Aerial Human-Comfortable Collision-free Navigation in Dense Environments

THÈSE N^{O} 7528 (2017)

PRÉSENTÉE LE 28 AVRIL 2017

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE SYSTÈMES INTELLIGENTS

PROGRAMME DOCTORAL EN ROBOTIQUE, CONTRÔLE ET SYSTÈMES INTELLIGENTS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Nicolas DOUSSE

acceptée sur proposition du jury:

Dr D. Gillet, président du jury
Prof. D. Floreano, directeur de thèse
Prof. H. Bülthoff, rapporteur
Prof. S. Bogdan, rapporteur
Prof. A. Martinoli, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2017

Once you have tasted flight,
you will forever walk the earth
with your eyes turned skyward,
for there you have been,
and there you will always long to return.
— assimilated to Leonardo da Vinci

To my family…

Acknowledgements

This thesis would never have been a success without the help and contribution from many persons. First, I would like to thank my supervisor Prof Dario Floreano for his guidance during my whole thesis. Dario gave me the freedom to give a personal direction to my research work as well as access to a world-class research environment, filled with a team of dynamic researchers and state-of-the-art facilities. He also taught me how to best present my work, keeping in sight at every moment the global picture of the problem.

Then, I would like to thank the members of my jury: Dr Denis Gillet, Prof Alcherio Martinoli, Prof Heinrich Bülthoff and Prof Stjepan Bogdan for taking the time to read my thesis, to provide comments that improved the manuscript and to travel to Lausanne for my thesis defense.

I would like to thank the European 7th framework program, whose funding of the myCopter project made this project possible.

I would also like to thank Doctor Felix Schill for the start of the project. Without his work, the development of the robot as well as the swarm simulator used in this thesis would not have been possible. I would also like to thank Felix for the numerous and fruitful discussions that improved the project, as well as for his precious time reviewing papers especially after he finished his duty for the lab.

A very special thank to my friend and colleague, Grégoire Heitz, who spend countless hours with me doing outdoor experiments, in every weather conditions, who supported me in difficult time and who never stop believing in me.

During my time in the lab, I supervised 7 semester projects and 1 Master project. Some of this work contributed to this thesis, especially the work from Gaëtan Burri, whose motivation and work style were outstanding.

A special think for Ramon, who took the time to read this thesis and provide useful feedbacks.

Paul Ryan said “Every successful individual knows that his or her achievement depends on a community of persons working together. ” Even if the thesis is an individual accomplishment, the path to its completion was eased by wonderful colleagues at LIS. I especially think to Ludovic, Julien, Basil, Ramon, Stefano, Maja, Przemek, Pavan, Géraud, Andrea, Adrien, Alice, Carine and Michelle.

Finally, I would like to thank my family, especially my parents André and Remei and my brother Alexandre for their endless support during all these years, through good and bad times. I feel blessed to have grown up in such a loving, supportive and united family.

Last but not least, I would like to thank Céline, for her unconditioned support and her endless love during the difficult time she spent being besides me these last months as well as for all

the good time we spent together.

Lausanne, 14 March 2017 N. D.

Abstract

With current overuse of the road transportation system and planned increase in traffic, innovative solutions that overcome environmental and financial cost of the current system should be assessed. A promising idea is the use of the third dimension for personal transportation. Therefore, the European project myCopter, funded under the 7^{th} framework, aimed at enabling the technologies for Personal Aerial Transportation Systems as breakthrough in 21^{st} century transportation systems. This project was the starting point of this thesis.

When multiple vehicles share a common part of the sky, the biggest challenge is the management of the risk of collision. While optimal collision-free navigation strategies have been proposed for autonomous robots, trajectories and accelerations for Personal Aerial Vehicles (PAVs) should also take into account human comfort for their passengers, which has rarely been the focus of these studies. Comfort of the trajectories is a key factor in order for this new transportation mean to be accepted and adopted by everyday users.

