UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2020-73
The Faculty of Sciences, Technology and Medicine

DISSERTATION

Defence held on 07/12/2020 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN SCIENCES DE L'INGENIEUR

by

Min WANG
Born on 23 June 1989 in Hubei, China

ROBUST REAL-TIME SENSE-AND-AVOID SOLUTIONS FOR REMOTELY PILOTED QUADROTOR UAVS IN COMPLEX ENVIRONMENTS

Dissertation defence committee

Dr-Ing. Holger VOOS, dissertation supervisor
Professor, Université du Luxembourg

Dr Jean-Régis HADJI-MINAGLOU, Chairman
Professor, Université du Luxembourg

Dr Fulvio MASTROGIOVANNI
Professor, Università di Genova

Dr Toshiyuki MURAKAMI
Professor, Keio University

Dr Radu STATE, Vice Chairman
Professor, Université du Luxembourg

Thesis supervisor: Prof. Holger VOOS Min WANG

Robust Real-time Sense-and-Avoid Solutions for Remotely Piloted Quadrotor UAVs in Complex Environments

Abstract

UAV teleoperation is a demanding task: to successfully accomplish the mission without collision requires skills and experience. In real-life environments, current commercial UAVs are to a large extent remotely piloted by amateur human pilots. Due to lack of teleoperation experience or skills, they often drive UAVs into collision. Therefore, in order to ensure safety of the UAV as well as its surroundings, it is necessary for the UAV to boast the capability of detecting emergency situation and acting on its own when facing imminent threat. However, the majority of UAVs currently available in the market are not equipped with such capability. To fill in the gap, in this work we present 2D LIDAR based Sense-and-Avoid solutions which are able to actively assist unskilled human operator in obstacle avoidance, so that the operator can focus on high-level decisions and global objectives in UAV applications such as search and rescue, farming etc. Specifically, with our novel 2D LIDAR based obstacle detection and tracking algorithm, perception-assistive flight control design, progressive emergency evaluation policies and optimization based and adaptive virtual cushion force field (AVCFF) based avoidance strategies, our proposed UAV teleoperation assistance systems are capable of obstacle detection and tracking, as well as automatic obstacle avoidance in complex environment where both static and dynamic objects are present. Additionally, while the optimization based solution is validated in Matlab, the AVCFF based avoidance system has

been fully integrated with sensing system, perception-assistive flight controller on the basis of the Hector Quadrotor open source framework, and the effectiveness of the complete Sense-and-Avoid solution has been demonstrated and validated on a realistic simulated UAV platform in Gazebo simulations, where the UAV is operated at a high speed.

Contents

Listing of figures

To my dearest family and friends who always have faith in me!

Acknowledgments

Completing a PhD is not an easy thing to do, and during the process many things can go wrong unexpectedly. Fortunately, here I am, standing at the end of my PhD study. I owe my achievements to the following people:

Firstly, I would like to express my sincere gratitude towards my PhD supervisor Prof. Holger VOOS for his continuous support of my PhD study and related research. Without the his patience and guidance, this thesis work would not have been possible. I would also like to thank Prof. Jean-Régis HADJI-MINAGLOU and Prof. Fulvio MASTROGIOVANNI for always being present for my PhD related meetings and for their insightful comments and continuous encouragement. Prof. Fulvio MASTROGIOVANNI and his wife Dr. Suha ABUARQUB have always been my life mentors since my Master studies and I can not say enough to thank them for taking time to take part in the journey of my life, for always having faith in me and lifting me up, and for setting the role models for me to follow. My sincere thanks also go to Prof. Radu STATE and Prof. Toshiyuki MURAKAMI for taking valuable time to support my research and share your insights. Additionally, I would also like to express appreciation to my colleagues and friends who have helped and cheered me up along the way especially Dr. Daobilige SU, Elena and Guido, as well as Julie, and to the Luxembourg National Research Fund (FNR) for their financial support. Last but not least, I would like to thank my beloved family: my parents, my aunt and uncle, and my husband. You were the reason for me to continue even in the darkest days.

I thank you all very much!

Good judgment comes from experience; experience comes from bad judgment.

Rita Mae Brown

1 Introduction

Recent years have seen rising interest in Unmanned Aerial Vehicles (UAVs) from both the research community and industry. Although there are many people who believe that UAVs are a recent invention which dates back to at most two or three decades, unmanned flight actually has a rich history that goes back all the way to ancient times. Of course, the first systems that qualify the modern definition of UAVs are quite recent and mainly involve the reconnaissance drones developed and deployed during the cold war. Nowadays, UAV systems have evolved and expanded into a wide range of designs such as quadrotors, ducted fan, and blimps etc. in addition to the classic fixed-wing and helicopter systems. Modern UAVs have found their way into various civilian applications beyond the military domain, such as weather monitoring, forestry, search and rescue, surveillance, entertainment, farming etc.

This chapter provides an introduction on unmanned aerial vehicle's definition

and classification as well as a brief history of UAVs' evolution over time. Additionally, the motivation for our research problems is also presented.

1.1 Definition of UAV

Although there is no uniform definition of unmanned aerial vehicle (UAV), which is also known as drone or unmanned aerial system (UAS) or remotely piloted vehicle (RPV), and different organisations or parties define it using distinct wording or terminologies, their underlying ideas are similar [60]. For example, the definition of unmanned vehicle given in the 2007 – 2012 unmanned systems roadmap by U.S. Department of Defense is

> A powered vehicle that does not carry a human operator, can be operated autonomously or remotely, can be expendable or recoverable, and can carry a lethal or nonlethal payload. Ballistic or semi-ballistic vehicles, cruise missiles, artillery projectiles, torpedoes, mines, satellites, and unattended sensors (with no form of propulsion) are not considered unmanned vehicles. Unmanned vehicles are the primary components of unmanned systems. [46]

While the U.S. Federal Aviation Administration defined unmanned aircraft in 2008 as

> A device used or intended to be used for flight in the air that has no onboard pilot. This includes all classes of airplanes, helicopters, airships, and translational lift aircraft that have no onboard pilot. Unmanned aircraft are understood to include only those aircraft controllable in three axes and therefore, exclude traditional balloons. [1]

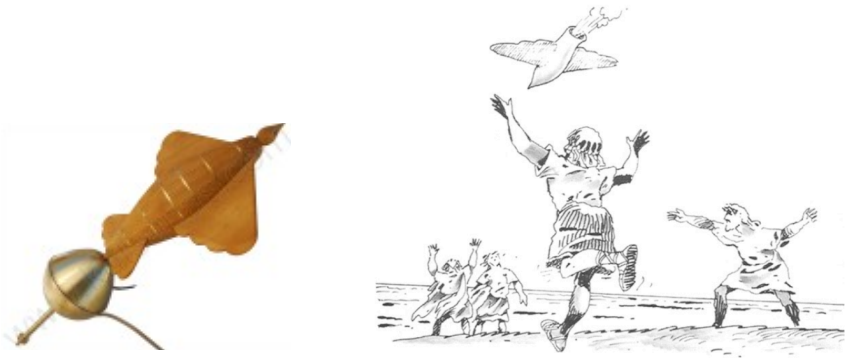
As another example, the European Aviation Safety Agency in 2009 defines UAS as

> An Unmanned Aircraft System (UAS) comprises individual system elements consisting of an "unmanned aircraft", the "control station" and any other system elements necessary to enable flight, i.e., "command and control link" and "launch and recovery elements". There may be multiple control station, command & control links and launch and recovery elements within a UAS. [2]

In other words, an unmanned aerial vehicle refers to an aircraft without onboard human pilot or passenger and it is usually recovered after mission. It is controlled via a remote control mechanism or pre-programmed onboard control units.

1.2 History of UAV

This section introduces unmanned aerial vehicles from a historical perspective starting from the ancient times up to now, so as to provide an insight on the evolution of UAVs over time [60].



(a) The pigeon model **(b)** Artist's depiction of the pigeon

Figure 1.2.1: The pigeon

Though not to the common belief, the idea of building "flying machines" appeared about 2,500 years ago in ancient Greece and China. The first known autonomous flying machine was known as Archytas the Tarantine, credited to Archytas from the city Tarantas or Tarentum in South Italy. Additionally,

Archytas built a mechanical bird in 425*BC* which he called "the pigeon", as shown in Figure 1.2.1. The bird was made out of wood, according to Cornelius Gellius's Noctes Atticae [21], and it is reported that Archytas' pigeon flew around 200*m* before the energy was used up and fell to the ground.

During the same era during years c.a. 400 BC in another part the globe, the Chinese were the first to document the invention of a vertical flight aircraft. The earliest version of the aircraft, Chinese top (as illustrated in Figure 1.2.2), was recorded to consist of feathers at the end of a stick. As the stick was spun rapidly between hands so as to generate enough lift to support the flight after releasing the stick from hands.

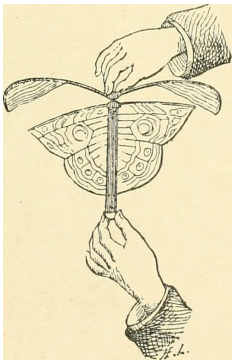


Figure 1.2.2: Chinese Top

Various flying machines has been designed since in many different countries and the need for fast and safe transportation of people and cargo have always been the main drive behind aircraft development. In the meanwhile, the military gradually realized the potential benefits of unmanned aircraft and started to make great efforts in adapting flying machines to operate without an onboard pilot. These adapted unmanned aircraft were initially unmanned ordinance delivery systems, what in modern standards would be referred to as "missiles" or "smart bombs". Another use of such systems is that they were served as "targets" in the training of antiaircraft gun operators.

The first modern unmanned aircraft, namely the Hewitt-Sperry Automatic plane, was introduced in 1916, less than 15 years after the Wright brothers

historical flight. This aircraft was named after its two inventors, which was based on the previous work of Sperry on gyroscopic devices to provide flight stabilization. U.S. Navy and Armty Air Force sponsored such and similar unmanned aircraft development projects. However, the interest on "automatic" planes was gradually lost due to technical challenges and lack of accuracy, but people came to realize the potential of remotely operated aircraft for target practice.

Around 1940, Reginald Denny'work on the radio-controlled airplanes for the U.S. Army lead to the development of very successful target UAVs, which were used extensively during WWII.

Soon after the end of WWII, the interest in reconnaissance drones rose due to the cold war between U.S. and Soviet Union. During this period, military forces of different countries invested heavily on reconnaissance unmanned aircrafts: they either developed or acquired a series of reconnaissance drones. On the US side, The first reconnaissance drone, the SD-1 (also known as MQM-57 Falconer, as shown in Figure 1.2.3) , was based on the descendants of Reginald Denny's target drones. It was remotely piloted with a camera onboard, and returned to base after a flight of duration 30*min* and recovered with parachute [67]. Later the US Navy acquired a helicopter drone called the QH-50 DASH from the Gyrodine Company, as shown in Figure 1.2.4. It is the first rotary-winged unmanned aircraft.

Meanwhile, the Soviet Air Force developed their own reconnaissance unmanned aircrafts such as TBR-1 and DBR-1. The DBR-1 (Figure 1.2.5) was designed to be only partially recovered: when it reached the recovery area, it would dump the fuel, eject the nose containing the sensor suite and leave the rest to crash. It is later replaced by the Tu-141/143 in order to reduce the high costs induced by aircraft partial recovery.

Besides these two major players in the area of unmanned aircraft, other parts of the world were also actively involved. In Europe, the French and German

Figure 1.2.3: SD-1

Figure 1.2.4: QH-50 DASH

Figure 1.2.5: DBR-1

armies acquired CL-89 Midge which was developed by Canadair. It was designed to follow a predefined flight course and return to be recovered by parachute after taking photographs [67]. The Israeli started off by purchasing a few reconnaissance drones from the US for the Yom Kippur War, and later developed their own aircrafts such as the Scout (Figure 1.2.6) and Mastiff (Figure 1.2.7).

Figure 1.2.6: Scout

Figure 1.2.7: Mastiff

Modern unmanned aerial systems are much more diverse in terms of their appearances and areas of utilization etc, and they are moving towards larger, more capable systems with higher endurance such as the MQ-9 Reper, the Neptune, A-160 Hummingbird and the Helios UAV etc.

Small (or particularly miniature) UAVs have attracted significant attention,

due to the fact that they are generally deemed as the entry points to the civilian market. Miniature UAVs boast great advantages in the civilian applications because they are low-cost, portable, agile and easy to maintain. Though their smaller sizes inevitably leads to smaller payload capacities, it does not stop them from being actively employed in many civilian applications. Examples of small and miniature UAVs are DJI Matric 600 Pro, DJI Matrice 100 (Figure 1.2.8a) and Parrot AR drone(Figure 1.2.8b) etc.

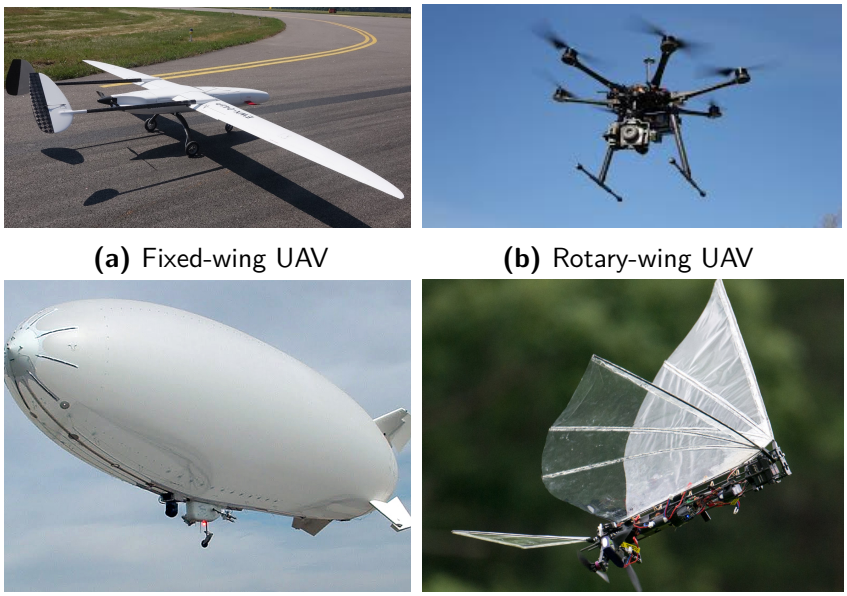
(a) DJI Matrice 100 **(b)** Parrot AR Drone

Figure 1.2.8: Examples of Miniature UAVs

1.3 UAV Classification

There are many different metrics which have been used for UAV classification, such as size, operating conditions, capabilities etc. Here we present a commonly used UAV classification which is based on the type of wings the UAV has. According to this classification metric, UAV platforms typically fall into one of the following four categories [42]:

- Fixed-wing UAVs (e.g. Figure 1.3.1a) refer to unmanned airplanes (with wings) that require a runway to take-off and land, or catapult launching. These generally have long endurance and can fly at high cruising speeds.



(a) Fixed-wing UAV **(b)** Rotary-wing UAV

(c) Blimps UAV **(d)** Flapping-wing UAV

Figure 1.3.1: Types of UAVs

- Rotary-wing UAVs(e.g. Figure 1.3.1b) are also called rotorcraft UAVs or vertical take-off and landing (VTOL) UAVs. They have the advantages of hovering capability and high maneuverability. These capabilities are useful for many robotic missions, especially in civilian applications. A rotorcraft UAV may have different configurations, with main and tail rotors (conventional helicopter), coaxial rotors, tandem rotors, multi-rotors, etc.
- Blimps (e.g. Figure 1.3.1c), such as balloons and airships. They are filled with gas that are lighter than air to ensure long endurance, but they fly at low speeds and are generally large in size.
- Flapping-wing UAVs (e.g. Figure 1.3.1d) have flexible and/or morphing small wings inspired by birds and flying insects.

In this dissertation, we mainly focus on quadrotor UAVs, which falls into the category of Rotary-wing UAVs.

1.4 Motivation

UAVs, especially quadrotor UAVs, are becoming increasingly accessible in the consumer market as they are getting more and more miniaturized and affordable. Quadrotor UAVs are mechanically simple, their sizes are generally small in comparison to other types of UAVs, and they can maneuver agilely. They are capable of flying at various speeds and in different directions, hovering and performing maneuvers in close proximity to objects in the environment [27].

In most of the civilian applications, UAVs constantly face local complex environments where both moving and static objects exist, and should be able to navigate through them in a collision-free manner [4]. In majority of the applications, UAVs are teleoperated by human operators as the maturity of fully autonomous UAVs is foreseen yet years to come. UAV teleoperation is a demanding task which requires skills and rich experience in order to ensure successful mission completion without collision. Nowadays highly-skilled UAV operators are usually sought after due to abundant UAV applications, which results in a shortage in supply. Additionally, it is usually expensive to hire those with high qualifications. Therefore, current commercial UAVs are to a large extent remotely piloted by amateur human pilots. Due to lack of teleoperation experience or skills, they often drive UAVs into collisions.

Remote operation of UAVs is challenging mainly due to the loss of perception information when a human pilot is not physically onboard on the UAV. In many cases, UAV designers try to compensate this loss of information by equipping UAV with an onboard camera to provide the remote pilot with a forward-looking view of the environment. Though such cameras may provide additional visual information when the UAV is out of sight, the view provided by the onboard camera has been described as "looking at the world through a soda straw" [65]. The decreased situation awareness usually results in more challenges on the UAV pilot and often lead to collisions with objects in the environment.

The ability of quadrotor UAVs to fly in any directions is a considerable advantage in many cases, however, it also poses more challenges on the remote operator as

omni-directional motion of the quadrotor may lead it to move in directions that are not within the onboard perception sensors' field of view, which makes the UAV susceptible to collisions especially when the UAV is out of the direct sight of the operator.

In order to address the pressing challenge of alleviating the burden on unskilled UAV operators during the task of collision avoidance, it is desirable to develop a system that automatically assists amateur operators to accomplish missions successfully in aforementioned tasks, where the UAV has the capability to safely maneuver in potentially unknown outdoor environment, due to operator's limited situation awareness.

Always keep an open mind and to listen to what those around you have to say about you. You might actually learn something that may help you become a better person!

Auliq Ice

2

Related Work

2.1 Obstacle Avoidance (Sense-and-Avoid)

Obstacle avoidance is essential to ensure the safety of UAVs as well as its surrounding environment during its navigation. In a shared airspace, an UAV may come across many objects (or obstacles) which would endanger its mission.

For pure human-remotely-piloted UAVs, the task of obstacle avoidance is performed by human operators. However, unskilled operators do not produce satisfying performance of obstacle avoidance, and with the hope of achieving full UAV autonomy, many researchers have resorted to collision avoidance systems.

A collision avoidance system, or Sense-and-Avoid system, usually starts with sensing the environment around the UAV with onboard sensors and ends with

Figure 2.1.1: Functional Diagram of Collision Avoidance System

taking actions to avoid a determined threat to the UAV. It is in general divided into three major functional components (though different works name them slightly differently) [35] [31], which are as follows (as shown in Figure 2.1.1):

- Sensing: represents the capability of a system to perceive its surrounding environment and collect information for situation awareness.
- Emergency evaluation: enables UAVs to evaluate potential collision risks based on sensing data, and decide if imminent collision threat is present, and if avoidance actions should be invoked.
- Collision Avoidance: determines what maneuvers should be performed in order to avoid a potential near-future collision.

Despite the general system structure, these three fundamental components of a Sense-and-Avoid system can be implemented in different ways, which results in variations in different solutions.

2.1.1 Sensing

Depending on the information sources, sensing systems can be classified into two general categories, namely cooperative sensing and non-cooperative sensing. Cooperative sensing systems utilize transponders to exchange information with other vehicles which are equipped with similar sensing systems, while non-cooperative sensing systems are capable of detecting the presence of objects without active interactions with other vehicles [18].

Cooperative Sensing

Cooperative sensing technology works based on information (such as position, heading, speed, waypoints etc.) exchange between aircrafts in the vicinity which are equipped with similar cooperative sensors.

The primarily used cooperative sensing device is Air Traffic Control (ATC) transponder, which transmits a specific reply upon receiving a certain radio frequency interrogation. This sensing approach performs well in a controlled airspace where all aircrafts are equipped with the transponders. However, when there are aircrafts without the transponders in the airspace, they would not be perceived by others utilizing transponders and thus require other means of sensing. Transponders are usually sold for powered airplanes, since they consume a large amount of power, thus in general they are not suitable for battery powered UAVs.

The ADS-B (Automatic Dependent Surveillance Broadcast) is an alternative cooperative sensing technology. The ADS-B sensing systems provides more information with higher accuracy than transponders among similarly equipped aircrafts. They depend on Global Navigation Satellite System (GNSS) for precise 3D positions, which along with other information such as unique identifier, speed, heading intent etc. are periodically updated and broadcasted. This allows for accurate monitoring of the UAVs in the vicinity. The ADS-B sensing system depends heavily on the integrity of the UAV navigation system in order to broadcast its accurate position. Another drawback of the ADS-B is that sensed fraudulent messages can easily lead to simulated non-existent traffic. [35]

Non-cooperative Sensing

Non-cooperative sensing system does not require other objects in the airspace to be similarly equipped, instead, the sensing units allow UAVs to perceive objects in the environment without external support. These sensors usually perform a scan of the region of interest in order to utilize the detected objects information for

collision threat assessment. This is probably the most intuitive form as it is the form that most resembles a human's sensing system.

Depending on the circumstances of the applications, the non-cooperative sensors employed can be active or passive sensors. During sensing process, active sensors transmit energy that is needed for perceiving the environment, while passive sensors rely on resources such as light, heat emission etc. Commonly used active sensors are, for example, radar, LIDAR (Light Detection and Ranging), ToF (Time-of-Flight) cameras, active sonar etc, while electro-optical (EO) cameras and passive sonar are the main examples in the passive sensor category.

Non-cooperative sensors can be installed either on the aircraft or on the ground, which result in very different sensing perspectives. Sensors fitted on the UAV provide an ego-centric view, while those based on the ground offer an environment-based view. In majority cases, the sensors are installed onboard of the UAV. Generally, active sensors are more accurate and more reliable than passive sensors, but are much more demanding in terms of onboard resources such as electrical power, space, and weight.

2.1.2 Emergency Evaluation

The emergency evaluation system constantly evaluates the situation around the UAV based on objects information from the sensing system, and decides if there is imminent threat. Once the presence of imminent threat is determined, the UAV should take actions to avoid potential collisions. There are a variety of conflict metrics which are utilized to determine the level of threat (or emergency), for example from as simple as using minimum separation distance to estimated Time-To-Collision and calculating conflict probability.

Conflict evaluation is usually associated with one or more threshold values, i.e., a decision regarding whether it is necessary to invoke avoidance action will be made by comparing conflict metrics with the threshold value(s). The choosing of the threshold values depend on many factors such as the surrounding

environment and states of the UAV etc. Ideally, the threshold values should be dynamically adapted based on various situations.

2.1.3 Collision Avoidance

The collision avoidance system should be triggered once it is determined that there is imminent threat (i.e. emergency situation) to the UAV.

Collision avoidance techniques can be mainly categorized based on four design aspects, namely, the avoidance manoeuvre, the avoidance action generation algorithm, the handling approach of multiple conflicts and the coordination assumption between objects. [35]

Avoidance manoeuvres

The avoidance manoeuvres are a set of actions which can be used to avoid a potential collision. Fundamental manoeuvres may include speed changes (e.g. speedup or slowdown), horizontal manoeuvres (e.g. turn left or right) and vertical manoeuvres (climb or descend). It is quite often to combine some of these fundamental manoeuvres during the process of avoidance, either simultaneously or sequentially.

Avoidance action generation algorithms

Avoidance action generation algorithms usually comprise following types of approaches: rule based, virtual force field field based, optimization based and bearing angle based.

In the rule based approaches, potential collisions are avoided by employing a set of predefined rules. For example, in work [12] the UAV in threat would perform a fixed climbing turn in order to avoid potential collision with a parallel flying obstacle. Rule based approaches are generally easy to understand and

implement, and fast to respond. However, these naive approaches are not generally applicable for many different circumstances and especially when the UAV in threat and the obstacle(s) are not in coordination, the effect of the avoidance action can easily be diminished by the movement of the obstacle(s).

The virtual force field approach takes the intuition from the repulsive and attractive electrical force field, using the concept of "potential" as a measurement of desirable characteristics of the UAV trajectory which is collision-free. The most commonly used virtual force field approach is the potential field, since calculating the potential is a huge work, the potential gradient is usually calculated instead at each sampling point. With a naive virtual force field approach the feasibility of the computed avoidance actions are not guaranteed due to UAVs' dynamic limitations.

