keeping the features of the previous two approaches, a combined hierarchical solution is presented. Firstly, this approach incorporates the idea of global and local planners [41] to improve the real-time performance of the low-level

planner because the high-level planner provides efficient initial guesses that are consistent in time and near the optimal solution. For the low-level NMPC planner, the non-convexity of the problem is greatly reduced.

Secondly, this approach takes into account the properties of the airborne radar and utilizes the analytic collision cone approach in the high-level planner to efficiently generate a smoothened collision-free path in near real-time despite having no guarantee of a path with minimum fuel consumption. Therefore, if the proposed multi-threaded architecture is implemented, the shifting of the cost values in the low-level planner due to the shifting in position overtime is prevented.

Lastly, this approach also incorporates awareness of decentralized collision avoidance in a multi-agent system. Since the reference trajectory is generated by the collision cone approach and the relative velocity pair is centrosymmetric, the consensus of flying towards the tangent nearest to its relative velocity vector allows two encountering agents to self-separate from each other. Similar or derived concepts such as time-scaled collision cone and velocity obstacle are commonly used for multi-agent collision avoidance [62], [63].

5.3 Contributions

There are two major contributions of this work:

1. An airborne radar based non-cooperative DAA system of systems (SoS) for UAS BVLOS operation in a varying environment; and
2. A hierarchical NMPC based architecture for real-time obstacle avoidance trajectory generation in a varying environment

Each chapter has the following contributions:

In Ch. 3,

- Design and implementation of a distributed DAA system within a local area network for future integration of other UTM services and on-board or on-cloud computation requirements.

- Design and implementation of a DAA testing environment in a given proprietary SWIL autopilot simulation environment.
- Design and implementation of system operation and monitoring for the presented DAA system.
- Interpretation of airborne radar detections using quaternion-based computation.

In Ch. 4,

- Implementation and analysis of a reactive collision cone approach.
- Implementation and analysis of a NMPC for real-time collision avoidance in a varying environment. This NMPC takes into account the dynamic obstacle, reference trajectory tracking and fuel consumption.
- Design and implementation of a new two-layered hierarchical NMPC architecture. The architecture ensures the real-time trajectory generation with a relatively long prediction horizon for UAS at high speed. This approach incorporates the concept of global and local planners, collision cone approach, and the NMPC approach formulated in Sec. 3.3.

5.4 Limitations and Potential Solutions

For the DAA system, one major development direction is the high-fidelity simulation environment for more complex flight environments with weather and terrain information. Since the proprietary SWIL simulation environment used in this thesis does not support physical environment simulation, the establishment of a simulation environment powered by a physics engine such as Gazebo and Unreal Engine would address this problem.

In addition, due to the limited number of flight tests, the uncertainty of the radar sensor is not evaluated, and assumptions of ideal noiseless measurements are made in this thesis. One optimal solution would be to build a radar simulator in the high-fidelity simulation environment for the radar's range-velocity (RV) map and detection generation.

Not integrating the predictive approaches into the presented DAA system for proprietary software-in-the-loop simulation is another limitation of the collision avoidance study. Other than the limited time span of the project, one of the major difficulties is that the computer in control

(CIC) mode of the autopilot only takes only three input variables which is not sufficient for the adoption of the differentially flat model. An effort of clarifying the CIC control mechanics is required. Other programming difficulties include the selection and testing of optimization packages, development of multi-threaded DAA module, and modification of interfacing module for the derivation of system states that are not directly available.

For the hierarchical collision avoidance approach, other than implementing the function module in a software-in-the-loop simulation environment, improving high-level NMPC to incorporate global information such as threats from cooperative sensors and terrain information is a major future work focus. One possible approach is to construct a sampling-based path planner such as developing a “less random” RRT* tree with collision cone technique to improve the optimality and efficiency of high-level path planning.

REFERENCES

[1] Jovan Boskovic, Joseph A. Jackson, and Raman K. Mehra, “Sensor and tracker requirements development for sense and avoid systems for unmanned aerial vehicles,” in *Proc. AIAA MST Conf.*, Aug. 2013. {DOI: 10.2514/6.2013-4971}

[2] Witold Kinsner, *Fractal and Chaos Engineering: Monoscale, Multiscale and Polyscale Analyses*. Winnipeg, MB: OCO Research, February 2020, 1106 pages. {ISBN: 978-0-9939347-2-8, eBook}

[3] Witold Kinsner, *Microcontroller, Microprocessor and Microcomputer Interfacing for Real-Time Systems*. Winnipeg, MB: OCO Research, Nov 2019, 973 pages. {ISBN: 978-0-9939347-0-4}

[4] Jacob Mattingley, Yang Wang, and Stephen Boyd, "Receding horizon control," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52-65, June 2011. {DOI: 10.1109/MCS.2011.940571}

[5] Amazon Staff , “Amazon Prime Air prepares for drone deliveries.” aboutamazon.com. Jun. 2022. https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries (accessed Jul. 12, 2022).