Existing strategies used to maximize human-comfort of trajectories are based on path planning strategies, which compute beforehand the whole trajectory, implementing comfort as an optimization criteria. Personal Aerial Transportation Systems will have a high density of vehicles, where the time to react to potential threats might decrease to a few seconds only. This might be insufficient to compute a new trajectory each time using these path planning strategies. Therefore, in this thesis, a reactive decentralized strategy is proposed, maximizing the comfort of the trajectories for humans traveling in a Personal Aerial Vehicle.

To prove the feasibility of collision avoidance strategies, it is not sufficient anymore to validate them only in simulation, but, in addition, real-time tests in a realistic outdoor environment should be performed. Nowadays, single drones can be effectively controlled by a single operator on the ground. The challenge relies instead on an efficient management of a whole swarm of drone. In this thesis, a framework to perform outdoor drone experiment was developed in order to validate the proposed collision avoidance strategy. On the one hand, an autopilot framework was developed, tailored for multi-drone experiments, allowing fast and easy deployment and maintenance of a swarm of drones. On the other hand, a ground control interface is proposed in order to monitor, control and maintain safety in a flight with a swarm of drones.

Using the autopilot framework together with the ground control interface, the proposed collision avoidance strategy was validated using 10 quadrotors flying autonomously outdoor in a challenging scenario.

Acknowledgements

Key words: Human-comfort, Collision-free navigation, Human-Swarm interface, Swarm experiments, UAV, Drones, MAV, Safety, Regulations

Résumé

Avec la sur-utilisation du réseau routier ainsi que l'augmentation prévue du trafic, de nouvelles solutions doivent être évaluées afin de surmonter le coût environnemental et financier du système actuel. Une idée prometteuse est l'utilisation de la troisième dimension pour le transport personnel. Le projet myCopter financé par l'Union Européenne tend ainsi à développer les technologies pour un système de transport personnel aérien comme percée dans les systèmes de transport du $21^{ème}$ siècle.

Lorsque de multiples véhicules aériens se partagent une portion de l'espace aérien, le défi le plus grand est la gestion du risque de collision. Bien qu'il existe de nombreuses stratégies d'évitement de collisions pour des robots autonomes, les trajectoires et accélérations de véhicules personnels aériens doivent aussi prendre en compte le confort des passagers, ce qui a rarement été le cas des précédentes études. Le confort des trajectoires est un élément décisif quant à l'acceptation et à l'utilisation par une vaste majorité de ce nouveau moyen de transport.

Les stratégies existantes utilisées pour maximiser le confort de trajectoires pour les Hommes sont basées sur des stratégies de planification de trajectoire, calculant à l'avance l'entier de la trajectoire et utilisant le confort comme un objectif d'optimisation. Un système de transport aérien personnel aura une densité de véhicules élevée, dans lequel le temps de réaction envers de potentielles menaces de collision peut diminuer jusqu'à quelques secondes seulement. Cela peut être insuffisant pour recalculer à chaque fois une nouvelle trajectoire par le biais de ces stratégies de planification. Par conséquent, dans cette thèse, une approche décentralisée et réactive est proposée à la place, afin de maximiser le confort des trajectoires pour les Hommes voyageant à bord de véhicules personnels aériens.

Afin de prouver la faisabilité de ces stratégies, il n'est plus suffisant de les valider uniquement en simulation mais en plus, des tests en temps réel doivent être effectués dans un environnement réaliste et extérieur. De nos jours, le management d'un drone volant seul peut être effectué de manière efficiente par un seul opérateur depuis le sol. Le challenge réside dorénavant dans le management efficace d'un essaim de drones volant simultanément. Dans cette thèse, le cadre permettant d'effectuer des expériences avec plusieurs drones à l'extérieur a été développé afin de pouvoir valider la stratégie d'évitement de collision. D'une part, un système d'autopilote a été développé permettant un déploiement et une maintenance rapide et aisée d'un essaim de drones. D'autre part, une interface pour le contrôle au sol des drones est proposée afin de surveiller, contrôler et maintenir la sécurité d'un vol avec un essaim de drones. Grâce à cet autopilote ainsi que cette interface, la stratégie d'évitement de collision

proposée a pu être validée sur 10 quadrotors volant de manière autonome à l'extérieur dans un scénario ardu.