Bearing angle based approaches usually utilize visual sensor's ability to accurately return the relative angle of the obstacles toward the UAV. The main idea is that by keeping obstacles' images at the "safe" position in the sensor field of view, the UAV can effectively prevent collision. This type of approach features spiral flight path. The bearing angle approach is affected by all the disadvantages of the visual sensor, which are relying on image features processing techniques and significantly affected by all the disadvantages of the visual sensor, which are relying on image features processing techniques and significantly affected by external conditions [47]. For instance, [57] introduces a reactive guidance strategy for collision avoidance, which uses bearing-only measurements provided by a camera. The guidance law is derived from sliding mode control theory, and it moves an obstacle in the sensor field-of-view to a desired constant bearing angle, which leads the UAV to maintain a constant distance from the obstacle. The proposed guidance law can be used to avoid collision into circular obstacles and to follow straight and curved walls at a safe distance.

Optimization based approaches generate avoidance manoeuvres which minimize a certain cost function, such as flight time, deviation from the original trajectory etc. Due to the general high computation of this approach, it has been mostly utilized for dealing with static objects, and the perception sensors are

generally assumed working sensors, which makes this approach more theoretical than other approaches. However, there have been also many attempts in the research community to employ these approaches for dynamic obstacle avoidance scenarios.

Optimization based approaches can mainly be categorized into game theory based methods, genetic based methods and optimal control theory methods. Game theory based methods. Game theory based approaches formulates the avoidance problem as a cooperative or a non-cooperative game. In genetic based approaches avoidance actions are generated using techniques inspired by natural evolution such as crossing, mutation and selection. Optimal control theory methods require the UAV dynamics, constraints and cost function to be defined in order to solve for avoidance manoeuvres. Optimization based approaches may appear attractive since they often produce an ”optimal” solution, however, the complexity of the used functions can make these approaches hard to understand and computationally expensive. [35]

2.1.4 Handling approaches of multiple conflicts

When handling conflicts with multiple obstacles at the same time, there are mainly two approaches that are utilized for generating avoidance actions: pairwise and global. In the pairwise approach the UAV addresses the conflicts with obstacles sequentially in pairs, while in the global method the UAV addresses the entire situation as a whole, which is commonly accomplished by grouping aircrafts in clusters. Pairwise approaches may appear simpler than global method, but it may lead to unsafe situations. Global methods usually require more computation but are often more robust. [35]

Different sense-and-avoid solutions have been proposed in the literature with varying sensing modalities and avoidance techniques, mostly for autonomous UAVs. [36] presents a collision avoidance algorithm which only utilizes measuremnets from a gimbaled monocular camera. Line-of-Sight (LOS) angle and time-to-collision (TTC) are abstracted through computer vision algorithms,

which are subsequently used to form an avoidance control law controlling the angular rate of the UAV. Simulation is conducted in a single-moving-object scenario and a pure-static-objects scenario to validate the proposed avoidance control law.

In [24] a goal-directed 3D reactive obstacle avoidance algorithm is proposed for rotorcraft UAVs equipped with a stereo camera pair and a 2D LIDAR sensor. 3D occupancy map is used as world representation, where the avoidance algorithm projects a cylindrical safety volume ahead of the UAV to detect potential collisions. Once potential collision is detected, an expanding elliptical search is carried out to find an escape point. Both simulation and real flight experiments have been conducted to validate the algorithm in static environments with trees and communication towers. [57] introduces a reactive guidance strategy for collision avoidance, which uses bearing-only measurements provided by a camera. The guidance law is derived from sliding mode control theory, and it moves an obstacle in the sensor field-of-view to a desired constant bearing angle, which leads the UAV to maintain a constant distance from the obstacle. The proposed guidance law can be used to avoid collision into circular obstacles and to follow straight and curved walls at a safe distance.

Most of the existing works on sense-and-avoid systems are vision-based. Additionally, the proposed algorithms in most works are validated in simulation or experiments with either a single-moving obstacle or a pure static environment. Their capabilities of avoidance in complex environments where there exist multiple moving objects, are barely investigated.

2.2 Object Detection and Tracking

Object detection and tracking on mobile platforms in dynamic environments have been extensively studied over the last two decades, with focus on ground mobile robot platforms, while applications dedicated to UAV platforms are less explored. Depending on the sensor modalities employed for perception, the related works can mainly be divided into three categories: camera-based approaches, LIDAR-

based approaches and sensor fusion-based approaches.

Many research works focus on vision-based approach, and different camera types such as monocular camera [3, 66], stereo camera [17, 59], RGB-D camera [44] and thermal imaging camera [34] have been employed in the literature for the purpose of obstacle detection and tracking. In general, the performance of vision-based approaches is strongly subject to illumination conditions especially when in outdoors, and demands high computation power which makes it difficult for the task of detection and tracking to be completely online.

In comparison to vision-bases approaches, LIDAR-based systems are insensitive to lighting conditions and requires significantly less processing time especially when a 2D LIDAR is used. In addition, LIDARs generally have wider field of view (FOV) horizontally than cameras, and thus can be aware of threats not only in the front but also from the sides.

Most of the LIDAR-based works use 2D or 3D occupancy grid to represent the robot's environment such as [5, 7, 19, 20, 62, 63]. In these works, the occupancy grid is segmented into object list and then apply different shapes of trackers to track obstacles in the environment. Another commonly used representation of the environment is line segments such as in [16, 39, 53]. Others incorporates higher-level geometry primitives such as circles and rectangles to directly model and track the obstacles in the environment such as those in [50] and [37], this method is especially good for a sparse environment because of its compact representation of the environment and low memory consumption in comparison to other two methods.

In the literature, sensor fusion approaches which usually combine LIDAR with other types of sensors such as camera or RADAR are also employed to address object detection and tracking problem, such as those in [13, 15, 32] and [41]. In comparison to single-sensor approach, sensor fusion systems are usually more complex and demands more computation power.

2.3 Quadrotor Teleoperation Assistance System

Some research have been carried out in terms of UAV teleoperation assistance systems. There are mainly four types of UAV teleoperation assistance systems in terms of obstacle avoidance: visual feedback, haptic force feedback approach, automatic collision avoidance system, and combined methods.

2.3.1 Visual and Haptic Force Feedback

Many works are related to visual and haptic UAV teleoperation assistance systems in support of operators' operation. [33] describes the design and theoretical evaluation of a novel artificial force field (AFF), i.e., the parametric risk field, which generates haptic force feedback for assisting teleoperation of an UAV. The size, shape, and force gradient of the field can be adjusted through parameter settings, and they determine the sensitivity of the field. Simulations were conducted to evaluate the effectiveness of the field for collision avoidance for various parameter settings. Results indicate that the novel AFF more effectively performs the collision avoidance function than potential fields known from literature. [11] explores three different force feedback algorithms for haptic control of quadrotor UAVs in indoor environment, which are dynamic parametric field (DPF), time-to-impact (TTI) and virtual spring (VS) respectively. The TTI algorithm is based on the quadrotor velocity and distance to the obstacle, and in the DPF algorithm the feedback force is dependent on which zone (safe, warning, transition and collision) the UAV is located in. In VS algorithm, the force is modeled as a virtual spring, which varies linearly with the distance to the obstacle. The work aims to utilize haptic feedback generated by these proposed algorithms to supplement the limited visual visual feedback, in order to assist the pilot in avoiding obstacles. The force is not designed to overpower the operator, but to serve as a warning. Software was developed to simulate 3D motion and 3D force feedback in an indoor environment, and user studies were conducted using this simulation to compare the effectiveness of the three algorithms. Finally, the complete system was demonstrated on an actual

quadrotor in an indoor environment. The user study suggests that the time-to-impact algorithm is significantly better than the other algorithms and the no-force case. It is claimed in the work that the TTI algorithm significantly reduced the number of collisions compared to all other algorithms without increasing the time taken to complete the task or the workload on the pilot. The initial study also indicated that higher forces are typically more effective in decreasing the number of collisions compared to lower forces for the algorithms.

It is presented in [56] that a simulated haptic-aided UAV flight control system, which is utilized to invistigate the effects of haptic feedback on longitudinal flight control. Experimental evaluations of three flight control scenarios were conducted, namely without haptic feedback, with realistic haptic feedback and with exaggerated haptic feedback. The analysis shows that the proposed haptic aiding system has a significant effect on operator performance and situational awareness. On the basis of six different performance parameters, it is concluded that no force feedback is acceptable for altitude hold tasks, exaggerated force feedback works better for altitude gain, and realistic force feedback is adequate for altitude descent. The subjective assessment reported by the participants suggested an increased confidence with haptic feedback.

[54] presents a novel configuration of assisted flying and implements an experimental setup in order to prove its efficacy. The user controls an unmanned aerial vehicle with a force feedback device, where simultaneously an assisted navigation algorithm can manipulate this apparatus to divert the unmanned aerial vehicle from its path. Experiments confirm the authors' hypothesis that the unmanned aerial vehicle is deviated or maintains the same course at the operator's will. Unlike conventional controllers that dictate roll, pitch, and yaw, this implementation uses direct mapping between the position represented by the haptic device and the unmanned aerial vehicle. This configuration applies feedback before the unmanned aerial vehicle has reached the position referenced by the haptic device, providing valuable time for the user to make the necessary path correction.

[23] proposes to add in on top of the haptic feedback system its visualization

into the UAV teleoperation system in order to increase the pilot's task performance especially in terms of collision avoidance, as well as pilot's willingness to accept the haptic feedback system. It is reported that at some moments the haptic feedback could appear "too strong", and "unpredictable", as the pilots could not decipher what reasoning was underlying the feedback forces. Results of a human-in-the-loop experiment show that these visualizations indeed led to higher user accepting ratings, without affecting the performance increase brought in by force feedback aid.

[48] presents an improvement of previous haptic feedback system in order to better assist UAV operator in avoiding obstacles, especially dynamic obstacles. The new haptic assist algorithm is based on velocity obstacle theory, and firstly tested offline and then on real-time environment and evaluated in pilot-in-the-loop simulations. Results indicate that the velocity obstacle approach works for both stationary and moving obstacles, in most, but not all scenarios.

2.3.2 Automatic Obstacle Avoidance

Different UAV teleoperation assistance systems for automatic collision avoidance have been proposed in the literature with varying sensing modalities, environment representations and avoidance techniques.

[64] proposes a 2D LIDAR based optimization based Sense-and-Avoid solution which assists amateur human pilots in safe piloting in an environment where multiple moving objects exist. The proposed system, however, mainly targets at beginner-level operator who have near-zero UAV teleoperation skills or experience. Therefore, the operator's intention is not taken into account in this work. Additionally, in this work the sensing and avoidance subsystems are not yet integrated.

Another 2D LIDAR based approach is introduced in [6], which proposes a model-based stochastic automatic collision avoidance algorithm for assisting UAV teleoperation. The proposed algorithm takes pilot's input and robot's

dynamics to predict its trajectory so as to determine if a collision will occur. When a collision is imminent, the algorithm modifies the pilot's input to avoid the collision while attempting to maintain the pilot's intent. The environment is represented using line segments. Experiments are conducted in static indoor environment with walls.

An RGB-D camera based collision avoidance system is proposed in [43] to assist safe indoor navigation of teleoperated multirotor UAVs. The algorithm keeps track of detected obstacles in the local surroudings of the robot. The environment is represented with a set of cells in the near surrounding of the UAV. This representation is very similar to 3D occupancy grid. MPC is employed for generating avoidance control command, which modifies operator's input in order to avoid collision, while at the same time preserving the intent of the operator. Experiments are conducted in a static environment with a single object. [45] is an improvement of [43] with added functionalities.

Both [22] and [38] propose sonar-based teleoperation assistance systems for obstacle avoidance. [22] adopts MPC control technique to predict potential future collision, and uses SLAM and Octomap to construct a map of the environment. Simulation was conducted in a static environment with the UAV flies at a low speed. [38] uses FastSLAM to construct a rough occupancy grid map for the near surroundings. The operator's input is overridden proportionally to an approximate TTC (time to collision), based on the vehicle's current velocity and distance to the obstacle. Depending on the level of threat, different response is given, such as no action, slow, stop, evasive maneuver. Simulation is carried out in a static indoor environment with walls.

[25] presents an automatic collision avoidance approach which assumes full environment awareness. The proposed method is essentially an MPC approach. The mechanisms of sensing, threat evaluation, and interaction between human operator and UAV are not discussed and assumed to be working. Experiment with real quadrotor is performed in a static lab environment, with the assumption of full environment awareness.

Among the existing relevant works on UAV teleoperation assistance systems,

model predictive control is found to be a popular avoidance technique. However, optimization-based algorithms are in general relatively slow in comparison to reactive avoidance approaches. Additionally, different perception sensor choices and world representations affect UAV's capability of perceiving and interpreting its surrounding environment. In a high-speed flight, it is preferrable to choose a perception sensor with longer range of detection and better precision and fast update rate.

The greatest glory in living lies not in never falling, but in rising every time we fall.

Nelson Mandela

3

Problem Overview

3.1 Problem Statement

We would like to develop a Sense-and-Avoid teleoperation assistance system which automatically detects and avoids collisions, so that the operator can focus on high-level decisions and global objectives. The problems that we need to address in order to achieve the goal are as follows:

- How to perceive the environment? What perception sensor(s) to use? What world representation to use? How to mount the perception sensor on the robot to perceive surroundings effectively?
- How to evaluate the situation? what is the criteria used to identify an emergency situation? How is it related to the perception and avoidance

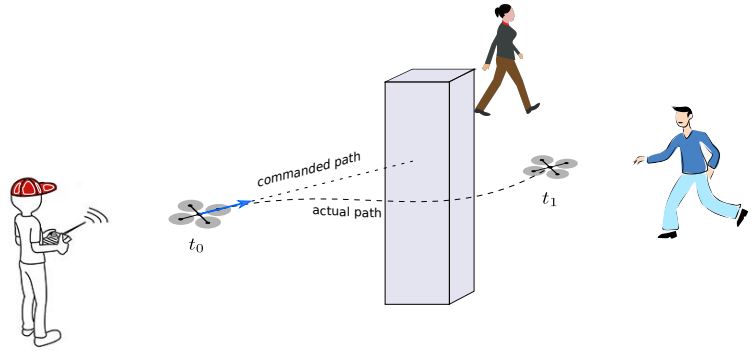


Figure 3.1.1: Real World Scenario

action?

- How to avoid collision in case of emergency? What kind of maneuvers to take? What circumstance is deemed as out of the emergency situation?
- How to control the quadrotor UAV? How to take into account operator's command and automatic avoidance control command? Can UAV controller be designed to be beneficial for both perception and collision avoidance?

3.2 System Overview

The section we introduce the overview of our proposed UAV operator assistance obstacle avoidance system.

We propose two collision avoidance solutions, one of which is optimization-based, and the other is AVCFF-based. In optimization-based approach the operator's command is completely overridden by the avoidance control command during the automatic collision avoidance process, while in the AVCFF-based approach the operator's intention right before entering the collision avoidance phase is taken into consideration for generating the avoidance controls.

In comparison to optimization-based avoidance approach, AVCFF-based

approach is more computationally efficient and thus faster in response, which suits better for a high-speed UAV setting.

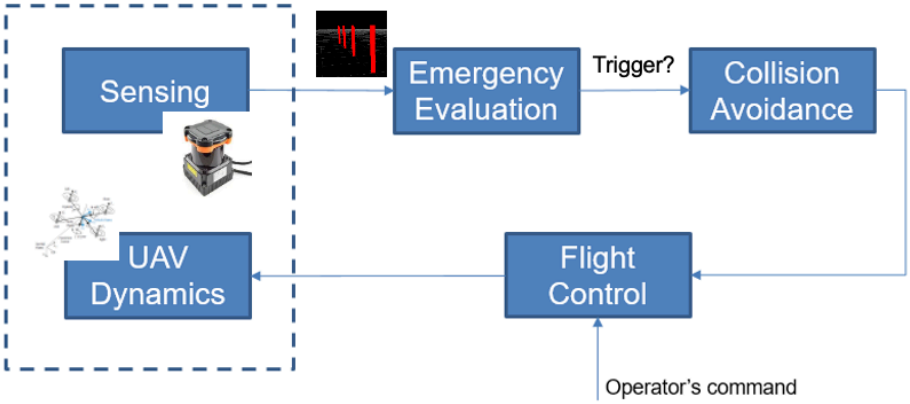


Figure 3.2.1: System Overview

3.3 Publications

The majority of the contents in this dissertation haven been published in the following conference proceedings:

- M. Wang, H. Voos and D. Su, "Robust Online Obstacle Detection and Tracking for Collision-Free Navigation of Multirotor UAVs in Complex Environments," 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2018, pp. 1228-1234, doi: 10.1109/ICARCV.2018.8581330.
- M. Wang and H. Voos, "Safer UAV Piloting: A Robust Sense-and-Avoid Solution for Remotely Piloted Quadrotor UAVs in Complex Environments," 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2019, pp. 529-534, doi: 10.1109/ICAR46387.2019.8981576.
- M. Wang and H. Voos, "An Integrated Teleoperation Assistance System for Collision Avoidance of High-speed UAVs in Complex Environments," 2020

17th International Conference on Ubiquitous Robots (UR), Kyoto, Japan, 2020, pp. 290-296, doi: 10.1109/UR49135.2020.9144902.

Always believe in yourself and always stretch yourself beyond your limits. Your life is worth a lot more than you think because you are capable of accomplishing more than you know. You have more potential than you think, but you will never know your full potential unless you keep challenging yourself and pushing beyond your own self imposed limits.

Roy T. Bennett

4

Conventions and Fundamentals

This chapter introduces the nomenclature, conventions as well as fundamentals used in this thesis.

4.1 Nomenclature

Data Types	**Examples**
Scalar Variable	x, y
Scalar Constant	M, N
Vector Variable	$\boldsymbol{x}, \boldsymbol{y}$
Matrix Variable	$\boldsymbol{A}, \boldsymbol{B}$
Function	$d(\bullet), f(\bullet)$
Probability	$P(x_t \vert u_t, x_{t-1})$

4.2 Frame Orientation Convention

There are different ways of describing the orientation of a frame, for example, using conventions of Z-Y-Z fixed angles, Z-Y-X Euler angles or Z-Y-Z Euler angles etc [14]. In this work, we use the convention of Z-Y-X Euler angles to represent the orientation of a frame (as shown in Figure 4.2.1 using intermediate frames): Start with the frame coincident with a known frame A. Rotate first about axis z^0 by an angle ψ to obtain frame A^1. Then rotate frame A^1 about axis y^1 by an angle θ in order to obtain frame A^2. Finally frame B is achieved by rotating frame A^2 about axis x^2 by an angle ϕ.

In this representation, each rotation is performed about an axis of the moving system rather than one of the fixed reference. Such sets of rotations are called Euler angles. Note that each rotation takes place about an axis whose location depends upon the preceding rotations. Because the three rotations occur about the axes z, y, and x, we call this representation Z-Y-X Euler angles.

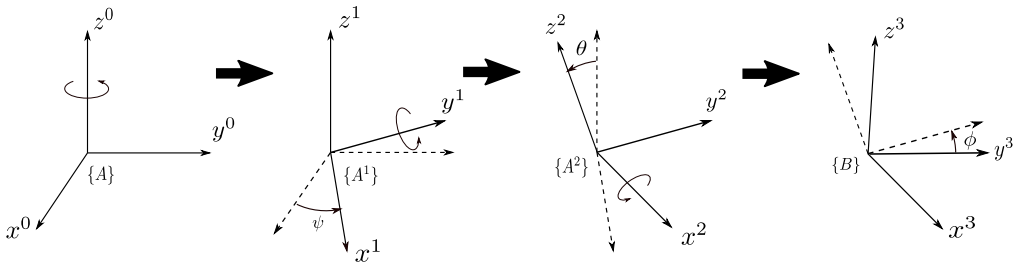


Figure 4.2.1: Z-Y-X Euler angles convention

$$ {}^{A}_{B}\boldsymbol{R} = {}^{A}_{A^1}\boldsymbol{R}\,{}^{A^1}_{A^2}R\,{}^{A^2}_{B}\boldsymbol{R} \tag{4.1} $$

The final orientation of frame B is given relative to frame A as

$$\begin{aligned} {}^{A}_{B}\boldsymbol{R} &= \boldsymbol{R}_{z^0}(\psi)\boldsymbol{R}_{y^1}(\theta)\boldsymbol{R}_{x^2}(\phi) \\ &= \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \end{aligned} \tag{4.2}$$

where $c\psi = cos\psi$, $s\psi = sin\psi$ and so on. Multiplying out the matrices, we obtain

$${}^{A}_{B}\boldsymbol{R}(\psi,\theta,\phi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\theta & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{4.3}$$

4.2.1 Homogeneous Transformation

Very often, we know the description of a vector (or a point) with respect to some frame e.g. $\{B\}$, and we would like to know its description with regard to another frame, e.g. $\{A\}$. In the general case of mapping, the origin of frame $\{B\}$ is not coincident with that of frame $\{A\}$. The vector that locates frame $\{B\}$'s origin in frame $\{A\}$ is denoted as ${}^{A}\boldsymbol{p}_{BORG}$. The rotation of frame $\{B\}$ with respect to frame $\{A\}$ is described by ${}^{A}_{B}\boldsymbol{R}$. Thus the task is to compute ${}^{A}\boldsymbol{p}$ given ${}^{B}\boldsymbol{p}$.

We can firstly change ${}^{B}p$ to its description relative to an intermediate frame that has the same orientation as $\{A\}$, but whose origin is coincident with the origin of $\{B\}$. We then account for the translation between origins. In mathematical form, we obtain

$${}^{A}\boldsymbol{p} = {}^{A}_{B}\boldsymbol{R}{}^{B}\boldsymbol{p} + {}^{A}\boldsymbol{p}_{BORG} \tag{4.4}$$

In order to facilitate easy understanding of the transformation mapping from one frame to another frame, we adopt the form of homogeneous transform matrix,

which is as follows:

$$\begin{bmatrix} {}^{A}\boldsymbol{p} \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & {}^{A}_{B}\boldsymbol{R} & & {}^{A}\boldsymbol{p}_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}\boldsymbol{p} \\ 1 \end{bmatrix} \tag{4.5}$$

The $4x4$ matrix in the equation above is the homogeneous transform, and we often write above equation in a more compact form:

$${}^{A}\boldsymbol{p} = {}^{A}_{B}\boldsymbol{T}{}^{B}\boldsymbol{p} \tag{4.6}$$

4.3 UAV Dynamical Model

Mathematical models describes the behavior of a system. The flight behavior of a quadrotor is determined by the speeds of its four motor/rotor. Given input, a mathematical model of a quadrotor can be utilized to predict its position and orientation. Additionally, mathematical models can also be used for developing control strategies. [52]

This section firstly introduces a commonly used coordinate frame system for UAVs and then presents the details of a quadrotor's dynamical model.

4.3.1 Reference frames

Before getting into the dynamics modelling of a quadrotor, it is necessary to specify the adopted coordinate systems and frames of reference, as well as how transformations between the different coordinate systems are carried out.

The use of different coordinate frames is essential for identifying the location and attitude of the quadrotor in six degrees of freedom (6 DOF). For example, in order to evaluate the equations of motion, a coordinate frame attached to the quadrotor is required. However, the forces and moments acting on the quadrotor, along with the inertial measurement unit (IMU) sensor values, are evaluated with

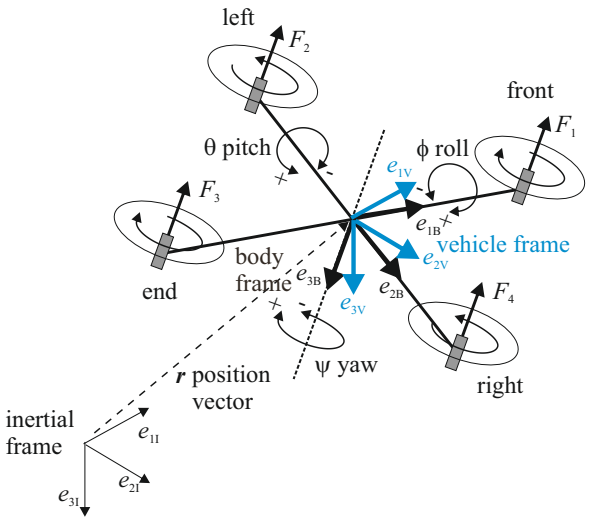


Figure 4.3.1: Reference Frame System of Quadrotor

reference to the body frame. Finally, the position and speed of the quadrotor are evaluated using GPS measurements with respect to inertial frame located at the base station.