[6] William Ralston, “Autonomous Drones Could Soon Run the UK’s Energy Grid,” wired.com. Jul. 2022. https://www.wired.com/story/autonomous-drones-could-soon-run-the-uks-energy-grid/?bxid=5bea16413f92a4046971ac19&cndid=46146261&esrc=WIRED_CRMSeries&mbid=CRMWIR092120&source=EDT_WIR_NEWSLETTER_0_DAILY_ZZ&utm_brand=wired&utm_campaign=aud-dev&utm_content=WIR (accessed Jul. 12, 2022).

[7] Stephen B. Hottman, K.R. Hansen, and M. Berry, “Literature review on detect, sense, and avoid technology for unmanned aircraft systems,” FAA, Washington, DC, USA, Rep. DOT/FAA/AR-08/41, Sept. 2009. [Online]. Avaliable: http://www.tc.faa.gov/its/worldpac/techrpt/ar0841.pdf

[8] Transport Canada. *Canadian Aviation Regulations*, SOR/96-433, 901.11 (1). [Online]. Avaliable: https://laws-lois.justice.gc.ca/eng/regulations/sor-96-433/page-110.html#h-1111650

[9] Guido Manfredi and Yannick Jestin, "An introduction to ACAS Xu and the challenges ahead," in *Proc. 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016, pp. 1-9. {DOI: 10.1109/DASC.2016.7778055}

[10] Lisa Fern, R. Conrad Rorie, Jessica S. Pack, R. Jay Shively, and Mark H. Draper, "An evaluation of detect and avoid (DAA) displays for unmanned aircraft systems: The effect of information level and display location on pilot performance," in *Proc. 15th AIAA Aviation Technology, Integration, and Operations Conference*, Jun. 2015. {DOI: 10.2514/6.2015-3327}

[11] Giancarmine Fasano, Domenico Accardo, Antonio Moccia, and David Moroney, "Sense and avoid for unmanned aircraft systems," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 31, no. 11, pp. 82-110, November 2016. {DOI: 10.1109/MAES.2016.160116}

[12] Roberto Opromolla, Giancarmine Fasano, and Domenico Accardo, "Perspectives and sensing concepts for small UAS sense and avoid," in *Proc. 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, 2018, pp. 1-10. {DOI: 10.1109/DASC.2018.8569338}

[13] Federal Aviation Administration. *Reneral Operating and Flight Rules*, 14 CFR Part 91, 113 (b). [Online]. Available: https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-91/subpart-B/subject-group-ECFRe4c59b5f5506932/section-91.113

[14] Hongru Li, Witold Kinsner, Yan Wang, Brenn Palma, and Alan Tay, "Airborne radar based collision detection and avoidance system for unmanned aircraft systems in a varying environment," in *Proc. 2021 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, 2021, pp. 43-48. {DOI: 10.1109/WiSEE50203.2021.9613842}

[15] Hongru Li and Witold Kinsner, "Hierarchical model predictive control for obstacle avoidance in airborne radar based detect-and-avoid systems operating in a varying environment," in *Proc. 2022 10th IEEE International Conference on Wireless for Space*

and Extreme Environments, WiSEE'22, (Winnipeg, MB; Oct 12-14, 2022), to be published.

[16] Hongru Li, "Software developed for the low-latency airborne collision detection and avoidance system for unmanned aircraft systems in a varying environment project," Technical Report DEL22-7, Dept. Electrical and Computer Engineering, University of Manitoba, Winnipeg, Canada, Aug. 2022. In preparation.