Mots clefs : Confort, navigation sans collision, Interface homme-essaim de drones, drones, sécurité aérienne, régulations

Contents

List of Figures

List of Tables

1 Introduction

WITH current overuse of the road transportation system and planned increase in traffic, innovative solutions that overcome environmental and financial cost of the current system should be assessed. A promising idea is the use of the third dimension for personal transportation. This statement was the basis of this thesis performed within the European funded project myCopter [1] that aimed to enable the technologies for Personal Aerial Transport Systems (PATS) as breakthrough in 21^{st} century transportation systems. The details of the myCopter project are presented in Section 1.1.1.

The more vehicles you have in the air, the higher the risk of collision between some of them is. To convince the general public to use Personal Aerial Vehicles (PAVs), it must be shown that it is safe enough to travel. In addition, with humans onboard, it adds a new dimension with the notion of comfort. The comfort of the trajectory should receive a particular attention in order for the passengers to experience a quiet journey. However, comfort has rarely been the focus of studies in collision avoidance with dynamic obstacles.

In addition, drones are leaving the controlled environments of laboratories with complex tracking systems and start to fly outdoor, relying only on GPS. Therefore, in order to be widely accepted, newly developed strategies should also now be tested and validated through realistic multi-drone outdoor flight tests.

In this thesis, a reactive decentralized collision-free strategy is proposed, maximizing the comfort for passengers of PAVs. This strategy is then validated through extensive simulations as well as outdoor flights using a framework developed to enable multi-drones experiments.

In this introductory chapter, the motivation and challenges to perform human-comfortable collision-free navigation validated not only by extensive simulations but also by realistic tests with drones are presented in details in Section 1.1. Then, the relevant state of the art is presented in Section 1.2 and gives potential solution to perform human-comfortable collision-free navigation. Finally, the main contribution of the thesis and its organization are presented in Section 1.3.

1.1 Motivation and challenges

1.1.1 Scope of the thesis

In many cities, the saturation of the road transportation system and predicted traffic increment [2] demand for innovative alternative solutions. A promising idea for congested cities is the use of the third dimension for personal transportation in a environmental neutral way (Fig. 1.1). Recently, several projects [3], [4] have investigated the technologies for Personal Aerial Transport Systems (PATS) as breakthrough in 21^{st} century transportation systems. Accordingly, semi- or fully-autonomous Personal Aerial Vehicles (PAVs) are currently studied and developed by public [1], [5] and private[1] organizations (Fig. 1.2), which confirms a trend towards Personal Aerial Transportation Systems.

This thesis was part of the myCopter project. This project was funded by the European Union under the 7^{th} Framework program.

The myCopter project focused on the system as a whole and not on the design of a particular platform and tended to enable the technologies for Personal Aerial Transportation Systems (PATS). We believe that the only approach leading to a popular acceptance of PAVs is to not only consider platform specifications but also more general aspects such as socio-economic impact and safe navigation of many entities.

In the myCopter project, we imagined a PATS confined below controlled airspace which extends down to 500-600 meters above ground and will not interact with airliners flying above. One possible application scenario for a PATS would be in a commuting scenario, where PAVs will leave the suburbs of large cities and fly towards downtown in the morning and fly back in the evening.

The research was split between six different partners. Novel human-machine interfaces were studied at the Max-Plack-Institut für biologische Kybernetik. Modeling of PAV concepts and

[1] The interested reader can find a large comparison of PAV at http://en.wikipedia.org/wiki/Comparison_of_personal_air_vehicles

Figure 1.1 – Concept for Personal Aerial Vehicles flying over a city [6].

(a) Transition by Terrafugia [7]. (b) M400 by Moller [8]. (c) Aerocar by Taylor [9].