Thus, three main frames of reference are adopted, as shown in Fig. 4.3.1:

1. The inertial frame (denoted using left superscript with letter i): also called the world frame, it is an earth-fixed coordinate system with the origin located on the ground. We use the standard *NED* (North, East, Down) convention: the x-axis points towards the north, the y-axis points towards the east, and the z-axis points towards the center of the earth.

2. The body frame (denoted using left superscript with letter b): it has its origin located at the center of gravity (COG) of the quadrotor, and its axes aligned with the quadrotor structure such that the x-axis is along the arm (lever) with the front motor, the y-axis is along the arm with right motor.

3. The vehicle frame (denoted using left superscript with letter v): it is the inertial frame with the origin located at the COG of the quadrotor, and it is the reference frame for Euler angle ψ. There are two intermediate vehicle

frames, which are used for obtaining Euler angles θ and ϕ respectively, for details please refer to section "Frame Orientation Convention" in Chapter 2. However, for ease of explanation and notation, we simply denote these angles as ${}^{v}\psi, {}^{v}\theta, {}^{v}\phi$.

Translation vector and rotation matrices are used to transform one coordinate reference frame into another desired frame of reference. For example, the transformation from ? to ? provides the displacement vector from the origin of the inertial frame to the COG of the quadrotor. Also, the transformation from vehicle frame to body frame is rotational in nature, therefore yielding the yaw, pitch and roll angles, which are ${}^{v}\psi, {}^{v}\theta, {}^{v}\phi$ respectively.

4.3.2 Quadrotor's Dynamics

The transitional dynamics of the vehicle in inertial (i.e. world) coordinate is given by Newton's second law

$$ {}^{i}\ddot{\boldsymbol{p}}M = \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix} - {}^{i}_{b}\boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ B\sum_{j=1}^{4} w_j^2 \end{bmatrix} \tag{4.7} $$

where

${}^{i}\boldsymbol{p}$ is the position of the vehicle in the inertial frame, ${}^{i}\boldsymbol{p} = (x, y, z)^T$;

G is gravitational acceleration;

M is the total mass of the vehicle;

B is a positive constant which represent the thrust factor which depends on the air density, the cube of the rotor blade radius, the number of blades, and the cord length of the blade;

w_j is the jth rotor's speed, $j = 1, 2, 3, 4$;

${}^{i}_{b}\boldsymbol{R}$ is the rotation matrix which transforms coordinates from body frame to vehicle frame using the z-y-x Euler angles convention. It has its full mathematical

form as follows

$$ {}^{i}_{b}\boldsymbol{R}_{zyx}({}^{v}\psi, {}^{v}\theta, {}^{v}\phi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\theta & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{4.8} $$

For simplicity and easy of representation, in the following notations, we will simply use ψ, θ and ϕ without reference frame superscripts to represent ${}^{v}\psi$, ${}^{v}\theta$ and ${}^{v}\phi$ respectively, while angles represented in body frame remain with reference frame superscripts, as ${}^{b}\psi$, ${}^{b}\theta$ and ${}^{b}\phi$.

The first term in Eq. (4.7) is the force of gravity which acts downward in the inertial frame and the second term is the total thrust in the vehicle frame rotated in the world coordinate frame.

$$ \boldsymbol{J}^{b}\dot{\boldsymbol{\omega}} = -{}^{b}\boldsymbol{\omega} \times \boldsymbol{J}^{b}\boldsymbol{\omega} + {}^{b}\boldsymbol{\tau} - {}^{b}\boldsymbol{\tau}_{G} \tag{4.9} $$

where

$\boldsymbol{J}$ is the 3×3 inertia matrix of the vehicle. It is a symmetrical matrix with only diagonal elements J_x, J_y, J_z when the object's mass distribution is symmetrical with respect to the 6coordinate frame;

${}^{b}\boldsymbol{\omega}$ is the angular velocity expressed in the body frame, ${}^{b}\boldsymbol{\omega} = ({}^{b}\dot{\phi}, {}^{b}\dot{\theta}, {}^{b}\dot{\psi})^{T}$;

${}^{b}\boldsymbol{\tau}$ is the torque applied to the vehicle in the body frame, ${}^{b}\boldsymbol{\tau} = [{}^{b}\tau_x, {}^{b}\tau_y, {}^{b}\tau_z]^{T}$;

${}^{b}\boldsymbol{\tau}_{G}$ is the gyroscopic torque in the body frame, which is usually ignored in most of the dynamical model of quadrotor.

The torque vector $\boldsymbol{\tau}^{b}$ is defined as

$$ {}^{b}\boldsymbol{\tau} = \begin{bmatrix} LB(w_2^2 - w_4^2) \\ LB(w_1^2 - w_3^2) \\ D(w_1^2 + w_3^2 - w_2^2 - w_4^2) \end{bmatrix} \tag{4.10} $$

where

L is the distance from the motor to the center of mass;

D is the drag factor which depends on the same factors as B.

The gyroscopic torque is described as

$$ {}^{b}\boldsymbol{\tau}_G = J_R({}^{b}\boldsymbol{\omega} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix})(w_1 - w_2 + w_3 - w_4) \tag{4.11} $$

where

J_R is rotor inertia;

If we expand Eq. (4.7), then we get

$$ M \begin{bmatrix} {}^{i}\ddot{x} \\ {}^{i}\ddot{y} \\ {}^{i}\ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix} - \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\theta & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ B\sum_{j=1}^{4} w_j^2 \end{bmatrix} \tag{4.12} $$

Solving Eq. (4.12) we can obtain

$$ \begin{bmatrix} {}^{i}\ddot{x} \\ {}^{i}\ddot{y} \\ {}^{i}\ddot{z} \end{bmatrix} = \begin{bmatrix} -(c\psi s\theta c\phi + s\psi s\phi)\frac{B}{M}\sum_{j=1}^{4} w_j^2 \\ -(s\psi s\theta c\phi - c\psi s\phi)\frac{B}{M}\sum_{j=1}^{4} w_j^2 \\ \frac{G}{M} - (c\theta c\phi)\frac{B}{M}\sum_{j=1}^{4} w_j^2 \end{bmatrix} \tag{4.13} $$

If we expand Eq. (4.9), we get

$$ \begin{aligned} \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \begin{bmatrix} {}^{b}\ddot{\phi} \\ {}^{b}\ddot{\theta} \\ {}^{b}\ddot{\psi} \end{bmatrix} = &- \begin{bmatrix} {}^{b}\dot{\phi} \\ {}^{b}\dot{\theta} \\ {}^{b}\dot{\psi} \end{bmatrix} \times \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \begin{bmatrix} {}^{b}\dot{\phi} \\ {}^{b}\dot{\theta} \\ {}^{b}\dot{\psi} \end{bmatrix} + \begin{bmatrix} LB(w_2^2 - w_4^2) \\ LB(w_1^2 - w_3^2) \\ D(w_1^2 + w_3^2 - w_2^2 - w_4^2) \end{bmatrix} \\ &- J_R({}^{b}\boldsymbol{\omega} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix})(w_1 - w_2 + w_3 - w_4) \end{aligned} \tag{4.14} $$

Solving Eq. (4.14), we can obtain

$$\begin{bmatrix} {}^b\ddot{\phi} \\ {}^b\ddot{\theta} \\ {}^b\ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x} {}^b\dot{\theta}\,{}^b\dot{\psi} + \frac{LB}{J_x}(w_2^2 - w_4^2) - \frac{J_R}{J_x} {}^b\dot{\theta}(w_1 - w_2 + w_3 - w_4) \\ \frac{J_z - J_x}{J_y} {}^b\dot{\phi}\,{}^b\dot{\psi} + \frac{LB}{J_y}(w_1^2 - w_3^2) + \frac{J_R}{J_x} {}^b\dot{\phi}(w_1 - w_2 + w_3 - w_4) \\ \frac{J_x - J_y}{J_z} {}^b\dot{\phi}\,{}^b\dot{\theta} + \frac{D}{J_z}(w_1^2 + w_3^2 - w_2^2 - w_4^2) \end{bmatrix} \tag{4.15}$$

The angular rates in the body and vehicle frames are related in the following form

$$\begin{bmatrix} {}^b\dot{\phi} \\ {}^b\dot{\theta} \\ {}^b\dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4.16}$$

If we assume that the quadrotor is not too far from the hovering state, the angles ϕ and θ are therefore small, then Eq. (4.16) can be reasonably approximated by

$$\begin{bmatrix} {}^b\dot{\phi} \\ {}^b\dot{\theta} \\ {}^b\dot{\psi} \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4.17}$$

More then often, we would like to represent the UAV dynamical model in the state space format. There are usually different ways of defining the state space vectors, in our example, we define the state space vector as

$$\begin{aligned} x &= ({}^i x, {}^i y, {}^i z, {}^i\dot{x}, {}^i\dot{y}, {}^i\dot{z}, \phi, \theta, \psi, {}^b\dot{\phi}, {}^b\dot{\theta}, {}^b\dot{\psi})^T \\ &= (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12})^T \end{aligned} \tag{4.18}$$

As a common practice, the input of the model is defined as

$$\boldsymbol{u} = (u_1, u_2, u_3, u_4)^T \tag{4.19}$$

with

$$
\begin{aligned}
u_1 &= B(w_1^2 + w_2^2 + w_3^2 + w_4^2) \\
u_2 &= B(w_2^2 - w_4^2) \\
u_3 &= B(w_1^2 - w_3^2) \\
u_4 &= D(w_1^2 + w_3^2 - w_2^2 - w_4^2)
\end{aligned}
\tag{4.20}
$$

Combining Eq. (4.13), (4.15), (4.17), (4.18) and (4.20), we obtain the state space representation as follows:

$$
\dot{\boldsymbol{x}} = \begin{bmatrix}
x_4 \\
x_5 \\
x_6 \\
-(\cos x_7 \sin x_8 \cos x_9 + \sin x_7 \sin x_9)\frac{u_1}{M} \\
-(\cos x_7 \sin x_8 \sin x_9 - \sin x_7 \cos x_9)\frac{u_1}{M} \\
G - (\cos x_7 \cos x_8)\frac{u_1}{M} \\
x_{10} \\
x_{11} \\
x_{12} \\
x_{11}x_{12}\frac{J_y - J_z}{J_x} - \frac{J_R}{J_x}x_{11}(w_1 - w_2 + w_3 - w_4) + \frac{L}{J_x}u_2 \\
x_{10}x_{12}\frac{J_z - J_x}{J_y} + \frac{J_R}{J_y}x_{10}(w_1 - w_2 + w_3 - w_4) + \frac{L}{J_y}u_3 \\
x_{10}x_{11}\frac{J_x - J_y}{J_z} + \frac{1}{J_z}u_4
\end{bmatrix}
\tag{4.21}
$$

To simplify Eq. (4.21), we use following substitutions

$$
\begin{aligned}
J_1 &= \frac{J_y - J_z}{J_x} \\
J_2 &= \frac{J_z - J_x}{J_y} \\
J_3 &= \frac{J_x - J_y}{J_z} \\
f(\boldsymbol{u}) &= w_1 - w_2 + w_3 - w_4
\end{aligned}
\tag{4.22}
$$

Therefore, Eq. (4.21) is simplied as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ -(\cos x_7 \sin x_8 \cos x_9 + \sin x_7 \sin x_9)\frac{u_1}{M} \\ -(\cos x_7 \sin x_8 \sin x_9 - \sin x_7 \cos x_9)\frac{u_1}{M} \\ G - (\cos x_7 \cos x_8)\frac{u_1}{M} \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{11}x_{12}J_1 - \frac{J_R}{J_x}x_{11}f(\boldsymbol{u}) + \frac{L}{J_x}u_2 \\ x_{10}x_{12}J_2 + \frac{J_R}{J_y}x_{10}f(\boldsymbol{u}) + \frac{L}{J_y}u_3 \\ x_{10}x_{11}J_3 + \frac{1}{J_z}u_4 \end{bmatrix} \tag{4.23}$$

4.4 Forward-facing (FF) Control

Quadrotor UAVs have high maneuverability. It is capable of flying at various speeds and in different directions, it can fly forward, backward, upward and downward, as well as hover. The commonly used perception sensors on UAVs usually have a limited field of view (FOV), be it a camera, or a LIDAR, or a sonar. In order to perceive the surroundings of the UAV effectively in order to support UAV's collision-free maneuver, it is of vital importance to direct the perception sensors in such a way that it focuses on the direction in which the UAV is flying. One way to achieve this forward-facing behavior of perception sensor is through careful design of the UAV controller. Therefore, a controller or control technique which ensures that the perception sensor is forward-facing during flight is called forward-facing control.

Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid.

Albert Einstein

5

Software Platforms and Frameworks

5.1 Robot Operating System (ROS)

It is commonly known that developing truly robust, general-purpose robot software is hard. Problems that seem trivial to humans often require significant efforts for the robot to solve due to variations of platforms, tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own. Therefore, the Robot Operating System (ROS) [51], an open-source and meta-operating system for robots, was built to encourage collaborative robotics software development, to support code reuse and to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

ROS provides services and programming interfaces one would expect from a

generic operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. Additionally, it provides tools and libraries for obtaining, writing, building and running code across multiple computers.

ROS was designed to be as distributed and modular as possible, so that users can take advantage of its many reusable modules. The modularity of ROS allows users to cherry pick parts that are especially useful for specific use cases, while also providing users with access to lower-level functionality. The distributed processes, which are referred to as "Nodes", of ROS enable software projects to be individually designed and loosely coupled. These processes can be further grouped into "Packages" and "Stacks", which can be easily shared and distributed, further enabling reuse of different modules.

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across nodes) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server.

Despite the importance of reactivity and low latency in robot control, ROS itself is not a real-time OS (RTOS). It is possible, however, to integrate ROS with real-time code. The lack of support for real-time systems has been addressed in the creation of ROS 2.0, a major revision of the ROS API which will take advantage of modern libraries and technologies for core ROS functionality and add support for real-time code and embedded hardware.

5.2 Robotics Simulators

Robot simulation is an essential tool for robot development. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Robot simulator

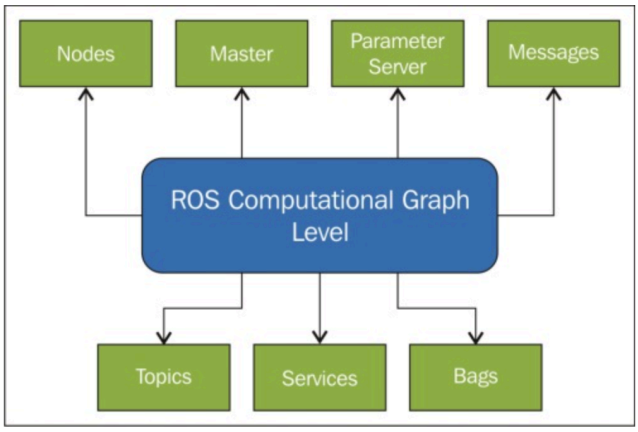


Figure 5.1.1: ROS Communication Layer [26]

allows for 3D modeling and rendering of a robot and its environment, and its realistic robot motion generations enable robotics practitioners to debug software algorithms off-line before finally deploying on real robots, thus saving both costs and time. The success of off-line programming depends on how similar the simulation is to the reality.

Two popular robotics simulators are V-REP (currently as CoppeliaSim) and Gazebo. V-REP is a commercial solution (with a free education version) provided by Coppelia Robotics, while Gazebo is an open source solution maintained by Open Source Robotics Foundation. Both Simulators can use multiple high-performance physics engines such as ODE, Bullet etc., and offer the ability to accurately and efficiently simulate populations of robots in complex indoors and outdoor environments. Additionally, both simulators come with a variety of existing robot and sensor models, as well as environment models. However, due to the commercial nature of V-REP simulator, its documentation is rather limited, especially in comparison to Gazebo. Thus Gazebo is easier to access, especially for new users, thanks to its rich documentation and active community. Besides, Gazebo is the default simulator used in ROS framework. There is mature interface between ROS and Gazebo, while interfacing between ROS and V-REP can sometimes be a tricky task. Furthermore, Gazebo and ROS already have a large base of community-developed plugins and code. While

V-REP, on the other hand, does not have the same amount of ready-to-use components.

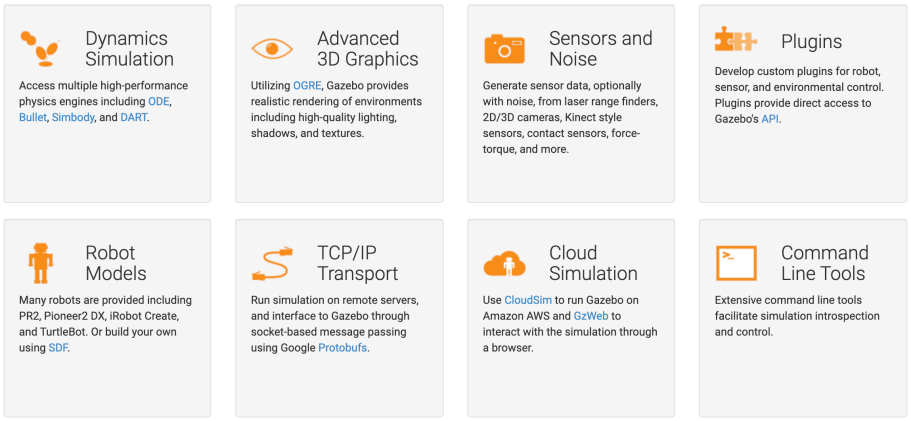


Figure 5.2.1: Gazebo Features

5.3 Hector Quadrotor Framework

During the process of learning to fly UAVs or development of UAV controllers, crashing is unavoidable. From the first-time learning of flight control to testing new hardware or flight algorithms, the resulting failures can lead to a huge costs due to broken hardware components. To that end, UAV simulation is desirable.

A simulated quadrotor UAV for the ROS Gazebo environment has been developed by the Team Hector Darmstadt from Technische Universität Darmstadt. This simulated quadrotor, called Hector Quadrotor, is enclosed in the hector_quadrotor metapackage. This metapackage contains the URDF description for the quadrotor UAV, its flight controllers, and launch files for running the quadrotor simulation in Gazebo.

Advanced uses of the Hector Quadrotor simulation allows the user to record sensor data such as LIDAR, depth camera, and many more. The quadrotor simulation can also be used to test flight algorithms and control approaches in simulation.

Quadrotor UAVs have successfully been used both in research and for commercial applications in recent years and there has been siginificant progress in the design of robust control software and hardware. Nevertheless, testing of prototype UAV systems still means risk of damage due to failures. Motivated by this, a system for the comprehensive simulation of quadrotor UAVs is presented in this chapter. Unlike existing solutions, the presented system is integrated with ROS and the Gazebo simulator. This comprehensive approach allows simultaneous simulation of diverse aspects such as flight dynamics, onboard sensors like IMUs, external imaging sensors and complex environmentss. The dynamic model validated by the quadrotor has been parameterized using wind tunnel tests and validated by a comparison of simulated and real flight data. The applicability for simulation of complex UAV systems is demonstrated using LIDAR-based and visual SLAM approaches available as open source software.

Quadrotor UAVs have successfully been used both in research and for commercial applications in recent years. Impressive results have been shown using quadrotor aircraft of various sizes and in different scenarios. The inherently instable nature of quadrotor flight can lead to loss or damage of UAVs easily, especially when evaluating prototype soft- or hardware. The lack of a simulation environment for quadrotor that covers realistic flight dynamics, camera and range sensors and an easy integration with existing robotic middleware solutions motivated this work. We present a comprehensive framework to simulate our quadrotor, that has been developed during the last few years. It is based on the Gazebo open source simulator and the Robot Operating System (ROS), that has become a de facto standard in robotics research and facilitates integration of contributions by other researchers. Common sensors for autonomous robots like LIDAR devices, RGB-D and stereo cameras are already available for Gazebo and can be attached to the robot, while plugins for other, more UAV-specific sensors like barometers, GPS receivers and sonar rangers have been added as part of this work.

As we aim at comprehensive simulation of all relevant components including low level sensing, system dynamics and control, we provide an overview of these parts independently. Gazebo provides a multi-robot simulation environment

including dynamics simulation, which is provided by the ODE or bullet physics engines. While the simulator considers gravity, contact forces and friction by its own, it does not cover aerodynamics and propulsion systems that are especially required for aerial vehicles. A plugin systems enables the user to add custom controllers for simulated robots and sensors or to influence the environment.

The robot geometry has been modeled using the open source software Blender. To be able to provide different colors (both texture or material based) for the model, the visual geometry is provided using the COLLADA format, while the collision geometry is modeled as a .stl mesh. The model is designed to have a low polygon count and still retain the relevant aspects of quadrotor geometry. As a trade-off between visual fidelity, collision detection and dynamics modeling needs, the propellers are modeled as discs.

different aspects of simulation are validated using experiments in this section. We also show examples of comprehensive simulation scenarios using the flight dynamics model as well as leveraging existing ROS open source software.

To validate the dynamics model, we let both the real and simulated UAV perform a test trajectory consisting of transitions between different velocities. All measurable variables of the real quadrotor show the same characteristics as the corresponding simulated counterparts.

The best and most beautiful things in the world cannot be seen or even touched – they must be felt with the heart.

Helen Keller

6 Perception

In this chapter, we elaborate on our proposed sensing solution. We start by introducing the commonly used sensor technologies for UAV and the world representations as well as the justifications of our respective choices. Afterwards we will introduce in detail the proposed sensing system.

6.1 Sensor Technologies and Selection

6.1.1 Sensor Technologies

In robotics, there are three types of commonly used perception sensors: sonar, LIDAR and camera.

SONAR

Sonar, also called ultrasound sensor, is a popular sensor in robotics which employs acoustic pulses at ultrasonic frequencies and their echoes to measure the range to an object as well as the direction. Since the sound speed is usually known, the object range is proportional to the echo's travel time. Its popularity is due to its low cost, light weight, low power consumption, and low computational effort, compared to other ranging sensors.

LIDAR

LIDAR, short for light detection and ranging, is also called laser range finder or laser scanner. In many respects recent progress on environment modeling and navigation are thanks to the emergence of low-cost and high-fidelity LIDAR systems.

Similar to sonar sensor, it also uses TOF (time of flight) to measure the distance to the object, however, based on the speed of light. Additionally, since robotics applications usually require more information, range data is usually supplied as a vector of range to surfaces lying in a plane, or as an image. To obtain these denser representations, the laser beam is swept by a set of mirrors rather than moving the laser and detector themselves (mirrors are lighter and less prone to motion damage). The most common technologies for this are using a stepper motor (for program-based range sensing) or rotating or oscillating mirrors for automatic scanning.

Camera

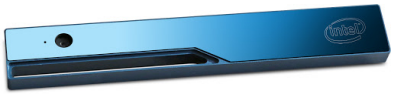
Imaging sensors (cameras) are a rich source of information for perception. Imaging sensors come in a wide variety of configurations, varying according to image geometry, image resolution, sensor technology and the range of sensed spectral bands. A traditional imaging sensor contains an optical system that

focuses light on a planar imaging array. In most cases, this system can be modeled using the classical pinhole camera model.

There are two important types of cameras, namely CCD camera and CMOS camera. The basis of semiconductor cameras is the "inner photoelectric effect": In certain materials electrons are set free under photon absorption (light), and the amount of electrons is correlated with the amount of photons. While CMOS sensors come with smaller size, bigger dynamics and no blooming effects, the CCD sensors offer higher photo sensitivity, higher uniformity and less noise. However, CMOS camera is usually more expensive than CCD camera.