[17] Federal Aviation Administration. *Unmanned Aircraft Systems (UAS) Operational Approval Notice,* Jul. 2013. [Online]. Avaliable: https://fsims.faa.gov/wdocs/notices/n8900_227.htm

[18] Federal Aviation Administration. *Pilots' Role in Collision Avoidance (With Change 1),* AC 90-48D, Apr. 2016. [Online]. Avaliable: https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentid/1029428

[19] Young K. Kwag, Min S. Choi, Chui H. Jung, and Kwang Y. Hwang, "Collision avoidance radar for UAV," in *Proc. 2006 CIE International Conference on Radar*, 2006, pp. 1-4. {DOI: 10.1109/ICR.2006.343231}

[20] Lasse Klingbeil et al., "Towards autonomous navigation of an UAV-based mobile mapping system," in *Proc. 4th Intl. Conf. on Machine Control & Guidance*, 2014, pp. 136-148. [Online]. Avaliable: https://www.researchgate.net/publication/260206334_Towards_Autonomous_Navigation_of_an_UAV-based_Mobile_Mapping_System

[21] Stéphane Ross *et al*., "Learning monocular reactive UAV control in cluttered natural environments," in *Proc. 2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1765-1772. {DOI: 10.1109/ICRA.2013.6630809}

[22] Jia Hu, Yifeng Niu, and Zhichao Wang, "Obstacle avoidance methods for rotor UAVs using RealSense camera," in *Proc. 2017 Chinese Automation Congress (CAC)*, 2017, pp. 7151-7155. {DOI: 10.1109/CAC.2017.8244068}

[23] Dhiraj Gandhi, Lerrei Pinto, and Abhinav Gupta, “Learning to fly by crashing,” in *Proc. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3948-3955. {DOI: 10.1109/IROS.2017.8206247}

[24] Peter Pinggera, David Pfeiffer, Uwe Franke, and Rudolf Mester, “Know your limits: accuracy of long range stereoscopic object measurements in practice,” in *Proc. ECCV 2014*, Sept. 2014, pp. 96–111.

[25] *Introduction to TCAS II Version 7.1 Booklet*, Federal Aviation Administration, Feb. 2011. [Online]. Available: https://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS II V7.1 Intro booklet.pdf

[26] Jawad N. Yasin, Sherif A. S. Mohamed, Mohammad-Hashem Haghbayan, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila, “Unmanned aerial vehicles (UAVs): collision avoidance systems and approaches,” *IEEE Access*, vol. 8, pp. 105139-105155, 2020. {DOI: 10.1109/ACCESS.2020.3000064}

[27] Yue Qu and Wenjun Yi, “Three-dimensional obstacle avoidance strategy for fixed-wing uavs based on quaternion method,” *Applied Sciences*, vol. 12, p. 955, Jan. 2022. {DOI: 10.3390/app12030955}

[28] Jung-Woo Park, Hyondong Oh, and Min-Jea Tahk, “UAV collision avoidance based on geometric approach,” in *Proc. 2008 SICE Annual Conference*, 2008, pp. 2122–2126. {DOI: 10.1109/SICE.2008.4655013}

[29] Anusha Mujumdar and Radhakant Padhi, “Nonlinear geometric and differential geometric guidance of UAVs for reactive collision avoidance,” *Journal of Guidance, Control, and Dynamics*, vol. 34, p. 69, Jul. 2009. {DOI: 10.2514/1.50923}

[30] Amit K. Tripathi, Ramsingh G. Raja, and Radhakant Padhi, “Reactive collision avoidance of UAVs with stereovision camera sensors using UKF,” in *IFAC Proceedings Volumes*, vol. 47, no. 1, pp. 1119–1125, 2014. {DOI: 10.3182/20140313-3-IN-3024.00171}

[31] Dongwoo Lee and Joohyun Woo, “Reactive collision avoidance of an unmanned surface vehicle through Gaussian mixture model-based online mapping,” *Journal of Marine Science and Engineering* , vol. 10, no. 4. 2022. {DOI: 10.3390/jmse10040472}

[32] Animesh Chakravarthy and Debasish Ghose, “Obstacle avoidance in a dynamic environment: a collision cone approach,” *IEEE Trans. on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998. {DOI: 10.1109/3468.709600}

[33] Oussama Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Proc. 1985 IEEE International Conference on Robotics and Automation*, 1985, vol. 2, pp. 500–505. {DOI: 10.1109/ROBOT.1985.1087247}

[34] Xin Gao *et al.*, “A rapidly exploring random tree optimization algorithm for space robotic manipulators guided by obstacle avoidance independent potential field,” *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, 2018. {DOI: 10.1177/1729881418782240}

[35] D. Choi, K. Lee, and D. Kim, “Enhanced Potential Field-Based Collision Avoidance for Unmanned Aerial Vehicles in a Dynamic Environment,” in *Proc. AIAA Scitech 2020 Forum*, Jan. 2020. {DOI: 10.2514/6.2020-0487}