Figure 1.2 – Selected examples of PAVs.

defining flying qualities were studied at the University of Liverpool. Automation algorithms for determining landing spots, automatic take-off and landing was studied at EPFL-CvLab. Definition and development of control strategies for automating flight of a single Personal Aerial Vehicle for automatic takeoff, navigation and landing were studied at ETHZ. Investigation of the socio-economico-technological contexts was studied at the Karlsruher Institut für Technologie. Evaluation of newly developed technologies using the Flying Helicotper Simulator was studied at the Deutsches Zentrum für Luft- und Raumfahrt in Braunschweig and finally development of collision avoidance strategies and formation flying were studied at EPFL-LIS. This last point is the starting point of this thesis.

1.1.2 Aerial collision-free navigation with human onboard

The planned high density of PAVs rises the issue of the risk of mid-air collisions between PAVs. There is thus a need for a collision avoidance strategy that provides safe navigation. To convince the general public to use PAVs, it must be shown that it is safe enough to travel.

To evaluate how bad the situation can be, i.e. the worst case scenario, where no collision avoidance strategy is applied, large scale simulations were performed. In these simulations, vehicles were autonomously flying between two waypoints with no collision avoidance strategy. The number of collision per hour was computed. Results are presented in Appendix A.1 and validate the need for a collision avoidance strategy for PAVs.

With passengers on board, the question of comfort during the flight has to receive a particular attention. For every day use, the collision avoidance strategy should be able to deal not only with passengers that don't want to be shaken in all directions during their commuting journey but also with short reaction time due to the high number of vehicles sharing the airspace. Numerous collision avoidance strategies exists (see Section 1.2.3 for a detailed analysis) but comfort has rarely been the focus of these study.

1.1.3 Experiments with multiple drones

Multi-drones experiments are not limited anymore to simulation only [10] or to outdoor experiments including 3-4 robots flying together with almost no direct interaction [11]. Both [12] and [13] demonstrates the feasibility of outdoor swarming experiments with 10 flying robots.

In order to be widely accepted, collision avoidance strategies should therefore be validated not only in simulation but also in a realistic environments, where flying vehicles really detect potential threats and perform the avoidance maneuvers.

There is thus a will to have a proof of feasibility in realistic environments for newly developed strategies.

The technical challenge of swarm applications does not rely anymore on the control of a single robot but on the efficient management of the whole swarm of robots. In [14] and [15], the authors showed that a single operator could handle at most five robots. In [16], the authors proposed metrics to predict the maximal number of robots that can be controlled by a single operator in a military scenario and deduced a similar number of 5 robots. In order to achieve swarming missions with more than five robots, the operator should either have tools to decrease the workload or the swarm should be controlled by multiple operators.

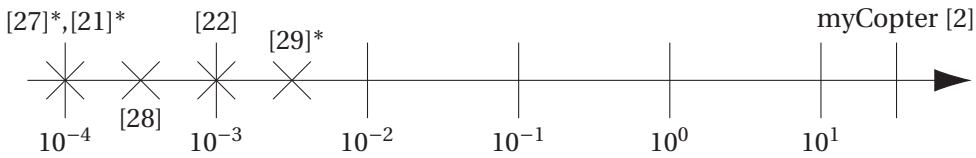
The decrease of workload can be done using two axis. First, interactions between the swarm and the human operator should be made such that we give the operator a view of the swarm as close as the reality as possible, in a shorter time as possible and show only the currently useful information. Second, the robot should have a level of autonomy such that the actions required to drive the mission are minimal.

1.2 State-of-the-art

In this section, first relevant publications in Air Traffic Management are first presented in Section 1.2.1 and motivate the use of decentralized collision avoidance strategies. Then, researches on human comfort are presented in Section 1.2.2 and show the limitations of existing approaches. In Section 1.2.4, bio-inspired strategies modeling systems with multiple entities navigating safely in dense environments are presented and give potential solution. Finally, in Section 1.2.5, relevant publications in human-swarm interactions are presented, driving the need for an framework that is tailored for multi-drone experiments.