Stereo cameras are able to recover depth information from images, just as humans are capable of perceiving depth with their two eyes. In a human's case, a distant object is projected onto the retina of the left and right eyes. Though the object's projections are from different perspectives, the human brain is able to generate a sense of depth for the object by analyzing the two images. Similarly, stereo cameras can reconstruct a depth map from at least two camera images of a 3D scene from different vantage points, and depth information is inferred by matching corresponding points in both images. [8]

RGB-D (red green blue - depth) cameras are increasingly more popular as they become widely available and cost effective. They are active ranging cameras which combine images from an optical camera with a dense range map. An RGB-D camera system usually consists of three cameras: a light projector, a detect camera and a visible light camera. Depth information is recovered by firstly performing triangulation between a light projector that emits a pattern with known structure on into the scene and a camera that views the scene and detects the pattern, and then matching or correlating the pattern elements with known relationship between the light projector and detect camera. This camera pair for depth recovery typically operate with infrared to avoid the emitted pattern being visible to human eyes. While the third camera is usually used to acquire a corresponding color image. Therefore, these three cameras combined would produce RGB-D images. [58]



(a) Intel Realsense r200

(b) ZED Camera

(c) Hokuyo URG-04lx-UG01

(d) Hokuyo UTM-30LX

Figure 6.1.1: Sensor Technologies

6.1.2 Sensor Selection

In order to select a suitable perception sensor for our application, we need to take into account following aspects:

- size and weight
- detection range
- field of view (perception coverage)
- accuracy of detected depth
- detection frequency
- economical cost
- computational cost

We would like to find an affordable sensor with adequate performance, possibly with potential to become cheaper with time.

Table 6.1.1: Sensor Comparison

Sensors	Category	Dimension(WxDxH) Weight	Range	FOV	Frequency (max)	Cost
Hokuyo URG-04LX-UG01	2D LIDAR	$50 \times 50 \times 70mm$ Approx. $160g$	$40m$	240°	$10Hz$	1080 USD
Hokuyo UTM-30LX	2D LIDAR	$60mm \times 60mm \times 87mm$ $210g$	$0.1 - 30m$	270°	$40Hz$	4500 USD
Velodyne Puck LITE	3D LIDAR	ø103.3mm $\times 71.7mm$ Approx. $590g$	$100m$	$360^{\circ}(H) \times 30^{\circ}(V)$	$5Hz - 20Hz$	c.a. 4000 USD
Intel RealSense r200	RGB-D Camera	$101.6mm \times 9.6mm \times 3.8mm$ $65g$	$0.5m - 3.5m$	$59^{\circ}(H) \times 46^{\circ}(V) \times 70^{\circ}(D)$	$60Hz$	c.a. 100 USD
ZED	Stereo Camera	$175 \times 30 \times 33mm$ $135g$	$0.3 - 25m$	$90^{\circ}(H) \times 60^{\circ}(V) \times 100^{\circ}(D)$	$100Hz$	349 USD

In general, the performance of vision-based approaches is strongly subject to illumination conditions especially when in outdoors, and demands high computation power which makes it difficult for the task of detection and tracking to be completely onboard.

In comparison to vision-based approaches, LIDAR-based systems are insensitive to lighting conditions and requires significantly less processing time especially when a 2D LIDAR is used. In addition, LIDARs generally have wider field of view (FOV) horizontally than cameras, and thus can be aware of threats not only in the front but also from the sides.

In comparison to single-sensor approach, sensor fusion systems are usually more complex and demands more computation power.

6.2 World Representation

6.2.1 Occupancy Grid

Depending on the application, the occupancy grid is usually two dimensional or three dimensional. The tessellation can either be uniform or tree-based using quadtree or octree. The tree-based methods are in particular well suited for handling of inhomogeneous and large scale data sets.

In a grid model each cell contains a probability over the parameter set. As an example, when using the grid model for representation of a physical environment, the cell specifies occupied (O) or free (F) and also encodes the probability $P(occupied)$. Initially where there is no information the grid is initialized to $P(O) = 0.5$ to indicate unknown. It is further assumed that sensor models are available that specify $P(R|R_{i,j})$, i.e., the probability of detection objects for a given sensor and location. Using Bayes theorem it is now possible to update the grid model.

The grid-based model has been widely used in mobile robotics and in medical imaging where image volumes are quite common. Volume models can be relative

large. As an example a millimeter-resolution grid model of the human head requires 4GB of storage, and thus demands significant computational resources for maintenance. [58]

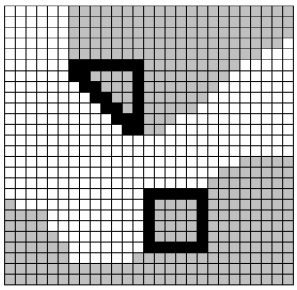


Figure 6.2.1: Example of 2D Occupancy Grid Map

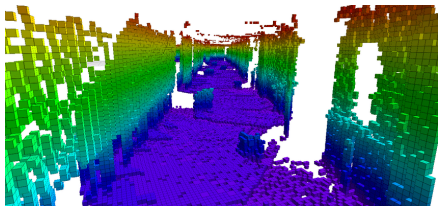


Figure 6.2.2: Example of 3D Occupancy Grid Map - OctoMap

6.2.2 Symbolic Model

In many cases there is an interest in representing the environment in such a way that reduces the storage and computational requirements demanded by the occupancy grid. Feature-based representations which employ points, lines or planes etc. to model the environment are not suitable for dynamic obstacle tracking in a complex outdoor environment. Thus, we propose to use symbolic shapes such as cylinder to represent objects in the UAV's near surroundings for the purpose of obstacle avoidance. Since our objective is not object recognition or identification, but to generate a dynamic local map of supporting collision-free navigation, an approximation of the real underlying model of the object is deemed acceptable.

The motivation of adopting infinite height cylinder for modeling the environment is threefold. Firstly, smaller objects such as moving persons, trees and small fixtures in the environment are generally in cylinder-like shapes. Secondly, this model of the environment is easy and inexpensive to compute and maintain, in comparison to occupancy grids. Thirdly, this model is more suitable for supporting fast short distance collision avoidance and path planning algorithms because of its geometrical simplicity and ability to enclose all LDIAR points incident on an object, in comparison to commonly used feature models such as lines and shapes such as polygons. Even though modelling objects with infinite heights could cause some loss in the operational or maneuverable space of UAV, it is worthwhile taking into account the benefits obtained from this model simplification.

6.3 2D LIDAR based Object Detection and Tracking

In this section, we focus on a simple, fast and robust 2D LIDAR-based object detection and tracking approach using cylinder, i.e. circle with infinite height, to model the obstacles in the environment. Our major contribution is that we proposed a heuristic method for modeling object in a compact form and subsequently using the model as object tracker. Another contribution is the application of this 2D LIDAR-based approach onboard of a UAV in experimental tests.

6.3.1 Overview

We consider that environment can be modeled with cylindrical shapes of infinite height, this assumption is valid in many aforementioned outdoor application scenarios. With this assumption, we could further simplify the model of the environment as a set of circles on the horizontal plane. The sensing functional module provides such an obstacle map (represented with circles) of the near surroundings of the UAV utilizing an onboard 2D LIDAR. It consists of three

major submodules, namely, scan pre-processing, shape generation and mapping and tracking, as shown in Fig. 6.3.1.

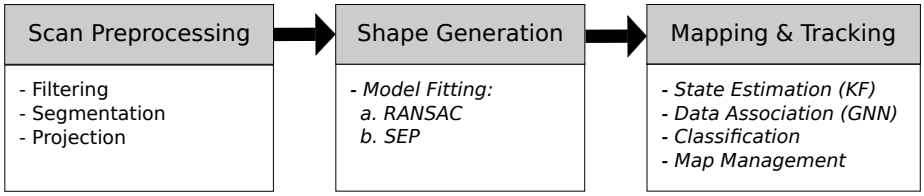


Figure 6.3.1: Workflow of Sensing Functional Module

6.3.2 LIDAR SENSOR UTM-30LX

UTM-30LX uses laser source ($\lambda = 905nm$) to scan 270° semicircular field, as in Figure 6.3.2. It measures distance to objects in the range and coordinates of those point calculated using the step angle. Sensor's measurement data along with the angle are transmitted via communication channel.

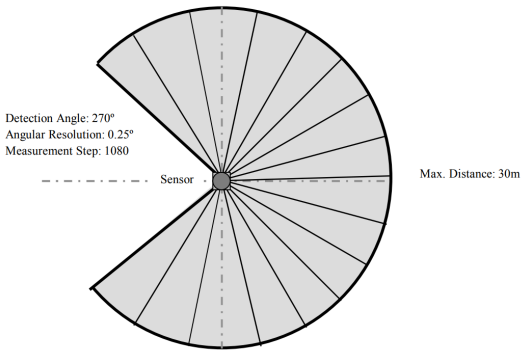


Figure 6.3.2: UTM-30LX LIDAR Field of View

Table 6.3.1: Hokuyo UTM-30LX LIDAR Specification

Model	UTM-30LX
Light Source	Laser Semiconductor $\lambda = 905nm.LaserClass1$
Supply Voltage	12VDC$\pm$10%
Supply Current	Max: 1A, Normal: 0.7A
Power Consumption	Less than 8W
Detection Range and Detection Object	Guaranteed Range: 0.1 ~ 30m (White Kent Sheet) Maximum Range: 0.1~60m Minimum detectable width at 10m: 130mm (Vary with distance)
Accuracy	Under 3000lx: White Ken Sheet:$\pm 30mm(0.1m\ to\ 10m)$ Under 100000lx: White Kent Sheet:$\pm 50mm(0.1m\ to\ 10m)$
Measurement Resolution and Repeated Accuracy	1mm 0.1 - 10m: $\sigma < 10mm, 10 - 30m : \sigma < 30mm(WhiteKentSheet)$ Under 3000lx: $\sigma < 10mm(WhiteKentSheetupto10m)$ Under 100000lx:$\sigma < 30mm(WhilteKentSheetupto10m)$
Scan Angle	270°
Angular Resolution	$0.25^\circ(360^\circ/1440)$
Scan Speed	25ms (Motor speed: 2400rpm)
Interface	USV Ver2.0 Full Speed(12Mbps)
Ambient Condition (Temperature, Humidity)	-10$^\circ C\tilde{}$ + 50 $\circ$ C Less than 85%RH (Without Dew, Frost)
Storage Temperature	-25~75$\circ C$
Environmental Effect	Measured distance will be shorter than the actual distance under rain, snow and direct sunlight
Vibration Resistance	10~55Hz Double amplitude 1.5mm in each X, Y, Z axis for 2hrs. 55~200Hz 98m/s^2*sweepof2minineachX*, *Y*, *Zaxisfor1hrs*.
Impact Resistance	196m/s^2*IneachX*, *Y*, *Zaxis10times*
Protective Structure	Optics: IP64
Insulation Resistance	10M$\Omega DC500VMegger$
Weight	210g (Without cable)
Case	Polycarbonate
External Dimension (W$\times D \times H$)	60mm$\times 60mm \times 87mm$ MC-40-3127

6.3.3 Range Limiting and Ground Removal

One full LIDAR scan is an ordered sequence of N (defined by the specification of the LIDAR sensor utilized) measurement points, denoted by $\boldsymbol{p}$. Each raw measurement point is represented in the form of Polar coordinates (r_n, θ_n), where r_n and θ_n denote the range and angle of each measurement point. Therefore, one complete LIDAR scan represented in mathematical form is as follows

$$\boldsymbol{p} \triangleq \{\boldsymbol{p}_n = (r_n, \theta_n)\}, n \in [1, N] \tag{6.1}$$

In practice, usually not all measurements from one LIDAR scan are usable or useful depending on the specific application scenario. Our algorithm provides a range filter which removes "invalid" measurements that are either invalid by value (represented by NaN) or not within certain predefined range thresholds (configurable) and/or height bounds (configurable).

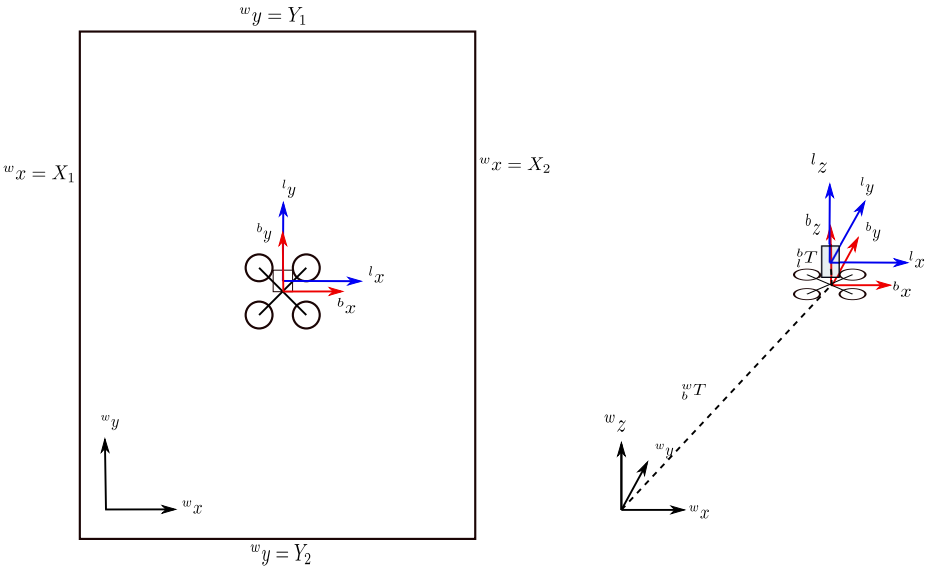


Figure 6.3.3: Illustration of range limit function

As shown in Figure 6.3.3, ${}^{b}_{l}\boldsymbol{T}$ and ${}^{w}_{b}\boldsymbol{T}$ are the homogeneous transform matrices which transform coordinates from LIDAR frame to UAV body frame, and from

body frame to world frame respectively. For example, transforming a point in LIDAR frame e.g. ${}^{l}p$ into world frame can be expressed mathematically as follows:

$$ {}^{w}\boldsymbol{p} = {}^{w}_{b}\boldsymbol{T}^{b}_{l}\boldsymbol{T}^{l}\boldsymbol{p} \tag{6.2} $$

After filtering the LIDAR measurements with range thresholds $R_{thresmin}$ and $R_{thresmax}$, we firstly convert the LIDAR measurements into point cloud in the LIDAR frame, which is then transformed into world frame, where we limit the detecting range of the LIDAR measurements to our experimental arena. If we assume ${}^{w}\boldsymbol{p} = \left[{}^{w}p_x \ {}^{w}p_y \ {}^{w}p_z\right]^T$, then a LIDAR measurement is kept if following criteria are fulfilled:

$$ X_1 \leq {}^{w}p_x \leq X_2 \tag{6.3} $$

$$ Y_1 \leq {}^{w}p_y \leq Y_2 \tag{6.4} $$

where X_1, X_2, Y_1 and Y_2 are given by the four borders of the experimental arena as shown in Figure 6.3.3.

If following criteria is not satisfied, then we consider that the measurement point incidents on the ground and it is thus considered as invalid measurement.

$$ {}^{w}p_z \geq Z_{thres} \tag{6.5} $$

For ease of obtaining angular position of each measurement in the Polar space later on, the algorithm keeps the original index in the range measurement array by replacing all the "invalid" range measurements with NaN. Meanwhile, the indices of all the "valid" (as apposed to "invalid") measurements are stored, and this indices set $\boldsymbol{v}$ of "valid" measurements has size M.

$$ \boldsymbol{v} \triangleq \{v_i\},\ i \in [1, M],\ 0 \leq M \leq N \tag{6.6} $$

6.3.4 Segmentation

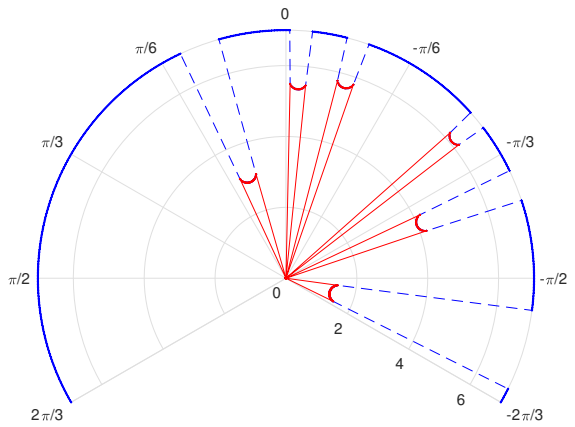


Figure 6.3.4: 2D Polar space of one LIDAR scan

In order to extract objects from LIDAR measurements, our algorithm firstly separate each scan into segments which potentially represent objects in the environment, for example, Fig. 6.3.4 shows the segmentation results of one LIDAR scan in Polar space.

Our segmentation algorithm utilizes a variant of the PDBS (Point-Distance-Based Segmentation) methods [49], where not merely Euclidean distance, as in Eq. 6.7, is used for segmentation, but it is combined with a distance from Polar space as in Eq. 6.8, that is, distance between indices of two adjacent "valid" measurements (from set $\boldsymbol{v}$). The latter is used to reduce the impact of distance between LIDAR and objects on the segmentation results.

$$d_1(v_i, v_{i+1}) = \|v_{i+1} - v_i\| \tag{6.7}$$

$$d_2(p_{v_i}, p_{v_{i+1}}) = \sqrt{r_{v_i}^2 + r_{v_{i+1}}^2 - 2r_{v_i} r_{v_{i+1}} cos\Delta\alpha} \tag{6.8}$$

The first "valid" LIDAR measurement initiates a new segment, and its distance (using our custom distance metric) to the next "valid" measurement is calculated and compared with the distance threshold for separating segments or joining

segments. If one of following criteria is satisfied (as in Eq. (6.9) and (6.10)), then we consider the next measurement is joined into current segment. Otherwise, a new segment is initiated with the next measurement. Then repeat this process for the whole LIDAR scan in order to find out the segments.

$$d_1(v_i, v_{i+1}) \leq C_1 \;\&\; d_2(p_{v_i}, p_{v_{i+1}}) \leq D_1 \tag{6.9}$$

$$C_1 < d_1(v_i, v_{i+1}) \leq C_2 \;\&\; d_2(p_{v_i}, p_{v_{i+1}}) \leq D_2 \tag{6.10}$$

After obtaining all the segments, the ones with less than three measurement indices are considered as having indices of spurious measurements, and are therefore discarded. Finally we obtain a segments-set $\boldsymbol{s}$, and an illustration of the segmentation result is shown in Fig. 6.3.5.

$$\begin{aligned} \boldsymbol{s} \triangleq \{\boldsymbol{s}_k = \{v_j\},\, j \in [h, g], 0 \leq h < g \leq M\}, \\ k \in [1, K],\; 0 \leq K \leq M \end{aligned} \tag{6.11}$$

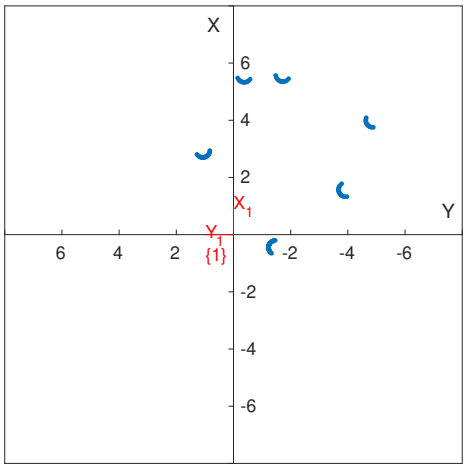


Figure 6.3.5: Segmentation result of one LIDAR scan in 2D Cartesian space

6.3.5 Projection onto 2D Plane

Multirotor aerial robot has poses with 6 DOF in space. Thus slight pitch or roll of the robot will cause significant change in the shape of the measurements in LIDAR measurement plane, as shown in Fig. 6.3.6: the intersection of pitched LIDAR measurement plane on the cylinder is an ellipse, and the corresponding incident LIDAR measurements reside on the ellipse curve. Therefore, it would be inappropriate if we directly model objects with circles in the LIDAR measurement plane, because the size of estimated circle would already vary significantly without taking into account variation caused by noise. Thus we project the LIDAR scan onto the ground plane, in order to obtain a relative stable model of the object.

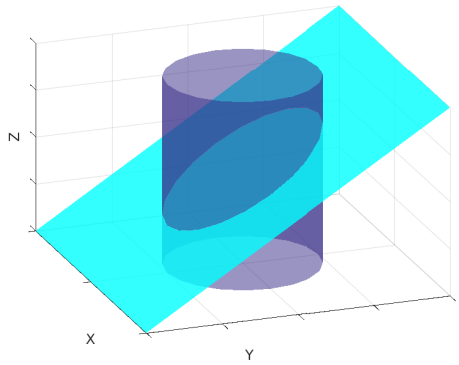


Figure 6.3.6: Intersection of LIDAR scan with cylinder-shaped object

6.3.6 Object Model Fitting

RANSAC Algorithm

Random sample consensus (RANSAC) is a learning technique to estimate parameters of a mathematical model by random sampling of observed data. Given a dataset whose data elements contain both inliers and outliers, RANSAC

uses the voting scheme to find the optimal fitting result. Data elements in the dataset are used to vote for one or multiple models. The implementation of this voting scheme is based on two assumptions: the noisy features will not vote consistently for any single model (few outliers) and there are enough features to agree on a good model (few missing data). The RANSAC algorithm is essentially composed of two steps that are iteratively repeated:

1. In the first step, a sample subset containing minimal data items is randomly selected from the input dataset. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. The quantity of the sample subset is usually the smallest sufficient to determine the model parameters.

2. In the second step, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the model within some error threshold that defines the maximum deviation attributable to the effect of noise.

The set of inliers obtained for the fitting model is called the consensus set. The RANSAC algorithm will iteratively repeat the above two steps until the obtained consensus set in certain iteration has enough inliers.

The input to the RANSAC algorithm is a set of observed data values, a way of fitting some kind of model to the observations, and some confidence parameters. RANSAC achieves its goal by repeating the following steps:

1. Select a random subset of the original data. Call this subset the hypothetical inliers.

2. A model is fitted to the set of hypothetical inliers

3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the consensus set.

4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.

5. Afterwards, the model may be improved by reestimating it using all members of the consensus set

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size. In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.

Implementation

We use cylinders to model objects in the environment. Since cylinders are essentially circles on the horizontal plane with certain heights, and we consider all objects in the environment having infinite height, therefore we only need to concentrate on circle as model to fit the LIDAR data projected onto the horizontal plane.

Our model fitting algorithm is twofold. We use RANSAC as the first line of model fitting for each segment, and when no appropriate model can be found from RANSAC estimation, then the second line of model fitting comes to play, which ensures that all points in the segment are covered by (within) the resulting circle.

RANSAC algorithm is employed to estimate circle. We choose RANSAC algorithm for model fitting because it is robust against noise and outliers. The raw LIDAR measurements are usually noisy, and even for strictly cylinder-shaped object in real world there would be outliers or somewhat shape-distortion in the raw LIDAR measurements. Additionally, even for LIDAR measurements from non-circle-shaped object in the environment, RANSAC can still make a circle estimation to approximately model the object in the environment. Since our goal is not object recognition or identification, but to generate a dynamic map for the purpose of collision-free navigation, an approximation of the real underlying

model of the object is deemed acceptable.

We call the second line method Starting-Ending-Points (SEP) method. We take the starting and ending points in the segment, and calculate the middle point of them, and then use this middle point as the center point of the best estimate. Then, we calculate the distance between this center point and each point in the segment in order to obtain the maximum distance between the center point and the segment points. Finally this obtained maximum distance is used as the radius of the best circle estimate, as shown in Figure 6.3.7. At the end of model fitting

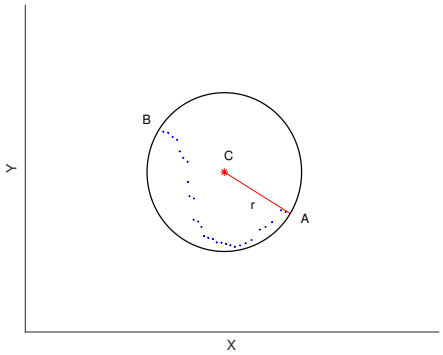


Figure 6.3.7: Model fitting of a scan segment

process, we obtain an object list ***o*** from segments set ***s***. The detailed description of this functionality is presented as pseudo code as in Algorithm 1.

6.3.7 Kalman Filter

Kalman Filter Algorithm

Kalman Filter is one of the most popular techniques for state estimation. The Kalman Filter was invented in 1950s by Rudolph Emil Kalman, as a technique for filtering and prediction in linear systems.