[36] Maxim Likhachev and Dave Ferguson, “Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles,” *The International Journal of Robotics Research*, Jun. 2009, vol. 28, no. 8, pp. 933–945. {DOI: 10.1177/0278364909340445}

[37] Michael Montemerlo *et al.*, “Junior: the Stanford entry in the urban challenge,” in *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, Martin Buehler, Karl Iagnemma, and Sanjiv Singh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 91–123. {DOI: 10.1007/978-3-642-03991-1_3}

[38] Sertac Karaman and Emilio Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, 2011, vol. 30, pp. 846–894. {DOI: 10.1177/0278364911406761}

[39] K. R. Jayasree, P. R. Jayasree, and A. Vivek, “Smoothed RRT techniques for trajectory planning,” in *Proc. 2017 International Conference on Technological Advancements in*

Power and Energy (TAP Energy), 2017, pp. 1–8. {DOI: 10.1109/TAPENERGY.2017.8397376}

[40] Jeong h. Jeon, Sertac Karaman, and Emilio Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. 2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 3276–3282. {DOI: 10.1109/CDC.2011.6161521}

[41] Omar Mechali, Limei Xu, Mingzhu Wei, Ilyas Benkhaddra, Fan Guo, and Abdelkader Senouci, "A Rectified RRT with Efficient Obstacles Avoidance Method for UAV in 3D Environment," in *Proc. 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2019, pp. 480–485. {DOI: 10.1109/CYBER46603.2019.9066691}

[42] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, April 2016. {DOI: 10.1109/TITS.2015.2498841}

[43] Sébastien Glaser, Benoit Vanholme, Saïd Mammar, Dominique Gruyer, and Lydie Nouvelière, "Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road With Traffic and Driver Interaction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 589-606, Sept. 2010. {DOI: 10.1109/TITS.2010.2046037}

[44] Daniel Mellinger and Vijay Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. 2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520-2525. {DOI: 10.1109/ICRA.2011.5980409}

[45] Charles Richter, Adam Bry, and Nicholas Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. Robotics Research: The 16th International Symposium ISRR*, Masayuki Inaba and Peter Corke, Eds. Cham: Springer International Publishing, 2016, pp. 649–666.

[46] Helen Oleynikova, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran, "Continuous-time trajectory optimization for online UAV replanning,"

in *Proc. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5332-5339. {DOI: 10.1109/IROS.2016.7759784}

[47] Manuel Castillo-Lopez, Seyed A. Sajadi-Alamdari, Jose L. Sanchez-Lopez, Miguel A. Olivares-Mendez, and Holger Voos, "Model predictive control for aerial collision avoidance in dynamic environments," in *Proc. 2018 26th Mediterranean Conference on Control and Automation (MED)*, 2018, pp. 1-6. {DOI: 10.1109/MED.2018.8442967}

[48] Abhishek Dixit, Ajay Misra, and Sanjay E. Talole, "Model predictive control based collision avoidance controller for octocopter," in *Proc. 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2020, pp. 630-635. {DOI: 10.1109/SPIN48934.2020.9071236}

[49] Erfu Yang and Dongbing Gu, "A multiagent fuzzy policy reinforcement learning algorithm with application to leader-follower robotic systems," in *Proc. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3197-3202. {DOI: 10.1109/IROS.2006.282421}

[50] Paulo Tabuada, George J. Pappas, and Pedro Lima, "Feasible formations of multi-agent systems," in *Proc. 2001 American Control Conference. (Cat. No.01CH37148)*, 2001, pp. 56-61 vol.1. {DOI: 10.1109/ACC.2001.945513}

[51] Steve Wollkind, John Valasek, and Thomas Ioerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, 2004. {DOI: 10.2514/6.2004-4992}

[52] Yongwoo Lee and Youdan Kim, "Distributed unmanned aircraft collision avoidance using limit cycle," in *Proc. 2011 11th International Conference on Control, Automation and Systems*, 2011, pp. 121-125.

[53] "AAT-1200 Technical Specifications." Aurora Aerial – Last Mile Delivery Logistics Solution. https://www.auroraaerial.aero/technology/ (accessed Aug. 3, 2022).

[54] "EchoFlight product description." EchoFlight UAV Radar – Echodyne. https://www.echodyne.com/defense/uav-radar/ (accessed July 28, 2022).

[55] "ENU Longitude Latitude relationships." Wikipedia. https://en.wikipedia.org/wiki/File:ECEF_ENU_Longitude_Latitude_relationships.svg (accessed Aug. 3, 2022).