1.2.1 Air Traffic Management

Current Air Traffic Management (ATM) relying on a centralized human controller using voice to control its sector of the airspace has reached its limits. Delays due to ATM and inefficient routing count as more than 50% of the whole cost of delays [17]. Air Traffic growth has been and planed to be exponential with a planed increase of about 3% in the incoming



Vehicles per cubic kilometer.

*: Vehicles per square kilometer.

Figure 1.3 – Density considered in ATM.

years [18]. The aeronautical community is developing new strategies to overcome these issues. The most widespread idea is the Free Flight concept [19]. Free Flight wants the shift of collision avoidance responsibility from a central entity to the user of the Air space i.e. to use decentralized control strategies for collision avoidance. Decentralized control strategies will allow more direct flight routes from departing to destination airport as the controller will not be required to ensure separation between aircraft by forcing flights in airways, over defined waypoints. This approach has the appreciable side effect of suppressing the single point of failure when communication with ground based Air Traffic Controller is cut. Furthermore, [20] showed that the distance of flight could be reduced by 25% if Free Flight concept would be applied.

In a PATS, there could be up to 40 vehicles per cubic kilometer [2] while the densities usually assumed in Air Traffic systems are up to 4 orders of magnitude below (Fig. 1.3) [21], [22]. Thus, proposed strategies for air traffic systems (decentralized [23], [24] or centralized [25], [26]) may not be suited for high densities.

In ATM, vertical separation is ensured by flying on fixed flight levels. Most of conflicts occurs in the horizontal plane. Collision avoidance strategies therefore limit their avoidance maneuver to the horizontal plane in order not to violate the vertical separation and thus do not take advantage of full 3D liberty. Vertical avoidance is used only as last resort option. With increasing densities, the use of the whole 3D avoidance maneuver scope might become necessary.

NASA acknowledges that the current Air Traffic system reaches saturation and would not scale much further. As NASA is expecting that 10 millions vehicles per day will share the US airspace system, they recently launched a Sky for All challenge [30] to find “creative, clean-slate design constructs, or enabling component technologies and concepts, that will inform the design of real-world future air transportation“ [31].

Therefore, it is worth considering decentralized collision avoidance strategy in PATS with such high densities.

1.2.2 Comfort

While optimal collision-free navigation strategies have been proposed for autonomous robots (see Section 1.2.3 for a detailed analysis), trajectories and accelerations for Personal Aerial Vehicles (PAVs) should also take into account human comfort. Indeed, PAVs will become widely used not only if it is safe enough to travel but also if it is comfortable enough to travel on a regular basis. Regular use of this new transportation system will be achieved only if a certain level of comfort can be guaranteed [32].

It is not yet fully understood how and what influences human comfort as it is largely influenced by subjective perception, which varies a lot between individuals [33]. What is considered as comfortable for someone might already be uncomfortable for someone else.

Comfort studies in the aeronautical world highlight some of the conditions that lead to discomfort. Hinninghofen highlight the in-cabin factors that influence the comfort of passengers such as noise, seat size, mobility, cabin pressure and humidity [34]. Other studies measured the effects on comfort of the roll oscillation frequency [35], combinations of roll and pitch motion, or combination of roll and vertical oscillations [36], [33]. However, none of these studies propose a solution to minimize discomfort.

Factors that influence comfort can still be separated in two categories [37]: physical motion quantities (acceleration and angular motion) and human interaction variables (human factors such as age, gender, posture, past experience and environmental factors such as temperature, noise, seat shape, leg room). In this thesis, we focus only on physical motion quantities, also named the physiological component of comfort [38].

Looking at an automatic pilot flight manual [39] gives us an insight of what is considered as comfortable in Commercial aviation. An comfortable acceleration is defined as an acceleration of maximum $0.3g$ or $3m/s^2$. With large aircraft traveling at high speeds but in low density (Fig. 1.3), avoidance maneuvers are performed 8 to 15 minutes ahead and usually require only a few degree of heading change [21], implying only small and brief accelerations. In opposite, in dense environments, the time to react to a new threat decreases significantly from a few minutes to only a few seconds. At $100km/h$, the planned cruising velocity [40], in order to keep a separation distance of $50m$ and an acceleration below $0.3g$, one should see other threats at least 8 seconds before collision. Therefore, there is not much time to compute the solution and thus approaches needing expensive computation of the optimal solution in real-time will produce more collisions than strategies with little computation.