The Kalman Filter represents beliefs by the moments representation [9]: At time t, the belief is represented by the mean μ_t and the covariance Σ_t. Posteriors are

Algorithm 1: Object_Model_Fitting($\boldsymbol{s}_k^i$)

Data: $I_{max} \leftarrow$ maximum number of iterations;
$R_{max} \leftarrow$ maximum radius value;
$N_{seg} \leftarrow$ number of points in segment;
$i \leftarrow 0$;
$n_{in,max} \leftarrow 0$;
$\boldsymbol{c}_{best} \leftarrow [0, 0, 0]$;
Result: Estimated circle parameters $[x, y, r]^T$

```
while i ≠ I_max do
    p_3 = drawThreePointsFromSegment();
    c = calcCirGeometricParameters(p_3);
    n_in = calcNumberOfInliers(c);
    if n_in > n_in,max then
        c_best = c;
        n_in,max ← n_in;
    end
    i ← i + 1;
end
d_min = calcMinDistFromSegPointToLidarPos();
d = calcDistFromCirCenterToLidarPos(c_best);
if c_best.r > R_max or n_in < (1/2)N_seg or d > d_min then
    [p_start, p_end] = GetSEPInSegment(s_k^i);
    c_best.x = (p_start.x + p_end.x)/2;
    c_best.y = (p_start.y + p_end.y)/2;
    c_best.r = calcMaxDistToSegPoints(c_best.x, c_best.y);
end
```

Gaussian if the Markov assumptions of the Bayes filter hold, as well as the following three properties:

1. The next state probability $p(x_t|u_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise, which is expressed in mathematical form as follows

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \tag{6.12}$$

where u_t is the control vector at time t, and x_t and x_{t-1} are both state vectors with following form

$$x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n,t} \end{bmatrix} \quad \text{and} \quad u_t = \begin{bmatrix} u_{1,t} \\ u_{2,t} \\ \vdots \\ u_{m,t} \end{bmatrix} \tag{6.13}$$

A_t is a square matrix of size $n \times n$ with n being the dimension of the state vector x_t, while B_t is a matrix of size $n \times m$ with m being the dimension of the control vector u_t. The state transition function is linear in its arguments, state and control vectors, thus Kalman Filter assumes linear system dynamics. The random variable ϵ_t is a Gaussian random vector that models the random noise in the state transition, and it has the same dimension as the state vector. It has zero mean and its covariance will be denoted as R_t.

The state transition probability $p(x_t|u_t, x_{t-1})$ is produced by $A_t x_{t-1} + B_t u_t$ and R_t as follows

$$\begin{aligned} &p(x_t|u_t, x_{t-1}) \\ &= det(2\pi R_t^{-\frac{1}{2}} exp\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\} \end{aligned} \tag{6.14}$$

2. The measurement probability $p(z_t|x_t)$ must also be linear in its arguments

with added Gaussian noise as follows

$$z_t = C_t x_t + \delta_t \tag{6.15}$$

where C_t is a matrix of size $k \times n$ with k being the dimension of the measurement vector z_t. The vector δ_t describes the measurement noise. The distribution of δ_t is a multivariate Gaussian with zero mean and covariance Q_t. The measurement probability $p(z_t|x_t)$ is thus given by the following multivariate normal distribution

$$p(z_t|x_t) = det(2\pi Q_t)^{-\frac{1}{2}} exp\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\} \tag{6.16}$$

3. The initial belief $bel(x_0)$ must be normal distribution. The mean of this belief is denoted by μ_0 and the covariance by Σ_0, and thus it has following form

$$bel(x_0) = p(x_0) = det(2\pi\Sigma_0)^{-\frac{1}{2}} exp\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\} \tag{6.17}$$

Algorithm 2 depicts the Kalman Filter algorithm. In Kalman Filter the belief $bel(x_t)$ at time t is represented by its mean μ_t and covariance Σ_t. The input of the Kalman Filter is the belief at time $t-1$, represented by μ_{t-1} and Σ_{t-1}. In order to update these parameters, Kalman Filters require the control u_t and the measurement z_t. The output of Kalman Filter is the belief at time t, which is represented by μ_t and Σ_t.

In lines 2 and 3 in the Algorithm 2, $\bar{\mu}$ and $\bar{\Sigma}$ are the predicted mean and covariance representing the predicted belief $barbel(x_t)$ at one time step later, but before incorporating the measurement z_t. Subsequently, the belief $\bar{bel}(x_t)$ is transformed into the desired belief $bel(x_t$ from lines 4 to 6, by incorporating the measurement z_t this time. The variable K_t is called Kalman gain, which describes how much influence the measurement would have on the new state estimate.

In line 5 the mean is adjusted in proportion to the Kalman gain K_t and the difference between the actual measurement z_t, and the predicted measurement.

Finally, the new covariance of the posterior belief is calculated in line 6, which adjusts for the information gain by incorporating the measurement.

Algorithm 2: Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)

Result: μ_t, Σ_t

1 $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$;
2 $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$;
3 $K_t = \bar{}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$;
4 $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$;
5 $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

Implementation

In order to track obstacles in the environment, we use discrete time Kalman Filter (KF) to estimate the states of detected objects. Each newly detected object will initiate a Kalman filter. We denote the state of the environment (also referred to as map in this work) as $\boldsymbol{x}_k$, which is composed of a list of objects states. At time instant t_k it can be formulated as follows:

$$\boldsymbol{x}_k = \begin{cases} \{\boldsymbol{x}_k^i\} & \text{for } i \in [1, n],\ n \geq 1,\ k \geq 1 \\ \emptyset & \text{for } \textit{others} \end{cases} \tag{6.18}$$

where n is the number of obstacles in the map and

$$\boldsymbol{x}_k^i = \left[{}^w x_k^i \ {}^w y_k^i \ r_k^i \ {}^w \dot{x}_k^i \ {}^w \dot{y}_k^i \ s_k^i\right]^T \tag{6.19}$$

is the state vector for the ith object in the environment, with ${}^w x_k^i$ and ${}^w y_k^i$ being the position of the object in world coordinates, r_k^i the circle radius, ${}^w \dot{x}_k^i$ and ${}^w \dot{y}_k^i$ the linear velocities, and s_k^i the ID of the object.

Since the goal of our algorithm is to track the detected objects in the environment for the purpose of collision-free navigation, it would be useful to estimate not only the position and shape (circle radius) of the objects in the environment, but also their velocities. Due to the fact that once an ID number

has been assigned to an object in the map, it will not change over time as the way other state elements do. Thus, the state element s_k^i is treated separately and in the following description we will only concentrate on the other five state elements that change over time.

For each object in the map, a Kamlan Filter is initiated to estimate its state. The process model and measurement model of each Kalman Filter are formulated as follows:

$$\boldsymbol{x}_{k+1}^i = \boldsymbol{A}_k^i \boldsymbol{x}_k^i + \varepsilon_{p,k} \tag{6.20}$$

$$\boldsymbol{z}_k^i = \boldsymbol{C}_k^i \boldsymbol{x}_k^i + \varepsilon_{m,k} \tag{6.21}$$

where

$$\boldsymbol{A}_k^i = \begin{bmatrix} 1 & 0 & 0 & \Delta t_k & 0 \\ 0 & 1 & 0 & 0 & \Delta t_k \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.22}$$

$$\boldsymbol{C}_k^i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.23}$$

Δt_k is the time interval between two consecutive prediction steps and is defined as follows

$$\Delta t_k = t_k - t_{k-1} \tag{6.24}$$

The Gaussian random variables $\varepsilon_{p,k}$ and $\varepsilon_{m,k}$ model the process noise and measurement noise respectively, and they are defined as follows

$$\varepsilon_{p,k} \sim \mathcal{N}(0,\ \boldsymbol{Q}_{p,k}) \tag{6.25}$$

$$\boldsymbol{\varepsilon}_{m,k} \sim \mathcal{N}(0,\ \boldsymbol{Q}_{m,k}) \tag{6.26}$$

where $\boldsymbol{Q}_{p,k}$ and $\boldsymbol{Q}_{m,k}$ are their covariance matrices with following forms

$$\boldsymbol{Q}_{p,k} = \begin{bmatrix} \sigma^2_{p,\dot{x}}\Delta t_k & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{p,\dot{y}}\Delta t_k & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{p,r} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{p,\dot{x}} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{p,\dot{x}} \end{bmatrix} \tag{6.27}$$

$$\boldsymbol{Q}_{m,k} = \begin{bmatrix} \sigma^2_{m,x} & 0 & 0 \\ 0 & \sigma^2_{m,y} & 0 \\ 0 & 0 & \sigma^2_{m,r} \end{bmatrix} \tag{6.28}$$

We could also formulate the above models in vectorized form as follows

$$\boldsymbol{X}_{k+1} = \boldsymbol{A}_k \boldsymbol{X}_k + \boldsymbol{\varepsilon}_{p,k} \tag{6.29}$$

$$\boldsymbol{Z}_k = \boldsymbol{C}_k \boldsymbol{X}_k + \boldsymbol{\varepsilon}_{m,k} \tag{6.30}$$

with

$$\boldsymbol{X}_k = \left[\boldsymbol{x}_k^0\ \boldsymbol{x}_k^1\ \boldsymbol{x}_k^i\ \cdots\ \boldsymbol{x}_k^n\right]^T \tag{6.31}$$

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{A}_k^0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \boldsymbol{A}_k^1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 & \vdots & \vdots \\ 0 & 0 & 0 & \boldsymbol{A}_k^i & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \boldsymbol{A}_k^n \end{bmatrix} \tag{6.32}$$

$$\boldsymbol{C}_k = \begin{bmatrix} \boldsymbol{C}_k^0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \boldsymbol{C}_k^1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 & \vdots & \vdots \\ 0 & 0 & 0 & \boldsymbol{C}_k^i & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \boldsymbol{C}_k^n \end{bmatrix} \tag{6.33}$$

6.3.8 Data Association

The track updating process typically begins with a procedure that is used to choose the best observation to track association. This procedure is known as data correlation and is conventionally comprised of two steps called gating and association.

Gating and Association

Gating is a coarse test for eliminating unlikely observation-to-track pairing. A gate is formed around the predicted position. All measurements that satisfy the gating relationship fall within the gate and are considered for track update. The manner in which the observations are actually chosen to update the track depends on the association method but most association methods utilize gating in order to reduce later computation.

In the dense target environment additional logic (i.e. association) is required when an observation falls within the gates of multiple target tracks or when multiple observations fall within the gate of a target track. The optimal assignment minimizes a total distance function which is the sum of the distances for all the individual assignments. Thus it is necessary to define a distance measure from the predicted positions of track i to observation m_j, as shown in Figure 6.3.8

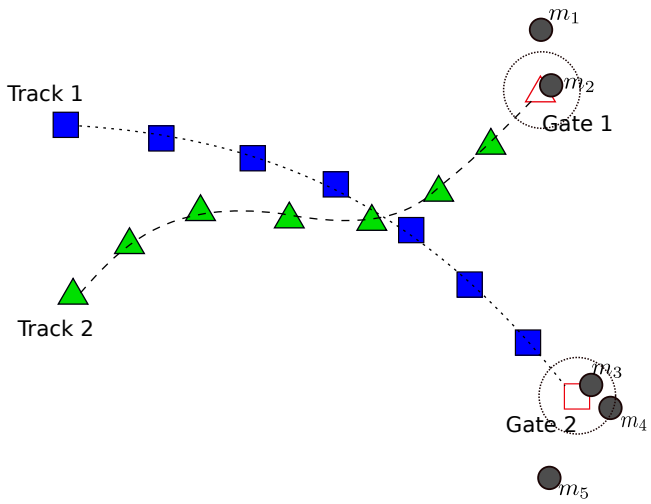


Figure 6.3.8: Illustration of gating

Implementation

We use the global nearest neighbor (GNN) method for data association, which evaluates each measurement and finds the "best" fits for the existing tracks. If no correspondences are found for some measurements, then new tracks will be initiated.

In our implementation, we firstly use gating threshold to preliminarily select measurement candidates for tracks updates. We utilize Euclidean distance as evaluation metric to calculate the distances between the predicted measurements and actual object measurements. For example, Fig. 6.3.8 shows two existing tracks, and the red triangle and red rectangle are the predictions of the two tracks respectively. The solid black dots are the current observations (actual measurements). The dotted circles represent the gates defined for preliminary matching of observations and tracks predictions. As can be seen, for track 1 there is only one observation falls into the gating area, while for track 2 there are two observations within the gating area. Thus these three observations will be considered for existing track association and update in the later steps, while the rest of the observations will directly initiate new tracks.

Algorithm 3: Data_Association($\bar{\boldsymbol{z}}_k, \boldsymbol{z}_k$)

Data: $T_{max} \leftarrow$ maximum distance threshold
Result: Lis of correspondence pair
($Index\_in\_map, Index\_in\_measurement$)

$\boldsymbol{M}_d$ = generateDistanceMap($\bar{z}_k^i, z_k^j$);
$\boldsymbol{M}_{d1}$ = sortDistanceMap($\boldsymbol{M}_d$);
$\boldsymbol{M}_c$ = generatePreliminaryCorrespondenceMapWithGating($\boldsymbol{M}_{d1}$);
$[\boldsymbol{M}_{c1}, \boldsymbol{M}_r]$ = removeRedundantMeasurementAssociation($\boldsymbol{M}_c$);
$\boldsymbol{M}_{c2}$ = recoverRedundantMeasurements($\boldsymbol{M}_{c1}, \boldsymbol{M}_r$);
$\boldsymbol{M}_{c3}$ = applyDistanceThreshold($\boldsymbol{M}_{c2}, T_{max}$);

As presented in Algorithm 3, we firstly calculate the distances between each measurement and each map element. Then, for each measurement, we sort its distances with map elements in an ascending order. Thirdly, for each measurement, a map element with minimum distance to it is extracted. With these pairing information, a preliminary correspondence map $\boldsymbol{M}_c$ is generated. It is possible that in this resulted correspondence map one map element is associated with multiple measurements.Therefore, we need to remove these redundant associations from the correspondence map. So fourthly, for each map element with associations, we compare distances of each associated measurement to it, and only keep the measurement with minimum distance in the updated correspondence map $\boldsymbol{M}_{c1}$, while the redundant measurements are removed and stored in $\boldsymbol{M}_r$. Fifthly, for each of the measurements in $\boldsymbol{M}_r$, we check if the map element with the second minimum distance to it has been associated with any other measurements. If not, then we pair them up, and add this association to the correspondence map, thus updating $\boldsymbol{M}_{c1}$ to $\boldsymbol{M}_{c2}$. Lastly, a predefined distance threshold T_{max} is applied to the correspondence map $\boldsymbol{M}_{c2}$ to remove the correspondences with distances greater than the threshold. The final correspondence map is stored in $\boldsymbol{M}_{c3}$.

6.3.9 Classification and Map Management

Map management includes two major functionalities, namely, initiation of new map element and deletion of "inactive" map element . Each detected object in the environment corresponds to a map element in the dynamic map.

After data association step, there might be some measurements that are not associated with any map element. These unassociated measurements are then perceived as new objects in the environment, so they will be added into the dynamic map and tracked by initiated Kalman Filters.

We consider a map element active if it is frequently updated with measurements, that is, the robot's confidence about the state of this map element is high. We use eigenvalues of the covariance matrix of the object to estimate the confidence of the state.

Among all eigenvalues of the state covariance matrix, the maximum one represents the largest spread of data, indicated by value λ, in a certain direction. We use value g_k^i, as defined in Eq. 6.35, to estimate the confidence of the state. If it is above a predefined threshold, then we think that the robot is reasonably confident about the current state of the map element, otherwise, we think that the confidence of the map element is not sufficient to continue to keep it in the map.

$$\boldsymbol{\Sigma}_k^i \boldsymbol{v} = \lambda \boldsymbol{v} \tag{6.34}$$

$$g_k^i = \frac{1}{max(\lambda_1 \dots \lambda_n)} \tag{6.35}$$

Additionally, Our algorithm offers classification capabilities which classifies detected objects into two categories: static and dynamic. Once an object is classified as static, then its velocity is set to zero and not any more estimated by KF, in other words, only the position and shape size are still estimated by KF. While if it is classified as dynamic, then its velocity remains to be estimated by

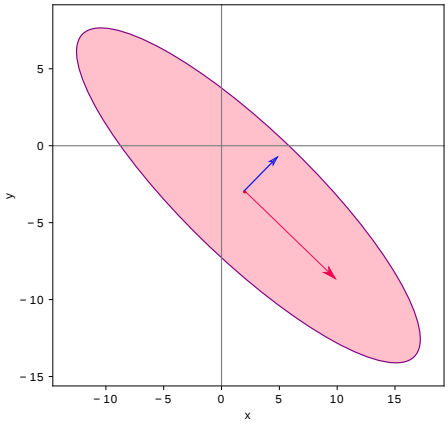


Figure 6.3.9: Confidence ellipse

KF.

Since it usually takes some time for KF to converge to a reasonable state estimate, we only start classification and checking the ”activeness” of objects after an observation period.

6.3.10 Experiments and Results

Experiment Setup

Our lab is equipped with an OptiTrack system, which we use in our experiment to obtain pose information of the robot, and as ground truth the positions of the static and dynamic objects in the robot's environment.

LIDAR Hokuyo UTM-30LX is mounted horizontally on the robot. Due to the limited space of our lab, the detection scope is limited to a rectangular space with size 1.9m x 2.9m. This experimental arena is surrounded by a wall made of nets from all sides, as shown in Fig. 6.3.10. LIDAR runs at a rate of 20 Hz during detection.

The algorithm illustrated in this paper is implemented with C++, and can run fully online on a low-performance computing unit which is DJI Manifold.

The experimental scenario is as follows: Two vertically-piled barrels as static objects, one person moving around as dynamic object. The robot starts to fly after the person enters the experimental arena.

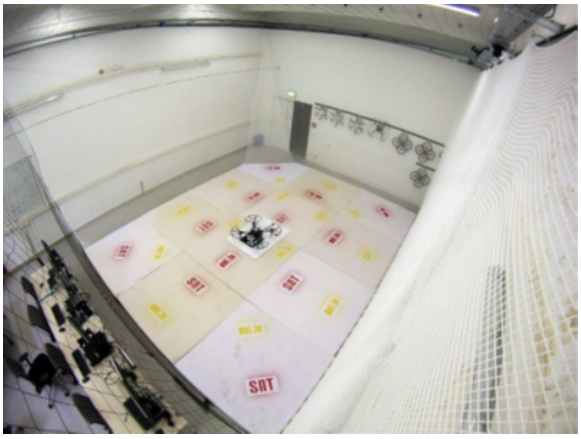


Figure 6.3.10: Lab setup

Results Discussion

Fig. 6.3.11 mainly illustrates the spatial evolution of detected objects in the robot's environment over a period of time. Axes X and Y provide the 2D spatial information in meters. Additionally, the initial position of the UAV is added to the figure, which is presented by the red dotted rectangle on the lower right part of the figure. A red solid line fan shape is placed over the rectangle to illustrate the FOV of the robot at the initial position.

In this figure, color green is used to represent static objects with IDs of 0, 1 and 2. Other colors (except for red) are used to represent the detected dynamic objects, and circle trail in the same color is the footages of same object over time, and their centers are connected with directed lines of the same color to show the moving

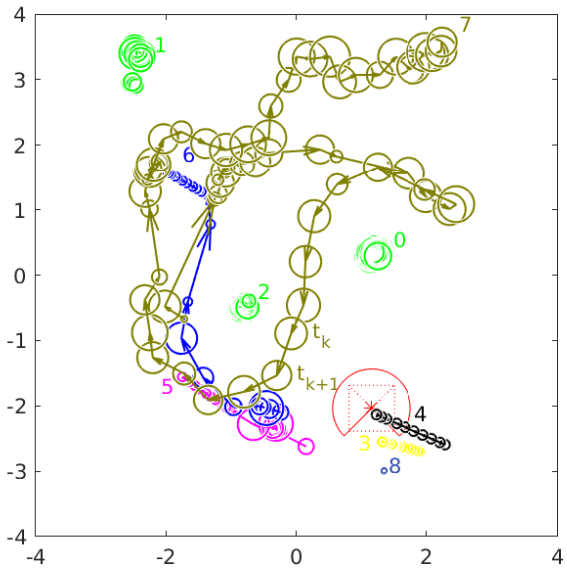


Figure 6.3.11: Experimental result: Tracks

direction of the object. Besides, the ID of the object is also written in the same color.

This figure is a result of following four experimental stages (below we will directly use ID number to differentiate objects):

First stage : Initially, the robot sits on the ground and only detects three objects in the environment: 0, 1 and 2, as shown in green. After an observation time, our algorithm classifies these objects as static. Detection of static object with ID of 0 and 2 are expected as experimental design. While detected object with ID of 1 appears because part of the net-wall intrudes the experimental arena.

Second stage: A person walks into the experimental arena (from the lower right corner in the 2D plot), and her legs are detected by the robot for a very short time, so 3 and 4 are initiated in the map. Then the person walks through the blind zone of the robot, where the robot can not "see". Due to the fact that Kalman filter initially needs some iterations with measurements in order to come to a reasonable estimate of the state, thus 3 and 4 drift and are later deleted from

the map due to their low state confidence.

Third stage: The person reappears in the robot's detection scope, therefore 5 and 6 are initiated for tracking the legs. As the person continues, one of the legs is occluded by another, thus 5 drifts and then deleted from the map, while 6 remains. Later 7 is initiated as the occluded leg is again detected by the robot. Meanwhile the robot starts flying. 6 and 7 are occluded by 2. Despite the occlusion, they remain being tracked after reappearing in the detection scope. Since the robot comes to a height of 1 meter from the ground, then the person is detected as a single object, then 7 survived and 6 drifts and is later deleted.

Fourth stage: The robot flies around the arena, however, tracking of the person has become relatively stable, solely represented by the golden circles. The person continues to walk to the right side of 0 and then goes down passing between 0 and 2, and subsequently walks up from the left side of 2 and turns to the right side passing 1. During the whole process, despite some occlusions from time to time, the person is well tracked with 7. Additionally, 8 represents a slight catch of the net-wall during detection, and is deleted from the map shortly after.

In order to evaluate the effectiveness of our detection and tracking algorithm, we need the ground truth of the objects' positions in the arena. For the static objects, it is relatively easy to obtain. However, it is a difficult task to obtain accurate ground truth for the position of a moving person. In our attempt, we let the moving person wear a hat which is tracked by the OptiTrack system. We observed that the head position sometimes does not accurately reflect the real ground truth of a person's position especially during motion, thus in this work we focus on qualitative analysis of the results. Detailed numerical analysis will be provided in the future work where more accurate ground truth can be extracted. Nevertheless, the head position, if no other option in mind, does provide a reasonable estimate. In the Fig. 6.3.12, we only take partial results data for simplicity and better illustration. The black diamonds represent the sampled ground truth positions. For 0 and 2, they only have one black diamond each. For 7, it has a trail of black diamonds which represent the sampled locations of the

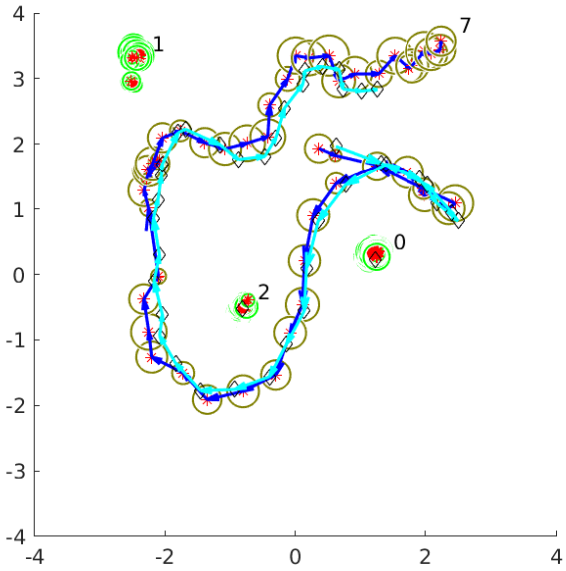


Figure 6.3.12: Experimental result: Ground truth

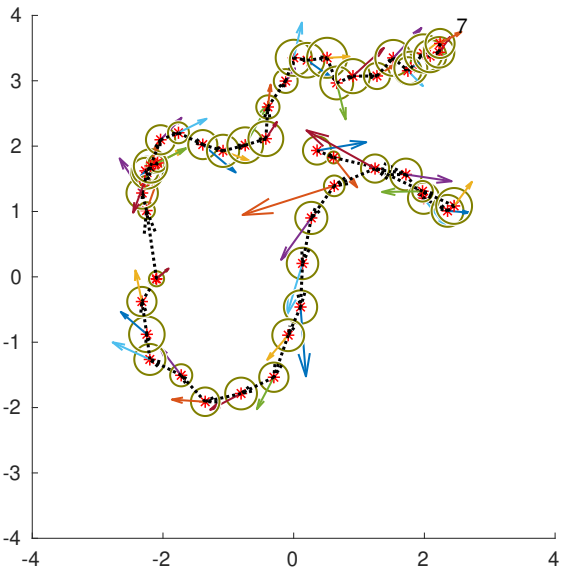


Figure 6.3.13: Experimental result: Velocity estimation

person's head. The red asterisks denote the centers of the circles. Cyan colored line represents the "ground truth" of the moving person's trajectory, while the blue line represents the KF-estimated trajectory. Their arrow directions show the moving direction of the person. We notice that the ground truth line (cyan) stops before the person tracking line (blue) stops, i.e. cyan line is shorter than blue lines, it is due to the fact that there is a blind spot for our OptiTrack system, which lies at the upper right corner in the figure. Additionally, though 1 is also classified as static object, it does not have ground truth, because it is not fixed as other static objects.