[56] D. M. Henderson, "Euler angles, quaternions, and transformation matrices for Space Shuttle analysis," McDonnell-Douglas Technical Services Co., Inc., Houston, TX, USA, Rep. DN-1.4-8-020, 1977. Accessed: Jul. 15, 2021. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770019231.pdf

[57] Brendon Smeresky, Alexa Rizzo, and Timothy Sands, "Kinematics in the Information Age," *Mathematics*, vol. 6, no. 9, 2018. Accessed: July, 28, 2021. {DOI: 10.3390/math6090148} [Online]. Avaliable: https://www.mdpi.com/2227-7390/6/9/148

[58] *Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems*, DO-365, Special Committee 228 (SC-228), Washington, DC, USA, 2017.

[59] Anežka Chovancová, Tomáš Fico, Ľuboš Chovanec, and Peter Hubinský, "Mathematical modelling and parameter identification of quadrotor (a survey)," in *Proc. Engineering*, vol. 96, Dec. 2014. {DOI: 10.1016/j.proeng.2014.12.139}

[60] Manuel Castillo-Lopez, Miguel A. Olivares-Mendez, and Holger Voos, "Evasive maneuvering for UAVs: an MPC approach," in *Proc. ROBOT 2017: Third Iberian Robotics Conf.*, 2017, vol 693. {DOI: 10.1007/978-3-319-70833-1_67}

[61] "AirMatrix UTM services." AirMatrix. https://www.airmatrix.ca/ (accessed July 28, 2022).

[62] Natalie Brace *et al.*, "Using collision cones to assess biological deconfliction methods," *Journal of the Royal Society Interface*, vol. 13, Sep. 2016. {DOI: 10.1098/rsif.2016.0502}

[63] James Douthwaite, Shiyu Zhao, and Lyumila S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Systems*, vol. 07, Jan. 2019. {DOI: 10.1142/S2301385019400065}

APPENDIX A
SOURCE CODES AND SIMULATION PROCEDURES

This chapter briefly describes the access and execution procedures of the software for replicating the simulations presented in this thesis. A more detailed technical report (in preparation) for the software developed is [16].

Since the presented DAA system is the outcome of the Mitacs Accelerate program supported by AIRvanced Technologies Inc. and Mitacs, please contact AIRvanced Technologies Inc. for software availability.

The source codes of approaches presented in Sec. 4.3 and 4.4 are available and included in the "MATLAB-Collision-Avoidance-MPC" folder. The instructions for executing the simulations are listed below. MATLAB installed with Control System Toolbox and Model Predictive Control Toolbox is required for execution.

To run NMPC approach test scenarios presented in Ch. 4.3,

1. Open MATLAB.
2. Set current folder to */MATLAB-Collision-Avoidance-MPC*
3. run *final_main_MPC_no_cone_moving.m*
4. Press any key to continue when messages are printed in the command window

To run hierarchical MPC approach test scenarios presented in Ch. 4.4,

5. Open MATLAB.
6. Set current folder to */MATLAB-Collision-Avoidance-MPC*
7. run *final_main_MPC_cone_moving.m*
8. Press any key to continue when messages are printed in the command window

Please note that a motion replay function, *FlyingRobotPlotReplay*, is called at the end of the simulation. The motion replay can be either controlled by key presses or automatic by toggling the last Boolean parameter.

APPENDIX B
EXPERIMENTS DATA

This chapter describes the instructions for visualization of the experiments data generated by the simulations.

The actual workspace data used for both Sec. 4.3 and 4.4 are saved in the source folder as .mat files. Readers can also regenerate the data by following instructions as shown in Appendix A.

The easiest way to visualize and analyze the experiment data is to open the MATLAB script with the prefix "*demo_*". Below are two examples of the demo script:

1. *demo_two_moving_obs_MPC_cone.m* is the demonstration of the combined solution (*MPC_cone*) with a test scenario of two moving obstacles (*two_moving_obs*).
2. *demo_single_moving_obs_MPC_no_cone.m* is the demonstration of NMPC-only approach (*MPC_no_cone*) with a test scenario of one moving obstacles (*single_moving_obs*).

In each demo script, the first line of the code will load the corresponding workspace data. It is the reader's freedom to evaluate the selected lines of code, where:

1. *FlyingRobotPlotTracking* plots all UAV states and outputs.
2. *FlyingRobotPlotReplay* plays an animation of collision avoidance.