Many studies showed that the level of physiological discomfort was correlated with the magnitude of jerk [37], [38], [41], which is the time derivative of acceleration.

Existing studies for ground vehicles minimize the time integral of the square of the jerk in path planning strategies in order to improve the physiological comfort [42], [43]. The ISO norm 2631-4 used for fixed-guidway systems (e.g. trains) also recommends the use of the jerk as

mean to measure and optimize comfort [44]. In addition, aircraft manufacturer companies have submitted patents where jerk is used as design criteria in path planning strategies in order to minimize the discomfort for aircraft in airborne [45], [46] and ground phases [47].

Solving path planning strategies has an exponential complexity in terms of number of vehicles [48]. In addition, using only local sensing, vehicles can only have knowledge of nearby vehicles. Path planning strategies might fail to provide safe and comfortable trajectories as the current situation might change faster than the time required to obtain the new trajectory.

In this thesis, we propose to use instead a reactive decentralized collision avoidance strategy to maximize the comfort of the trajectories for PAVs, flying in dense environments.

1.2.3 State-of-the-art in collaborative collision avoidance

In this section, the detailed state-of-the-art in collaborative collision avoidance is presented.

Research in collision avoidance is a widespread subject: numerous works have been done in this field. For complete surveys, the interested reader should refer to [49], [50] and [51].

The collision avoidance procedure can be divided in two steps: collision detection and avoidance maneuvers: each agent should acquire information about its neighbors, compute whether they are a threat or not and perform an avoidance maneuver when the threat is effective.

Collision detection should predict the future of the trajectory of detected agents. In [49], four types of prediction are presented: nominal prediction, probabilistic prediction, worst-case approach and flight plan sharing. With the small look-ahead time present in dense environments and by using only local non-cooperative sensing, prediction can be performed assuming a nominal (or probabilistic approach) where the agent is assumed to continue or slightly alter his intent during the few next time steps.

Centralized collision avoidance (e.g. [52]) suffer from single point of failure (i.e. when the central planner fails, the whole system fails). In addition, in dense and dynamic environments, the time required by a central planner to compute collision-free trajectories and to transmit them might be prohibitively high.

With local sensing and finite communication range, only an incomplete view of the environment is available to the agents. At any moment, they are only aware of actual or near-future threats. Therefore, the use of decentralized global path planning approaches alone that compute the whole path beforehand will eventually fail to provide a collision free path. Indeed, they can not deal with future threats out of sensing range. In addition, they scale poorly as they have an exponential complexity in terms of number of vehicles [48]. The approach should therefore be able to react to the actual situation and optimize the trajectory only on a finite time horizon related to the sensor capabilities and adapt to newly detected threats.

This is why it is worth considering the use of decentralized reactive collision avoidance strategies. A decentralized strategy is here defined as a strategy where each agent decides its trajectory solely based on locally available information. The focus is made on such decentralized strategies for the rest of this section.

In this section, decentralized collision avoidance strategies are classified by two parameters: collaboration and cooperation. Strategies can by collaborative, i.e. agents agree on an avoidance maneuver or non-collaborative where each agent performs avoidance maneuver in an egoistic optimal way. Strategies can also be cooperative, i.e. agents actively exchange information via a communication link or non-cooperative where only local sensing (e.g. radar, cameras, LIDAR) is used.

Non-collaborative approaches require both agents to perform a full collision avoidance maneuver assuming other agents to continue on their actual track and not reacting to the threat. Therefore they overreact and escape from their optimal navigation route more than necessary. On the contrary, collaborative approaches split responsibility between the two agents, each of them performing half of the maneuver. Such approaches are appealing in order to minimize the deviation and thus minimize the energy cost. Nevertheless, they are affected by failures and agents not respecting the rules. As only half of the avoidance maneuver is performed, if an agent does not react to the threat because of failure, a collision could occur.