For future use of improving obstacle avoidance performance, our algorithm also offers the capability to estimate object velocities. As indicated in Fig. 6.3.13, each identified dynamic object is associated with an estimated velocity vector, which is represented by the vector pointing out from the center of the circle. The velocity vector portrays the magnitude and direction of the velocity.

6.4 Summary

In this chapter we present a practical 2D LIDAR-based object detection and tracking approach for the purpose of collision-free navigation, in which objects in the environment are modeled in a compact form as cylinders. The results suggest that the proposed system is capable of classifying and tracking objects with reasonable accuracy, as well as estimating velocities of the dynamic objects. This approach represents objects in the environment with a compact form, therefore it is memory-efficient, and the amount of data need to be processed from 2D LIDAR is relatively few especially in comparison to cameras and 3D LIDAR, therefore this approach is not computationally expensive and can be run online, which thus enables the robot to be more responsive to the changes in the environment. The proposed model fitting method is robust, because it approximates the detected object into cylinder shape regardless of the underlying shape of the object in concern, while also keeping the fitted models good for collision-free navigation.

Success is not final; failure is not fatal: It is the courage to continue that counts.

Winston S. Churchill

7

Optimization-based Avoidance with Nonlinear FF Control

7.1 Nonlinear Forward-facing Flight Controller

Our UAV flight controller is designed in such a way that it takes into account the characteristics of the sensing unit, and thus supports effective sensing. We employ a 2D LIDAR (i.e. Hokuyo UTM-30LX) in our sense-and-avoid solution, which provides a 270° horizontal field of view (FOV). It is assumed that during the flight the UAV maintains a near-hovering state, which ensures that the LIDAR can effectively sense the surrounding of the UAV. Though quadrotor UAV is capable of flying in the x^b and y^b direction without yawing, this way of flying is not desirable for our solution as when the UAV flies sideways, the FOV of the sensing unit does not provide full coverage of the sides.

Therefore, in this section we propose a nonlinear vehicle control system which realizes a forward-facing flight behavior in the x-y-plane, while maintaining a desired altitude in z-direction. It means that the front arm of the quadrotor as defined in Fig. 4.3.1 must always point in the direction of the desired velocity vector in the x-y-plane in order to be able to always sense in the flight direction. The flight control consists of two main components, an inner attitude control loop that tries to achieve a desired orientation as fast as possible and an outer velocity control loop that realizes the desired velocity vector as well as the forward-facing flight.

Attitude Control System

The attitude control system is based on our previous work [61]. The attitude control considers the last six ordinary differential equations (ODEs) of Eq. (4.21), which describe the dynamics of Euler angles and angular rates of the quadrotor. Euler angles can be obtained by pure integration. For the submodel formed by the last three ODEs, we firstly simplify the model by neglecting the gyroscopic terms as follows:

$$\begin{bmatrix} \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} x_{11}x_{12}I_1 + \frac{L}{I_x}u_2 \\ x_{10}x_{12}I_2 + \frac{L}{I_y}u_3 \\ x_{10}x_{11}I_3 + \frac{1}{I_z}u_4 \end{bmatrix} \tag{7.1}$$

and then apply feedback linearization technique in order to obtain a linear system:

$$\begin{aligned} u_2 &= f_2(x_{10}, x_{11}, x_{12}) + u_2^* \\ u_3 &= f_3(x_{10}, x_{11}, x_{12}) + u_3^* \\ u_4 &= f_4(x_{10}, x_{11}, x_{12}) + u_4^* \end{aligned} \tag{7.2}$$

where u_2^*, u_3^*, u_4^* are introduced new input variables. In order to obtain a linear system, the following conditions must be fulfilled:

$$\begin{aligned} x_{11}x_{12}I_1 + \frac{L}{I_x}f_2(x_{10}, x_{11}, x_{12}) &= K_2x_{10} \\ x_{10}x_{12}I_2 + \frac{L}{I_y}f_3(x_{10}, x_{11}, x_{12}) &= K_3x_{11} \\ x_{10}x_{11}I_3 + \frac{1}{I_z}f_4(x_{10}, x_{11}, x_{12}) &= K_4x_{12} \end{aligned} \tag{7.3}$$

with K_2, K_3 and K_4 being constant parameters with negative real values. From Eq. 7.3 we can get

$$\begin{aligned} f_2(x_{10}, x_{11}, x_{12}) &= \frac{I_x}{L}(K_2x_{10} - x_{11}x_{12}I_1) \\ f_3(x_{10}, x_{11}, x_{12}) &= \frac{I_y}{L}(K_3x_{11} - x_{10}x_{12}I_2) \\ f_4(x_{10}, x_{11}, x_{12}) &= I_z(K_4x_{12} - x_{10}x_{11}I_3) \end{aligned} \tag{7.4}$$

With Eq. (7.2) and Eq. (7.4), Eq. (7.1) can be transferred into a set of linear and decoupled ODEs. Details about the dynamics and stability of this subcontrol system can be found in [61]. If we define $u_2^* = w_2(x_{7d} - x_7)$, $u_3^* = w_3(x_{8d} - x_8)$ and $u_4^* = w_4(x_{9d} - x_9)$, where x_{7d}, x_{8d} and x_{9d} are desired Euler angles generated from velocity control system, together with the current state vector of the UAV, we can ultimately obtain the overall control input for UAV u_2, u_3 and u_4.

Velocity Control System

The command to the vehicle control system is from the human pilot, a desired velocity vector in the x-y-plane given by $v_{xd} = x_{4d}$, $v_{yd} = x_{5d}$, and a desired altitude $z_d = x_{3d}$ in z-direction.

In order to obtain the desired Euler angles for the inner control loop and the UAV overall control input u_1, we consider following three equations from Eq.

(4.23):

$$\begin{aligned}
\dot{x}_4 &= -(\cos x_{7d} \sin x_{8d} \cos x_{9d} + \sin x_{7d} \sin x_{9d}) \frac{u_1}{m} \\
\dot{x}_5 &= -(\cos x_{7d} \sin x_{8d} \sin x_{9d} - \sin x_{7d} \cos x_{9d}) \frac{u_1}{m} \\
\dot{x}_6 &= g - (\cos x_{7d} \cos x_{8d}) \frac{u_1}{m}
\end{aligned} \tag{7.5}$$

Substitute the variables as follows

$$\begin{bmatrix} \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = f(x_{7d}, x_{8d}, x_{9d}, u_1) = \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{bmatrix} \tag{7.6}$$

where

$$\begin{aligned}
\tilde{u}_1 &= k_1(x_{4d} - x_4) \\
\tilde{u}_2 &= k_2(x_{5d} - x_5) \\
\tilde{u}_3 &= k_3(x_{3d} - x_3) - k_4 x_6
\end{aligned} \tag{7.7}$$

With this substitution, we essentially form a closed loop control of the UAV velocities, and apply P-controllers to *x*-*y*-plane velocities, and a PD-controller for altitude control. From Eq. (7.5) - (7.7) we can then obtain x_{7d}, x_{8d}, x_{9d} and u_1. Since we would like to maintain a forward-facing flight, the desired yaw angle is required to be as

$$x_{9d} = \arctan\left(\frac{x_{5d}}{x_{4d}}\right) \tag{7.8}$$

As we assume that the UAV flies near hovering state, the desired roll and pitch angles are then sufficiently small, which allows small angle approximation:

$$\begin{aligned}
\sin x_{7d} &\approx x_{7d}, \ \cos x_{7d} \approx 1 \\
\sin x_{8d} &\approx x_{8d}, \ \cos x_{8d} \approx 1
\end{aligned} \tag{7.9}$$

If we define

$$\cos x_{9d} = \alpha, \ \sin x_{9d} = \beta \tag{7.10}$$

and then insert Eq. (7.9) and (7.10) into Eq.(7.5), it yields

$$\begin{aligned}\tilde{u}_1 &= -(\alpha x_{8d} + \beta x_{7d})u_1/m \\ \tilde{u}_2 &= -(\beta x_{8d} - \alpha x_{7d})u_1/m \\ \tilde{u}_3 &= g - u_1/m\end{aligned} \tag{7.11}$$

Using the last equation of (7.11) allows the calculation of the unknown input u_1 as

$$u_1 = m(g - \tilde{u}_1) \tag{7.12}$$

Since u_1 is obtained, the first two equations of (7.11) become linear equations that can be solved as

$$\begin{bmatrix} x_{7d} \\ x_{8d} \end{bmatrix} = \begin{bmatrix} -\frac{u_1}{m}\beta & -\frac{u_1}{m}\alpha \\ \frac{u_1}{m}\alpha & -\frac{u_1}{m}\beta \end{bmatrix}^{-1} \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} \tag{7.13}$$

7.2 Optimization-based Avoidance

7.2.1 Situation Evaluation Policy

Though the LIDAR we currently employ has only 270° planar field of view, since the UAV is forward moving, i.e. the UAV is confined to move only in the direction where the LIDAR is front facing, and the sensing module memorizes the detected surroundings for a short period of time, we can reasonably predict that there would be no object in the blind zone of LIDAR, with the assumption that there are no object actively pursuing the UAV and aiming to hit it from behind.

Since we would like to assess the situation around the UAV in order to decide if there is imminent threat and if the UAV should react to resolve the emergency, it would be a good idea to divide the area around the UAV into regions of different safety levels. Similar idea appeared in [55], in this paper we propose a modified definition of safety zone set.

The nearest area around the UAV is defined as Anti-object Zone Ω, as indicated

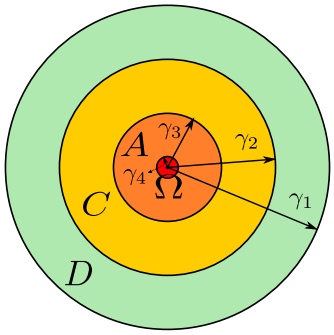


Figure 7.2.1: Defined Safety Zones around UAV for Emergency Evaluation

as red circular region in Fig. 7.2.1. This is the most safety-critical zone where no object should enter, otherwise collision with the UAV would take place.

We define an integer set $\mathbb{Z} = \{1, 2, \ldots, n\}$ with n being the number of detected objects in the environment, and define $\boldsymbol{P}_Q$ and $\boldsymbol{P}_{p_i}$ as the 2D positions of the UAV and the ith object (since in this paper we assume that the UAV keeps flight at a certain height level, we can then focus on x-y-plane), i.e. $\boldsymbol{P}_Q, \boldsymbol{P}_{p_i} \in \mathbb{R}^2$. The radius of the ith object is defined as λ_i. Anti-object Zone $\boldsymbol{\Omega}$ is then defined as follows:

$$\boldsymbol{\Omega} = \{\boldsymbol{P}_Q - \boldsymbol{P}_{p_i} - \lambda_i \leq \gamma_4,\ i \in \mathbb{Z}\} \tag{7.14}$$

where γ_4 denotes the required minimum separation distance between the center of the UAV and the periphery of an object in order to ensure safety.

The next level of safety zone is defined as Avoidance Zone, $\boldsymbol{A}$, as indicated in orange in Fig. 7.2.1. If there is object within this region, then imminent threat for the UAV is considered present, and action must be taken to avoid collision.

$$\boldsymbol{A} = \{\gamma_4 \leq \boldsymbol{P}_Q - \boldsymbol{P}_{p_i} - \lambda_i \leq \gamma_3,\ i \in \mathbb{Z}\} \tag{7.15}$$

Then we define a Conflict Zone (yellow zone in Fig. 7.2.1), $\boldsymbol{C}$, as the region in which the UAV feels the threat of the object, yet does not consider immediate action necessary until the estimated time to collision (TTC) between them is

below a predefined threshold.

$$\boldsymbol{C} = \{\gamma_3 \leq \boldsymbol{P}_Q - \boldsymbol{P}_{p_i} - \lambda_i \leq \gamma_2,\ i \in \mathbb{Z}\} \tag{7.16}$$

Lastly, Detection Zone, $\boldsymbol{D}$, as an exterior zone around the UAV is defined, as shown in green in Fig. 7.2.1. The UAV merely feels the presence of objects, and tries to establish tracking and velocity estimation of the objects in this region, in case they come closer and violate zone of another safety-level. Beyond this zone, objects are considered non-existent to the UAV since they are beyond the detection range of the LIDAR sensor.

$$\boldsymbol{D} = \{\gamma_2 \leq \boldsymbol{P}_Q - \boldsymbol{P}_{p_i} - \lambda_i \leq \gamma_1,\ i \in \mathbb{Z}\} \tag{7.17}$$

7.2.2 Collision Avoidance

In this subsection, we introduce our proposed collision avoidance controller, which generates collision avoidance control commands for lower-level vehicle controllers. This controller design is inspired by theory from missile guidance.

Fig. 7.2.2 depicts the engagement geometry between UAV and one object (represented as a circle), where σ is the line-of-sight angle and R is the Euclidean distance between them. In a classical missile guidance problem, their relative kinematics is described by following two differential equations:

$$\dot{R} = v_{p_x} \cos\sigma + v_{p_y} \sin\sigma - v_{Q_x} \cos\sigma - v_{Q_y} \sin\sigma \tag{7.18}$$

$$\dot{\sigma} = \frac{1}{R}(v_{p_y} \cos\sigma - v_{p_x} \sin\sigma - v_{Q_y} \cos\sigma + v_{Q_x} \sin\sigma) \tag{7.19}$$

where $\boldsymbol{v}_Q = (v_{Q_x}, v_{Q_y})$ and $\boldsymbol{v}_p = (v_{p_x}, v_{p_y})$ represent 2D velocities of the UAV and object respectively.

It is well known in missile guidance [68] that two vehicles are on a collision

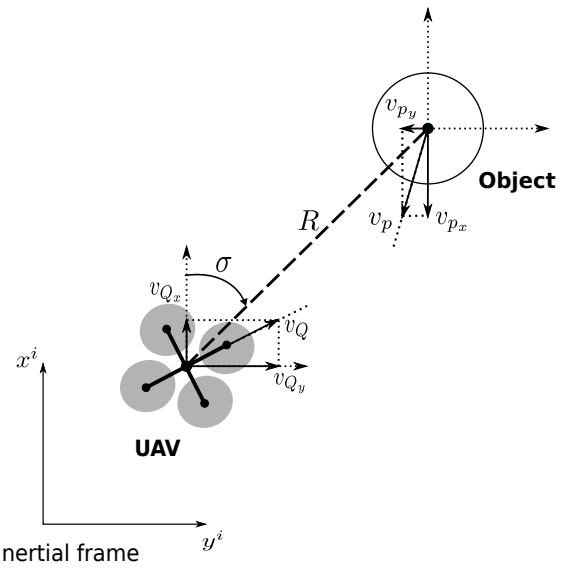


Figure 7.2.2: Engagement Geometry of UAV and an Object (top-down view)

course, if following conditions are satisfied:

$$\dot{R} < 0 \quad \text{and} \quad \dot{\sigma} = 0 \tag{7.20}$$

The intuition behind our avoidance strategy is that since we know the conditions for two entities being on a collision course, we could then control the UAV in such a way that they are constantly on a non-collision course by taking the reverse of these conditions, which are

$$\dot{R} \geq 0 \quad \text{and} \quad \dot{\sigma} \neq 0 \tag{7.21}$$

With above considerations in mind, we formulate the avoidance control

problem as follows:

$$\begin{aligned} \max_{r,\psi} \quad & \sum_{i=1}^{n} \zeta_i \dot{\sigma}_i(r,\psi) + \sum_{i=1}^{n} \eta_i \dot{R}_i(r,\psi) \\ \text{s.t.} \quad & 0 \leq r \leq r_{max} \\ & -\frac{\pi}{2} + \psi_0 \leq \psi \leq \frac{\pi}{2} + \psi_0 \\ & \zeta_i > 0, \eta_i > 0 \quad \forall i \in \mathbb{Z} \end{aligned} \tag{7.22}$$

where

$$\begin{aligned} r &= \boldsymbol{v}_Q \\ v_{Q_x} &= r\cos\psi \\ v_{Q_y} &= r\sin\psi \end{aligned} \tag{7.23}$$

and $\boldsymbol{v}_Q = (v_{Q_x}, v_{Q_y})$ represent 2D velocity of the UAV, ψ_0 is the initial yaw angle of the UAV, r_{max} is the magnitude of maximum desirable velocity of the UAV (it is set to $1.5m/s$ in the simulation).

In this formulation, we have chosen r and ψ as decision variables instead of v_{Q_x}, v_{Q_y} because in this way we simplify the optimization problem by avoiding nonlinear constraints.

7.2.3 Experiments and Results

We conducted simulation with proposed vehicle controller, emergency evaluation policy and avoidance strategy using Matlab/Simulink. The simulation scenario from top-down view is as follows: The UAV flies towards east from location $(0,0)m$ with initial velocity $(1,0)m/s$, while two objects fly towards the UAV with constant velocities: Object 1 (noted as o_1) from position $(30,-10)m$ with velocity $(-3,1)m/s$, object 2 (noted as o_2) from position $(30,30)m$ with velocity $(-2,-1)m/s$. The UAV has an anti-object zone with radius $\gamma_4 = 1.5m$, while object 1 and object 2 have minimum safety region with radii $\lambda_1 = 1m$ and $\lambda_2 = 2m$ respectively, all of which are indicated as red circles in the Fig. 7.2.3. We

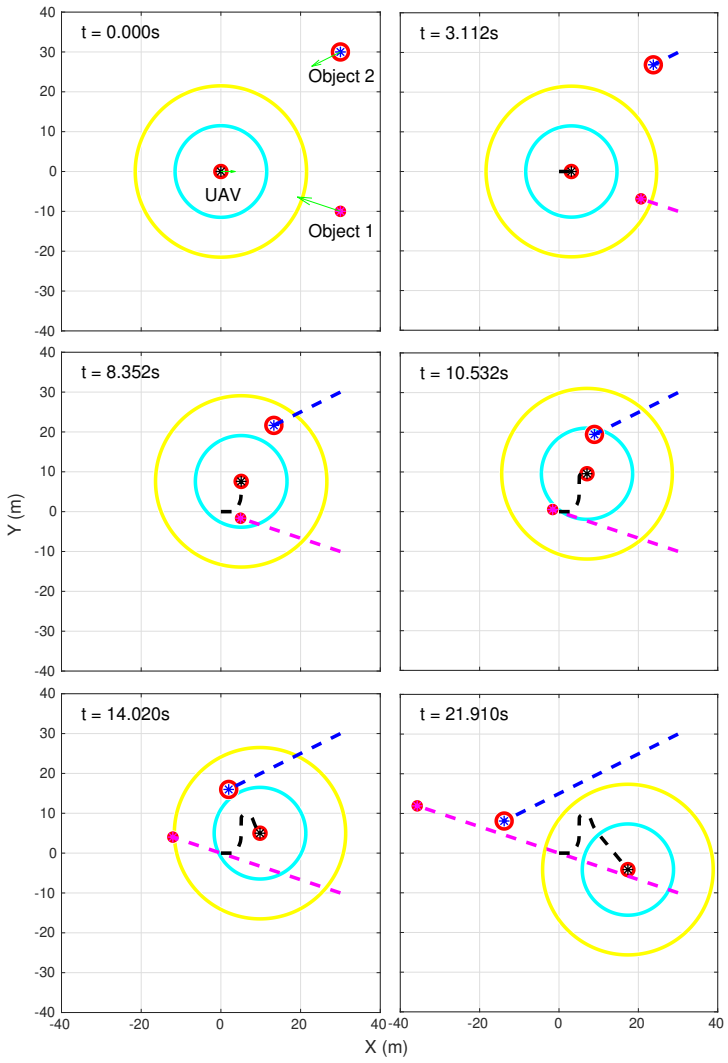


Figure 7.2.3: Simulated Flight from top-down view: thick dashed lines represent trajectories (black for UAV, magenta for object 1, blue for object 2), yellow and cyan circles around UAV visualize the exterior borders of conflict zone and avoidance zone, red circles around the UAV and both objects visualize the minimum safety zone

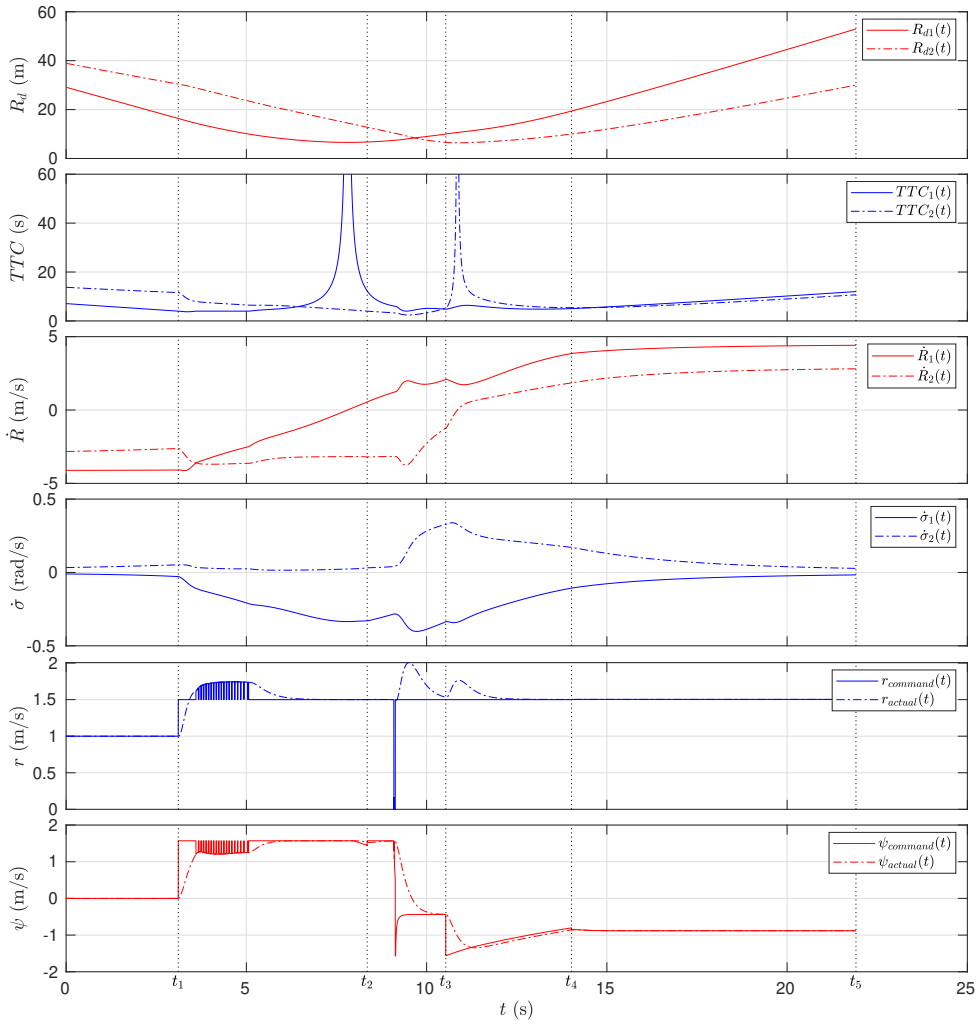


Figure 7.2.4: Simulation Result Analysis: $t_1 = 3.112s$, $t_2 = 8.362s$, $t_3 = 10.532s$, $t_4 = 14.020s$, $t_5 = 21.910s$

also set $\gamma_3 = 10m + \gamma_4$, $\gamma_2 = 20m + \gamma_4$ and $\gamma_1 = 30m$. Several salient moments of the simulated flight are captured, as shown in Fig. 7.2.3.

In subfigure at time $t = 0.000s$, the green arrows represent initial velocities, the sizes of the arrows are in proportion to the magnitude of the velocities. The yellow and cyan circles around UAV visualize exterior borders of the conflict zone and avoidance zone respectively. Initially both objects are outside the conflict zone of the UAV. Subfigure at time $t = 3.112s$ depicts the moment that o_1 is in the conflict zone of the UAV, and the UAV starts to deem o_1 as imminent threat as the their estimated time to collision TTC_1 now goes just below the predefined threshold (the threshold for TTC is set to $4s$ in this simulation). Emergency avoidance behavior of the UAV is thus triggered. At time $t = 8.352s$, both objects pose imminent threat to the UAV, and thus influence its avoidance maneuver, as o_1 is now in zone $\boldsymbol{A}$ of the UAV while o_2 is in zone $\boldsymbol{C}$ and TTC_2 is just below the predefined threshold. At time $t = 10.532s$, o_1 leaves zone $\boldsymbol{A}$ of the UAV, while o_2 is within the zone $\boldsymbol{A}$. Since TTC_1 is above threshold, at this point in time only o_2 is considered as imminent threat and causes the UAV to take avoidance action. At time $t = 14.020s$, both objects stop posing threat to the UAV, as o_1 is in the conflict zone, and o_2 just goes out of the avoidance zone, and both TTC_1 and TTC_2 are above the threshold. From this moment on, the UAV stops taking avoidance action, and continues to fly in current velocity. At time $t = 21.910$, both objects are outside zone $\boldsymbol{D}$ of the UAV, which means that at this moment the UAV no longer detects any object, it can now gradually come to a stop or continue flight depending on the predefined preference of the human pilot while signaling to the pilot that the control of the UAV is now returned.

In order to obtain a more qualitative view of the simulated flight, we present following relevant quantities over time in Fig. 7.2.4. Particularly, in order to take into account sizes of the UAV and the objects instead of considering them as point mass, we use R_{di} instead of R_i to represent the distance between an object and the UAV, where $R_{di} = R_i - \gamma_4 - \lambda_i,\ i \in \mathbb{Z}$. From the last two subfigures of Fig. 7.2.4 we can see that with our proposed vehicle controller the UAV is in general following commands quite well. We notice that in the $r(t)$ subfigure, despite that we set the maximum desirable velocity of the UAV to $1.5m/s$, the velocity still temporarily

goes up to $2m/s$. We also notice that this velocity overshoot happens particularly when the UAV is turning (yawing). It is mainly caused by the undesired rolling while yawing, since they are coupled in our UAV model, which also reflects real-world situation. From the first two subfigures of Fig. 7.2.4, we can understand why and when the avoidance behavior of UAV is invoked from a qualitative perspective. The two subfigures in the middle visualize the evolution of $\dot{R}$ and $\dot{\sigma}$ over time, and how they are influenced by the avoidance behavior of UAV during certain period of time.

After evaluation of the overall simulation results, we can see that with the proposed vehicle controller, emergency evaluation policy and collision avoidance strategy, the UAV is capable of avoiding multiple moving objects at the same time.

7.3 Summary

In this chapter we propose a novel nonlinear vehicle control system adapted to the characteristics of LIDAR sensing, a heuristic emergency evaluation policy and a novel avoidance control strategy which takes inspiration from missile guidance. The simulation results suggest that the proposed sense-and-avoid solution is capable of handling complex environments where multiple moving objects are present, and thus ensure safe flight of the UAV.

Your time is limited, so don't waste it living someone else's life.
Don't be trapped by dogma – which is living with the results
of other people's thinking.

Steve Jobs

8

AVCFF-based Avoidance with PID-based FF control

8.1 PID-based Flight Controller

The controller is implemented as a set of cascaded PID controllers, with the inner loop controlling the attitude, yaw rate and vertical velocity and an outer loop controlling the horizontal velocity, heading and altitude. This approach assumes that each axis and the altitude can be controlled independently, which is valid for moderate deviations from the hovering state. The output of the inner loop are commanded torques and vertical thrust, which are translated to motor voltages either by using a static mixture matrix or by feeding them into an inverted model of the propulsion system. [40]

we create the forward-facing vehicle controller on the basis of the PID controller provided by the Hector Quadrotor framework. In order to have a forward-facing flight controller in the existing framework, we modify the original vehicle controller (which we call lower-level controller), and cascade a higher level controller in front of the lower-level controller. Specifically, the higher-level controller takes in desired human operator velocity control commands $v^b_{x,hd}$ and $v^b_{y,hd}$, and generates control commands $v^b_{x,d}$ and ψ^b_d for the lower level controller, as shown in Fig. 8.1.1. The lower-level controller has been modified so that now it takes in yaw angle in body frame instead of that in inertial frame. The relationship between these two levels of controllers expressed in mathematical form is as follows:

$$v^b_{x,d} = f_1(v^b_{x,hd}, v^b_{y,hd}) = \sqrt{(v^b_{x,hd})^2 + (v^b_{y,hd})^2} \tag{8.1}$$

$$\psi^b_d = f_2(v^b_{x,hd}, v^b_{y,hd}) = \arctan \frac{v^b_{y,hd}}{v^b_{x,hd}} \tag{8.2}$$

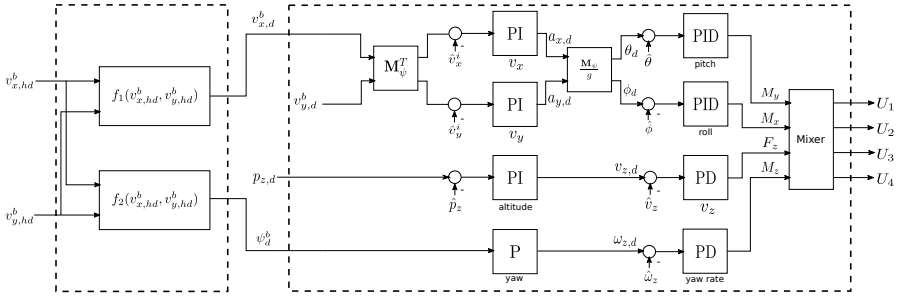


Figure 8.1.1: Quadrotor Forward-facing Flight Controller Diagram

8.2 AVCFF-based Reactive Avoidance

8.2.1 Situation Evaluation Policy

The UAV is constantly evaluating the situation of its perceived surrounding to predict potential collisions. We incorporate two progressive situation evaluation

policies to identify the level of collision threat posed by obstacles (or objects), namely, relative peripheral distance based evaluation and relative velocity based evaluation. The identified collision threat level, if any, will then be utilized later for adapting the avoidance strategy accordingly.

Evaluation based on relative distance

We adopt the safety zone set concept from [64], however, in this work the safety zone settings are modified to accomodate the challenges brought in by the high speed of UAV. The adapted safety zone set is illustrated in Fig. 8.2.1.

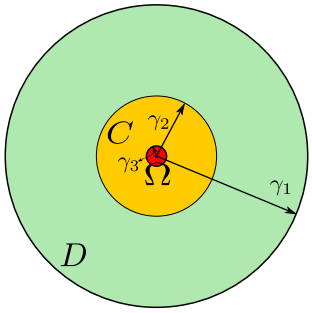


Figure 8.2.1: Safety Zones for Collision Evaluation

For the purpose of illustration, some notations need to be defined. We define an integer set $\mathbb{Z} = \{1, 2, \ldots, n\}$ where n is the number of detected objects in the environment. $\boldsymbol{p}_u$ and $\boldsymbol{p}_{o_j}$ represent the x-y positions of the UAV and the jth object, i.e. $\boldsymbol{p}_u, \boldsymbol{p}_{o_j} \in \mathbb{R}^2$. We assume that the radius of detected object o_j is noted as λ_j. Additionally, d_{c_j} is defined as the distance between centers of the UAV and o_j, then its mathematical form is as follows:

$$d_{c_j} = \boldsymbol{p}_u - \boldsymbol{p}_{o_j} \tag{8.3}$$

The nearest area around the UAV is defined as Anti-object Zone $\boldsymbol{\Omega}$, indicated as red circular region in Fig. 8.2.1. This is the most safety-critical zone where no object should enter under any circumstance, because collision with the UAV would take place once the zone is intruded. Anti-object Zone $\boldsymbol{\Omega}$ is defined as

follows:

$$\boldsymbol{\Omega} = \{d_{c_j} \leq \gamma_3 + \lambda_j,\ j \in \mathbb{Z}\} \tag{8.4}$$

The next level of safety zone is defined as Conflict Zone, $\boldsymbol{C}$, as indicated in yellow in Fig. 8.2.1. Any object within this region of the UAV is considered as a potential collision threat and will trigger UAV's avoidance action.

$$\boldsymbol{C} = \{\gamma_3 + \lambda_j \leq d_{c_j} \leq \gamma_2 + \lambda_j,\ j \in \mathbb{Z}\} \tag{8.5}$$

Lastly, Detection Zone $\boldsymbol{D}$ as an exterior zone around the UAV is defined, as shown in green in Fig. 8.2.1. The UAV is merely observing the objects, and tries to establish tracking of objects and estimation of their velocities in this region for later use in case the objects come closer and violate another safety-level zone. Beyond zone $\boldsymbol{D}$, objects are considered non-existent to the UAV since they are beyond the detection range of the LIDAR sensor.

$$\boldsymbol{D} = \{\gamma_2 + \lambda_j \leq d_{c_j} \leq \gamma_1 + \lambda_j,\ j \in \mathbb{Z}\} \tag{8.6}$$

Evaluation based on relative velocity

If an object intrudes zone $\boldsymbol{C}$ of the UAV, then a further evaluation on the collision threat level is carried out, which is evaluating their relative velocity. Since we assume that the pose and velocity of the UAV are known, with the estimated velocity of the detected object from sensing module, we can calculate the estimated relative velocity between them. If their relative velocity is above a predefined velocity threshold V_{thres}, then we consider a higher level of threat is present, the collision avoidance functional module will generate a more aggressive avoidance action to ensure timely collision avoidance.

8.2.2 Avoidance Strategy

This subsection presents a novel adaptive virtual cushion force field (AVCFF) based control approach which generates avoidance maneuver commands to the UAV while preserving operator's intent during the course of collision avoidance.

Adaptive Virtual Cushion Force Field

The proposed AVCFF method utilizes dynamic obstacle map generated from our sensing functional module where each obstacle is represented as a cylinder (viewed as circle on x-y-plane), as well as evaluated threat level from the situation evaluation functional module. The core idea of AVCFF method is that when the UAV encounters an obstacle intruding its conflict zone $\boldsymbol{C}$, as a collision avoidance strategy the UAV reacts as if it carries a circular safety cushion around it, that is, bouncing away from the obstacle due to the virtual repulsive force exerted on it, as illustrated in Fig. 8.2.2. More specifically, this virtual safety cushion is associated with an adaptive virtual force field function (which is detailed later in this section), which has zero influence on the UAV when the virtual cushion is not violated. However, when there are intrusions, it generates virtual forces to be exerted on the UAV in order to recover the virtual cushion from the deformation caused by obstacles. The UAV is then steered aways from the obstacles under the influence of the active virtual force field.

In traditional virtual force field method, such as potential field based collision avoidance method [10] [28][30], where virtual force functions (i.e. potential functions) are constructed around the obstacles and goals, the robot is constantly under the influence of the combined virtual force field, and guided by it towards the goal.

The novelty of our method is that contrary to the traditional method, we construct the virtual force function around the robot (in our case, UAV) instead. Additionally, the virtual force generated by the virtual force function is adaptive

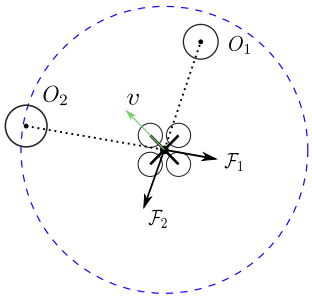


Figure 8.2.2: Illustration of Adaptive Virtual Cushion Force Field (AVCFF) Collision Avoidance Method from Top-down View: o_1 and o_2 are two objects intruding the safety cushion of the UAV (i.e. zone $\boldsymbol{C}$), $\boldsymbol{\mathcal{F}}_1$ and $\boldsymbol{\mathcal{F}}_2$ are the resulting virtual forces due to the intrusion (sizes are not in proportion). The blue spreading shade provides an intuitive impression of the virtual force field function which determines the magnitude of virtual force field

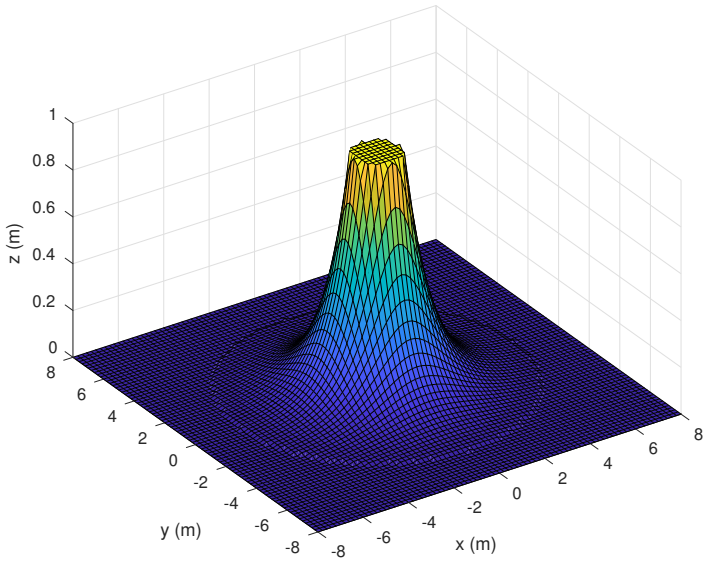


Figure 8.2.3: Visualization of Virtual Force Field Function ($\mathcal{F}_{max} = 1$, $\mathcal{D}_{thres} = 6$, $d_{m_1} = \mathcal{D}_{min} = 1$)

based on the relative peripheral distance and relative velocity between the UAV and obstacle. The advantage of the proposed AVCFF collision avoidance method is that it is simpler in calculation and more time efficient as we do not need to calculate the gradient of potential functions to find the direction and magnitude of the virtual force. Additionally it is more suitable for real-world dynamic environment, as in a real-world scenario sensor noise would cause variation in the estimated cylindrical shapes of the obstacles, and would further results in constant changing of the potential functions around the obstacles, which is not desirable as it is time consuming and not necessary.

The virtual force vector $\boldsymbol{\mathcal{F}}_j$ caused by o_j is defined as

$$\boldsymbol{\mathcal{F}}_j = \mathcal{F}_j \cdot \boldsymbol{n}_j \tag{8.7}$$

where $\mathcal{F}_j$ is the magnitude of the virtual force, and $\boldsymbol{n}_j$ is its unit direction vector. The direction of the repulsive force $\boldsymbol{n}_j$ is defined as the direction pointing from the obstacle to the UAV, as indicated by $\boldsymbol{\mathcal{F}}_1$ and $\boldsymbol{\mathcal{F}}_2$ in Fig. 8.2.2.

The adaptive virtual force function determines the magnitude of the virtual force, which is described in mathematical form as

$$\begin{aligned} \mathcal{F}_j &= f(d_{p_j}, d_{m_j}) \\ &= \begin{cases} \mathcal{F}_{max} \cdot e^{d_{m_j} - d_{p_j}} & \text{for } d_{p_j} \leq \mathcal{D}_{thres} \text{ and } \mathcal{F}_j \leq \mathcal{F}_{max} \\ \mathcal{F}_{max} & \text{for } d_{p_j} \leq \mathcal{D}_{thres} \text{ and } \mathcal{F}_j > \mathcal{F}_{max} \\ 0 & \text{for } d_{p_j} > \mathcal{D}_{thres} \end{cases} \end{aligned} \tag{8.8}$$

where $\mathcal{F}_{max}$ is a predefined maximum magnitude for any virtual repulsive force. $\mathcal{D}_{thres}$ is a constant representing the non-conflict peripheral distance threshold, and is defined by following equation:

$$\mathcal{D}_{thres} = \gamma_2 - \gamma_3 \tag{8.9}$$

d_{p_j} is the peripheral distance between the UAV and object o_j and is defined as

$$d_{p_j} = d_{c_j} - \lambda_j - \gamma_3 \tag{8.10}$$

Additionally, d_{m_j} defines the minimum peripheral distance at which the repulsive force reaches its maximum magnitude $\mathcal{F}_{max}$. The value of d_{m_j} is a function of the relative velocity between the UAV and the object o_j, as described below:

$$d_{m_j} = f(\hat{v}_{r_j}) = \begin{cases} \mathcal{D}_{thres} & \text{for } \hat{v}_{r_j} > \mathcal{D}_{thres} \\ \hat{v}_{r_j} & \text{for } V_{thres} \leq \hat{v}_{r_j} \leq \mathcal{D}_{thres} \\ \mathcal{D}_{min} & \text{for } others \end{cases} \tag{8.11}$$

where $\hat{v}_{r_j}$ represents the estimated relative velocity and V_{thres} is its predefined velocity threshold. $\mathcal{D}_{min}$ is a predefined constant, and is the absolute minimum peripheral distance that we would like to keep between the UAV and the object o_j during the entire course of avoiding this object, when the estimated relative velocity is below V_{thres}. In this case, the maximum repulsive force is reached when their peripheral distance is at $\mathcal{D}_{min}$. When the estimated relative velocity is bigger than V_{thres}, the function for generating the virtual force is adapted so as to drive the UAV to turn away faster from the obstacle. The value of d_{m_j} is reasonably capped by $\mathcal{D}_{thres}$.

Finally, the virtual forces caused by a total number of n objects are aggregated to form a total repulsive force:

$$\boldsymbol{\mathcal{F}}_{total} = \sum_{j=1}^{n} \boldsymbol{\mathcal{F}}_j \tag{8.12}$$

Incorporation of Operator Intent

The final collision avoidance control command for the UAV, $\boldsymbol{v}^b_{avoid}$, takes into account both the generated repulsive force for collision avoidance and the pilot's operation intent.

The moment when the situation evaluation functional module identifies that potential collision threat exists and triggers generation of virtual force field, the most recently received control command from the human operator is memorized and converted into world frame, which is utilized during the course of avoidance as the intention velocity control command, noted as $\boldsymbol{v}^{w}_{intent}$.

The total repulsive force from Eq. (8.12) is translated into repulsive force velocity control command, $\boldsymbol{v}^{w}_{rpl}$. Their mathematical relationship is as follows:

$$\boldsymbol{v}^{w}_{rpl} = f(\boldsymbol{\mathcal{F}}_{total}) = \boldsymbol{\mathcal{F}}_{total} \tag{8.13}$$

With $\boldsymbol{v}^{w}_{intent}$ and $\boldsymbol{v}^{w}_{rpl}$, we can obtain $\boldsymbol{v}^{b}_{avoid}$ with following equations:

$$\boldsymbol{v}^{b}_{avoid} = \boldsymbol{R}^{b}_{w}(\boldsymbol{v}^{w}_{intent} + \boldsymbol{v}^{w}_{rpl}) \tag{8.14}$$

Finally, the direction of $\boldsymbol{v}^{b}_{avoid}$ will be used to guide the UAV's velocity, while the magnitude of the velocity remains constant. This combination is then used as control input for the controller displayed in Fig. 8.1.1, completely overriding the pilot's new control input during the course of collision avoidance. The course of collision avoidance is considered complete when the UAV detects no objects in its zone $\boldsymbol{D}$ and flies at the velocity of the operator's intent, $\boldsymbol{v}^{w}_{intent}$. We assume that at this point in time the UAV would send a signal to the remote control to trigger it to slightly vibrate/or make a sound or light so as to warn the human operator that the control over the UAV will be returned.

8.2.3 Experiments and Results

We conducted simulation experiments using ROS [51] and Gazebo [29] simulator. Since the simulated UAV model provided by Hector Quadrotor

Table 8.2.1: Algorithmic Parametrization for Simulated Flight

Parameter	Value	Measurement Unit
γ_1	30	m
γ_2	6.3	m
γ_3	0.3	m
$\mathcal{D}_{thres}$	6	m
$\mathcal{F}_{max}$	5	N
$\mathcal{D}_{min}$	1	m
V_{thres}	1	m/s

framework has been validated against its real-world counterpart, thus we believe that the experimental results obtained in these simulation are highly comparable to those from real-world.

8.2.4 Simulation Scenario and Setup

The simulation scenario consists of one static cubical object o_1 with size $1 \times 1 \times 3m$, one moving cylindrical object o_2 with size $0.3 \times 3m$, and one quadrotor UAV which is deemed as a sphere with radius γ_3. From top-down view, o_1 stands at x-y-location $(50, 1)m$, o_2 moves in a direction towards o_1 and the UAV with constant velocity $(-1.2, 0.8)m/s$, and the UAV is constantly commanded to move forward with a high speed $5m/s$ towards both objects, for example, as shown in Fig. 8.2.5. The moving object is designed in such a way that it mimics a person in walking speed. During the whole flight, the UAV is maintained at a certain height (in this case $2m$) by an altitude controller.

Additionally, our proposed collision avoidance system has been parametrized as in Table 8.2.1. Furthermore, we have pitched the 2D LIDAR onboard the simulated UAV by $0.2rad$ so that the LIDAR faces slightly upwards (the short red line on one arm of the UAV represents the front), as shown in Fig. 8.2.4. Since during a high-speed flight, a realistic UAV would constantly have a pitching angle in order to maintain the speed, which undesiredly results in that the onboard parallelly-placed LIDAR constantly facing the ground other than the front.

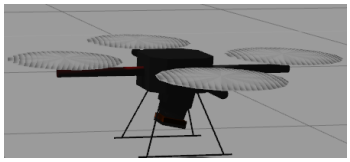


Figure 8.2.4: Simulated UAV with Adapted LIDAR

Therefore, it is necessary to make this adjustment to ensure effective perception.

8.2.5 Results Discussion

With the aforementioned simulation scenario, we run the simulation experiments 20 times, however, the UAV is commanded to take off at different timing so as to vary the timing and locations in which the UAV encounters the obstacles. The experiments results are shown in Table 8.2.2. Among the 20 simulated flights, only one flight failed which is caused by slow response of the UAV during dramatic turning. This issue can be potentially solved by tuning the parameters of the avoidance algorithm, which we will investigate further in future research. Among the successful 19 flights, the shortest peripheral distance between the UAV and obstacle is $0.64m$, which is sufficient distance to avoid obstacles.

In the following we detail one of the successful experiments, whose 3D view is as depicted in Fig. 8.2.5. Several salient moments of the flight are captured, as shown in Fig. 8.2.6. Illustration of the meaning of different colors and shapes used can be found in the caption of Fig. 8.2.6, while the evolution of the Sense-and-Avoid process presented in the subfigures is detailed as follows:

Subfigure at time $t_1 = 40.70s$ displays the moment when the UAV starts to detect the presence of static object o_1 in its conflict zone $\boldsymbol{C}$, and decides that avoidance action is necessary. It memorizes the most recently received control command from the human operator as intent velocity command $\boldsymbol{v}^b_{intent}$. Additionally, the intrusion of o_1 starts to cause generation of a virtual force field which subsequently acts on the UAV to deviate it from the originally intended

Table 8.2.2: Results of Multiple Flights

No. of Flights	Success Rate	Shortest Peripheral Distance
20	95%	$0.644m$

path if the UAV purely follows the intention velocity command $\boldsymbol{v}^b_{intent}$, so as to avoid collision. Subfigure at time $t_2 = 41.90s$ depicts the moment when the UAV is at the closest location to o_1 during the course of avoidance. From Fig. 8.2.7, we can see that their peripheral distance at this point is $d_{o_1,min} = 1.6519m$. Subfigure at time $t_3 = 43.30s$ shows the moment when the UAV starts to detect the presence of the approaching object o_2 in zone $\boldsymbol{C}$ shortly after o_1 is out of it. Therefore, o_2 triggers again the generation of virtual force field which redirects the UAV to avoid collision with o_2. Subfigure at time $t_4 = 44.00s$ depicts the moment when the UAV is at the closest location to o_2 during the course of avoidance. From Fig. 8.2.7, we can see that their peripheral distance at this point is $d_{o_2,min} = 1.8723m$. Subfigure at time $t_5 = 45.40s$ describes the moment when o_2 is just out of zone $\boldsymbol{C}$ of the UAV. From now on the UAV has no obstacles in sight, i.e. in both zone $\boldsymbol{D}$ and $\boldsymbol{C}$, it starts to restore its flight direction to the state before the avoidance course takes place. Subfigure at time $t_6 = 48.00s$ depicts the moment when the UAV has fully returned back to the intended velocity $\boldsymbol{v}^b_{intent}$.

We assume that after successfully returning back to the last commanded velocity (intent velocity), the UAV would send a signal to RC to trigger it to slightly vibrate or make a sound so that the human operator is notified that the control of the UAV has been returned.

Fig. 8.2.7 provides us with a numerical analysis of the simulated flight. From the first subfigure, we can see that the altitude controller is functioning well in maintaining the desired altitude. The control command from the operator has consistently been $5m/s$ forward in the body frame. Therefore at time t_1, the Sense-and-Avoid system of the UAV takes this command as its intended velocity. During the whole course of collision avoidance, the UAV tries to maintain its velocity magnitude v_{mag} constant at $5m/s$, though from the second subfigure in Fig. 8.2.7 we see that between t_1 and t_5 there are small deviations from the desired

value, which is caused by the rapid avoidance manoeuvre, which we could infer from the third and fourth subfigures. The UAV is, however, capable of rapidly restoring the desired velocity by adjusting the pitch angle accordingly, as shown in the Euler angle subfigure. Additionally, as the UAV steers its moving direction by yawing, for a realistic UAV it inevitably causes small roll movements, also as shown in the Euler angle subfigure.

In our supplementary video, we present the simulated flight in Gazebo and its corresponding visualization in rviz. In rviz, the created map consists of detected obstacles (circles) is visualized, as well as the generated virtual force caused by intrusion of the obstacles. The first scenario in the video is the corresponding simulation for the results discussed in this section. The second scenario in the video presents a more challenging situation.

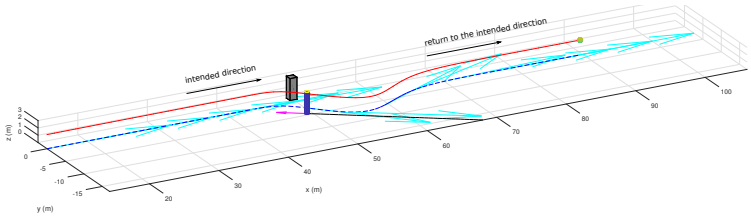


Figure 8.2.5: 3D View of Simulation Scenario: The red solid line represents traveled 3D trajectory of the UAV, the blue dashed line is its projection onto the ground, the cyan arrows attached to the blue dashed line represent projected velocity directions at different locations and time instances. The greenish sphere attached to one end of the red line represents the UAV. The black line is the projected trajectory of the moving cylindrical obstacle o_2, the magenta arrow attached the black line stands for the velocity direction of o_2. The cubical shape is the static obstacle o_1

8.3 Summary

In this chapter, we present a complete 2D LIDAR based UAV teleoperation assistance system to automate the collision avoidance task with focus on the progressive situation evaluation policy and the novel AVCFF-based reactive

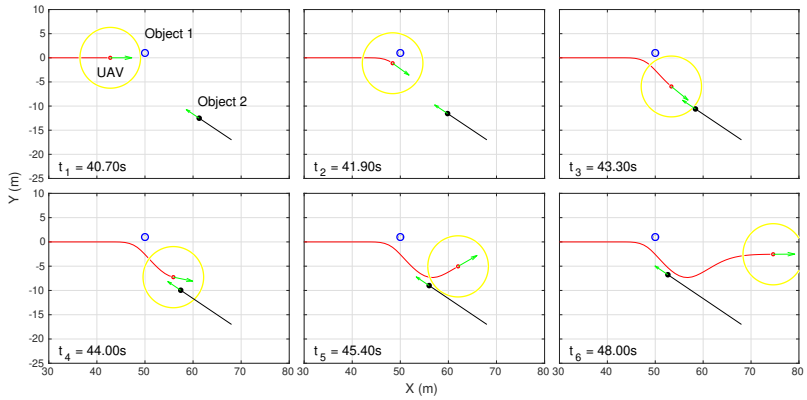


Figure 8.2.6: Simulated Flight from Top-down View: solid lines represent traveled trajectories (red for UAV, black for moving object o_2). The red circle visualizes the exterior border of minimum safety space for the UAV, so is blue circle for static object o_1 and black circle for object o_2. The green arrows represent the velocity directions of respective subjects at the locations and time instances

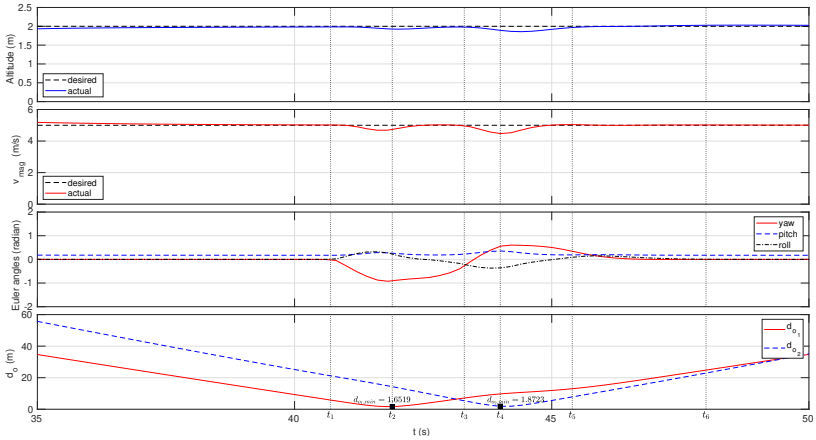
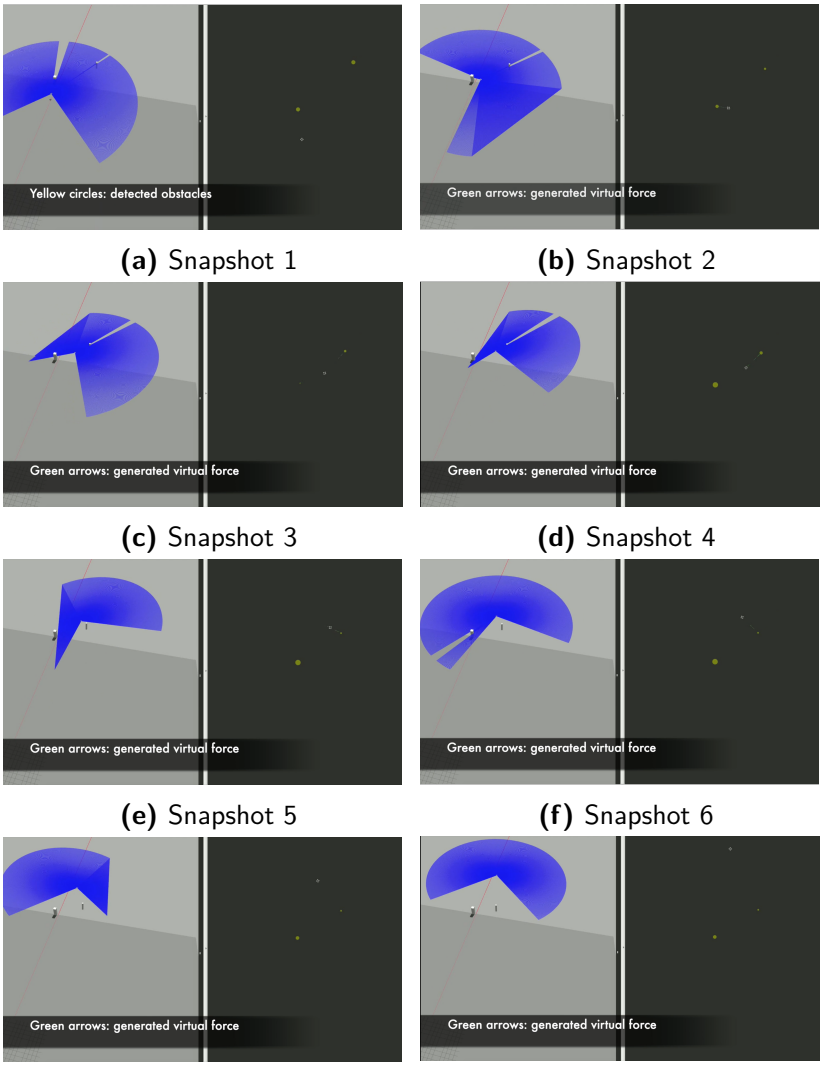


Figure 8.2.7: Numerical Analysis of Simulated Flight

avoidance method. With all of the functional components working in harmony, the effectiveness of the proposed assistive automatic collision avoidance system is demonstrated and validated on an realistic simulated UAV platform in several Gazebo simulations, where UAV is operated at high speed in complex environment. Some snapshots of the flight in Gazebo simulation are shown in Figure 8.3.1



(a) Snapshot 1 **(b)** Snapshot 2

(c) Snapshot 3 **(d)** Snapshot 4

(e) Snapshot 5 **(f)** Snapshot 6

(g) Snapshot 7 **(h)** Snapshot 8

Figure 8.3.1: Simulated Flight Visualized in Gazebo and RVIZ

Do not let the memories of your past limit the potential of your future. There are no limits to what you can achieve on your journey through life, except in your mind.

Roy T. Bennett

9

Conclusions and Future Work

This thesis investigated possible approaches for assisting amateur UAV pilots/operators during teleoperation by automating the obstacle avoidance task, which help to alleviate the workload on the UAV operator so that he/she can concentrate on higher-level decisions and global objectives during teleoperation.

We presented the complete concept of an automatic collision avoidance (Sense-and-Avoid) system for UAV, and proposed different approaches for each functional components of the system, and finally integrated all the functional components into one possible solution.

9.1 Sensing

Specifically for sensing system, we present a practical 2D LIDAR based object detection and tracking approach for the purpose of collision-free navigation, in which objects in the environment are modeled in a compact form as cylinders. The results suggest that the proposed system is capable of classifying and tracking objects with reasonable accuracy, as well as estimating velocities of the dynamic objects. This approach represents objects in the environment with a compact form, therefore it is memory-efficient, and the amount of data need to be processed from 2D LIDAR is relatively few especially in comparison to cameras and 3D LDIAR, therefore this approach is not computationally expensive and can be run onboard in real-time, which thus enable the robot to be more responsive to the changes in the environment. The proposed model fitting method is robust, because it approximates the detected object into cylinder shape regardless of the underlying shape for the object in concern, while also keeping the fitted models good for collision-free navigation.

In the future we plan to incorporate other forms of 2.5 models to represent environments where cylinder-shaped model is not optimal, for example, using cylinder shape to model rectangular-cuboid-shaped objects might cause unnecessary loss of operational or motion space of the UAV. We will also try to represent the objects in the environment directly in full 3D models. Additionally, we will combine this perception solution with obstacle avoidance algorithms to accomplish a complete Sense-and-Avoid solution taking advantage of the estimated velocity of objects in the environment.

9.2 Flight Controller

We also proposed two perception-assistive flight controllers which help to always direct the perception sensor in the direction where the UAV manoeuvres, one is a nonlinear flight controller based on feedback linearization technique, the other is an adapted PID controller constructed on the PID controller provided by the

Hector Quadrotor open source framework.

9.3 Emergency Evaluation and Avoidance

We proposed two sets of emergency evaluation policies and collision avoidance strategies, one set for the optimization based avoidance system, and the other set for the AVCFF (Adaptive Virtual Cushion Force Field) based avoidance system.

The emergency evaluation policy for optimization based avoidance system divides the space around the UAV into four types of safety zones, depending on the safety zone an obstacle is in and the estimated time-to-collision (TTC), the UAV will decide if it should take avoidance actions. The optimization based avoidance system takes inspiration from missile guidance, which formulates the reverse of the conditions for being on a collision course into an constrained optimization problem in order to find the appropriate velocity commands for avoiding potential collisions. This system has been validated together with the nonlinear perception-assistive flight controller in Matlab simulations. The simulation results suggest that our proposed Sense-and-Avoid solution is capable of handling complex environments where multiple moving objects are present, and thus ensure safe flight of the UAV.

The emergency evaluation policy for AVCFF based avoidance system is similar, but only divides the space around the UAV into three types of safety zones, as it is in a high-speed scenario. The core idea of AVCFF method is that the UAV carries a deformable circular safety cushion around it and whenever an obstacle intrudes this safety cushion causing its deformation, the safety cushion would want to recover its original form by bouncing away from the obstacle, which is caused by the repulsive force exerted on the UAV. This system has been integrated with our proposed sensing system and PID based perception-assistive flight controller, as well as the Hector Quadrotor framework. With all of these components working in harmony, the effectiveness of the proposed assistive automatic collision avoidance system is demonstrated and validated on an realistic simulated UAV platform in several Gazebo simulations, where UAV is

operated at high speed in complex environments.

In the future we plan to further validate proposed algorithms on a real-world UAV in different and even more challenging scenarios, and compare the proposed collision avoidance approaches quantitatively, in order to gain a more comprehensive view of the proposed solution.

References

[1] Federal Aviation Administration. *Unmanned aircraft systems operations in the U.S. national airspace system. Interim Operational Approval Guidance 08-01*. 2008.

[2] European Aviation Safety Agency. *Airworthiness certification of unmanned aircraft systems (UAS)*. Policy Statement, E.Y013-01, 2009.

[3] Abdulla Al-Kaff, Fernando García, David Martín, Arturo De La Escalera, and José María Armingol. Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for uavs. *Sensors*, 17(5): 1061, 2017.

[4] BM Albaker and NA Rahim. A survey of collision avoidance approaches for unmanned aerial vehicles. In *2009 International Conference for Technical Postgraduates (TECHPOS)*, pages 1–7, 2009. doi: 10.1109/TECHPOS.2009.5412074.

[5] Asma Azim and Olivier Aycard. Detection, classification and tracking of moving objects in a 3d environment. In *2012 IEEE Intelligent Vehicles Symposium*, pages 802–807, 2012.

[6] Daman Bareiss, Joseph R Bourne, and Kam K Leang. On-board model-based automatic collision avoidance: application in remotely-piloted unmanned aerial vehicles. *Autonomous Robots*, 41(7):1539–1554, 2017.

[7] Marcelo Becker, Richard Hall, Sascha Kolski, Kristijan Maček, Roland Siegwart, and Björn Jensen. 2d laser-based probabilistic motion tracking in urban-like environments. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 31(2):83–96, 2009.

[8] Karsten Berns and Ewald Von Puttkamer. Autonomous land vehicles. In *Vieweg+ Teubner, Wiesbaden*. Springer, 2009.

[9] Josh Bongard. Probabilistic robotics. sebastian thrun, wolfram burgard, and dieter fox.(2005, mit press.) 647 pages, 2008.

[10] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, Sep. 1989. doi: 10.1109/21.44033.

[11] A. M. Brandt and M. B. Colton. Haptic collision avoidance for a remotely operated quadrotor uav in indoor environments. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2724–2731, 2010. doi: 10.1109/ICSMC.2010.5641798.

[12] Brenda Carpenter, James Kuchar, Brenda Carpenter, and James Kuchar. Probability-based collision alerting logic for closely-spaced parallel approach. In *35th Aerospace sciences meeting and exhibit*, page 222, 1997.

[13] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843, 2014.

[14] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, Upper Saddle River, 3nd edition, 2005.

[15] Michael S Darms, Paul E Rybski, Christopher Baker, and Chris Urmson. Obstacle detection and tracking for the urban challenge. *IEEE Transactions on intelligent transportation systems*, 10(3):475–485, 2009.

[16] Martin Dekan, Duchoň František, Babinec Andrej, Rodina Jozef, Rau Dávid, and Musić Josip. Moving obstacles detection based on laser range finder measurements. *International Journal of Advanced Robotic Systems*, 15(1): 1729881417748132, 2018.

[17] A. Ess, B. Leibe, K. Schindler, and L. van Gool. Moving obstacle detection in highly dynamic scenes. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 56–63, 2009.

[18] Giancarmine Fasano, Domenico Accado, Antonio Moccia, and David Moroney. Sense and avoid for unmanned aircraft systems. *IEEE Aerospace and Electronic Systems Magazine*, 31(11):82–110, 2016.

[19] Lía García-Pérez, María C García-Alegre, Ángela Ribeiro, Domingo Guinea, and Jose María Cañas. Perception and tracking of dynamic objects for optimization of avoidance strategies in autonomous piloting of vehicles. In *International Conference on Spatial Cognition*, pages 500–517, 2004.

[20] Gwennael Gate and Fawzi Nashashibi. An approach for robust mapping, detection, tracking and classification in dynamic environments. In *2009 International Conference on Advanced Robotics*, pages 1–6, 2009.

[21] C. Gellius. Noctes atticae (attic nights) book 10 (trans. by rolfe, j. c.). http://penelope.uchicago.edu/Thayer/E/Roman/Texts/Gellius/10*.html, 1927.

[22] Bruno Giovanini, Hugo A Oliveira, and Paulo FF Rosa. An automatic collision avoidance approach to assist remotely operated quadrotors. In *International Conference on Intelligent Autonomous Systems*, pages 213–225. Springer, 2016.

[23] Victor Ho, Clark Borst, Marinus M van Paassen, and Max Mulder. Increasing acceptance of haptic feedback in uav teleoperation by visualizing force fields. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3027–3032. IEEE, 2018.

[24] Stefan Hrabar. Reactive obstacle avoidance for rotorcraft uavs. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4967–4974. IEEE, 2011.

[25] Jason Israelsen, Matt Beall, Daman Bareiss, Daniel Stuart, Eric Keeney, and Jur van den Berg. Automatic collision avoidance for manually tele-operated unmanned aerial vehicles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6638–6643. IEEE, 2014.

[26] Lentin Joseph and Jonathan Cacace. *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System*. Packt Publishing Ltd, 2018.

[27] Christoforos Kanellakis and George Nikolakopoulos. Survey on computer vision for uavs: Current developments and trends. *Journal of Intelligent & Robotic Systems*, 87(1):141–168, 2017.

[28] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, March 1985. doi: 10.1109/ROBOT.1985.1087247.

[29] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[30] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404. IEEE, 1991.

[31] James Kuchar, John Andrews, Ann Drumm, Tim Hall, Val Heinz, Steven Thompson, and Jerry Welch. A safety analysis process for the traffic alert and collision avoidance system (tcas) and see-and-avoid systems on remotely piloted vehicles. In *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, page 6423, 2004.

[32] Mustapha Lakrouf, Stanislas Larnier, Michel Devy, and Nouara Achour. Moving obstacles detection and camera pointing for mobile robot applications. In *Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering*, pages 57–62, 2017.

[33] Thanh Mung Lam, Harmen Wigert Boschloo, Max Mulder, and Marinus M Van Paassen. Artificial force field for haptic feedback in uav teleoperation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(6):1316–1330, 2009.

[34] Frederik Stendahl Leira. Object detection and tracking with uavs: A framework for uav object detection and tracking using a thermal imaging camera. 2017.

[35] Imen Mahjri, Amine Dhraief, and Abdelfettah Belghith. A review on collision avoidance systems for unmanned aerial vehicles. In *International Workshop on Communication Technologies for Vehicles*, pages 203–214. Springer, 2015.

[36] T. Marinho, M. Amrouche, V. Cichella, D. Stipanović, and N. Hovakimyan. Guaranteed collision avoidance based on line-of-sight angle and time-to-collision. In *2018 Annual American Control Conference (ACC)*, pages 4305–4310, June 2018. doi: 10.23919/ACC.2018.8431871.

[37] Abel Mendes, L Conde Bento, and Urbano Nunes. Multi-target detection and tracking with a laser scanner. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 796–801. IEEE, 2004.

[38] João Mendes and Rodrigo Ventura. Assisted teleoperation of quadcopters using obstacle avoidance. *Journal of Automation Mobile Robotics and Intelligent Systems*, 7(1):54–58, 2013.

[39] Christoph Mertz, Luis E Navarro-Serment, Robert MacLachlan, Paul Rybski, Aaron Steinfeld, Arne Suppe, Christopher Urmson, Nicolas Vandapel, Martial Hebert, Chuck Thorpe, et al. Moving object detection with laser scanners. *Journal of Field Robotics*, 30(1):17–43, 2013.

[40] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411. Springer, 2012.

[41] Gonçalo Monteiro, Cristiano Premebida, Paulo Peixoto, and Urbano Nunes. Tracking and classification of dynamic obstacles using laser range finder and vision. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–7, 2006.

[42] Kenzo Nonami, Farid Kendoul, Satoshi Suzuki, Wei Wang, and Daisuke Nakazawa. *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. Springer Science & Business Media, 2010.

[43] M. Odelga, P. Stegagno, and H. H. Bülthoff. Obstacle detection, tracking and avoidance for a teleoperated uav. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2984–2990, May 2016. doi: 10.1109/ICRA.2016.7487464.

[44] Marcin Odelga, Paolo Stegagno, and Heinrich H Bülthoff. Obstacle detection, tracking and avoidance for a teleoperated uav. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2984–2990. IEEE, 2016.

[45] Marcin Odelga, Paolo Stegagno, Nicholas Kochanek, and Heinrich H Bülthoff. A self-contained teleoperated quadrotor: On-board state-estimation and indoor obstacle avoidance. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7840–7847. IEEE, 2018.

[46] U.S. Department of Defense Office of the Secretary of Defense. *Unmanned systems roadmap 2007–2032*. Report, 2007.

[47] Hung Pham, Scott A Smolka, Scott D Stoller, Dung Phan, and Junxing Yang. A survey on unmanned aerial vehicle collision avoidance systems. *arXiv preprint arXiv:1508.07723*, 2015.

[48] Tomasz Piessens, Rene van Paassen, and Max Mulder. Haptic support for avoiding static and dynamic obstacles in uav tele-operation. In *48th International Symposium on Aviation Psychology*, page 169, 2019.

[49] Cristiano Premebida and Urbano Nunes. Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications. *Robotica*, 2005:17–25, 2005.

[50] Mateusz Przybyła. Detection and tracking of 2d geometric obstacles from lrf data. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, pages 135–141, 2017.

[51] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[52] Syed Ali Raza and Wail Gueaieb. Intelligent flight control of an autonomous quadrotor. In *Motion Control*. IntechOpen, 2010.

[53] K Rebai, A Benabderrahmane, O Azouaoui, and N Ouadah. Moving obstacles detection and tracking with laser range finder. In *2009 International Conference on Advanced Robotics*, pages 1–6, 2009.

[54] Ricardo Roberts, Manlio Barajas, Ernesto Rodriguez-Leal, and José Luis Gordillo. Haptic feedback and visual servoing of teleoperated unmanned aerial vehicle for obstacle awareness and avoidance. *International Journal of Advanced Robotic Systems*, 14(4):1729881417716365, 2017.

[55] Erick J Rodríguez-Seda, Dušan M Stipanović, and Mark W Spong. Collision avoidance control with sensing uncertainties. In *Proceedings of the 2011 American Control Conference*, pages 3363–3368. IEEE, 2011.

[56] E. Sand. Haptic feedback effects on human control of a uav in a remote teleoperation flight task. 2016.

[57] Rajnikant Sharma, Jeffery B Saunders, and Randal W Beard. Reactive path planning for micro air vehicles using bearing-only measurements. *Journal of Intelligent & Robotic Systems*, 65(1-4):409–416, 2012.

[58] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[59] Tetsuro Toda, Gakuto Masuyama, and Kazunori Umeda. Detecting moving objects using optical flow with a moving stereo camera. In *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*, pages 35–40, 2016.

[60] Kimon P Valavanis and George J Vachtsevanos. *Handbook of unmanned aerial vehicles*, volume 1. Springer, 2015.

[61] Holger Voos. Nonlinear control of a quadrotor micro-uav using feedback-linearization. In *2009 IEEE International Conference on Mechatronics*, pages 1–6. IEEE, 2009.

[62] Chieh-Chih Wang and Chuck Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2918–2924, 2002.

[63] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *2003 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 842–849, 2003.

[64] Min Wang and Holger Voos. Safer uav piloting: A robust sense-and-avoid solution for remotely piloted quadrotor uavs in complex environments. In *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019.

[65] D. D. Woods, J. Tittle, M. Feil, and A. Roesler. Envisioning human-robot coordination in future operations. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):210–218, 2004. doi: 10.1109/TSMCC.2004.826272.

[66] Yuanwei Wu, Yao Sui, and Guanghui Wang. Vision-based real-time aerial object localization and tracking for uav sensing system. *IEEE Access*, 5: 23969–23978, 2017.

[67] S.J. Zaloga. Unmanned aerial vehicles: Robotic air warfare 1917–2007. *No. 144 in New Vanguard (Osprey, Oxford/New York, 2008)*.

[68] Paul Zarchan. *Tactical and strategic missile guidance*. American Institute of Aeronautics and Astronautics, Inc., 2012.