Some collaborative approaches need to communicate in order to negotiate a common solution. The main drawback of such approaches is the scalability with respect to computation and negotiation time. In general, the poor scalability of communication bandwidth can be stated as major drawback for any method that needs to actively cooperate to find collision free trajectories. The capacity of a communication network is defined in terms of maximal throughput, which is the time average of the number of bits per second that can be transmitted by every node to its destination [53]. The maximum throughput in a multicast scenario (i.e. where nodes want to transmit to their m neighbors in a n nodes network) scales as $\frac{1}{\sqrt{m}\sqrt{n\log n}}$ [54] i.e. the capacity decreases when the size of the population and/or its density increases. The total communication flow scales quadratically with the number of agents [55] as well as the number of iterations needed to find a solution [56]. Therefore, strategies that need to cooperate in real time might fail to provide collision-free trajectories in dense environments before non real-time cooperative strategies.

Collision avoidance strategies can compute avoidance maneuvers in a sequential way or considering all threats together. In [57], the authors show that in dense environments, sequential conflict resolution leads to a domino effect. The domino effect occurs when the solution to each new threat creates multiple new threats. This reaction continues like a domino chain in the whole system. Therefore, in dense environments, avoidance maneuvers should take the whole local situation into account.

In [50], a taxonomy for collision avoidance is proposed. Collision avoidance strategies are classified in six different main categories. Here, the multi-agent approach has been included

in the Game theory approach. Furthermore, only collaborative collision avoidance strategies will be presented below. The categories are

- Rule-based strategies,
- Game theory strategies,
- Fields strategies,
- Geometric models,
- Numerical optimization.

1.2.3.1 Rule-based strategies

Rule-based strategies define common set of rules to be performed in case of agents on colliding tracks. In [58], the authors propose an extension to Visual Flight Rules (VFR), which are applied in General Aviation. Examples of such rules are: head-on approaching aircraft should both turn right, right of way priority is applied when two aircraft are on colliding tracks and priority is given to the aircraft which is the less maneuverable (e.g. gliders, balloons). These rules are easy to implement, fast in operation, reliable in simple situations and no communication between aircraft is required. However, in dense environments, collisions may occur not only between two aircraft but also between any higher number of aircraft. Therefore rules should be defined for all possible encountered scenarios which is prohibitively complicated.

1.2.3.2 Game theory strategies

Game theory strategies are based on negotiation between aircraft. Negotiation can be either egoistic [59] or altruistic [23], [60], [61]. In the egoistic approach, each agent negotiate in order to maximize its own benefit. In [59], deconfliction is performed by reserving some space ahead of the trajectory. However, for agents that are willing to cooperate, the egoistic solution consisting in maximizing their utility function leads to pessimistic solutions. Altruistic approaches are based on satisficing game theory [62]. Agents open the negotiation set by considering not only their optimal solution but also all "good enough" solutions. In addition, the decision making process is implemented as a *situational altruism*. Agents are willing to sacrifice their own interest to the other's desire if it can benefit to the whole group (altruism) and if and only if the other is taking advantage of it (situational altruism). Aircraft in a given range are ranked by priority, aircraft close to their goal, having higher delays and flying since a longer time, have a higher priority. Conflicts are solved by ranking: lower ranked aircraft will perform more maneuver than higher ranked ones. In [63], the authors extend this work to deal with severe weather conditions, obstacles and no-flying zones.

In [55], a Multi-Party collision avoidance strategy (MPCA) is presented. When two aircraft are on colliding track, they first try to solve the conflict together. If the solutions are either to alter

dramatically their flight plan or only slightly but collide with other aircraft, they can ask to those aircraft to join a group which tend to search for a solution together. New aircraft can join the group as soon as they see a potential collision with the group or that they are needed to seek for new solutions. In [56], efficiency is defined as the ratio of nominal flight plan length to the sum of flight trajectory length after application of conflict resolution. The authors show that their approach leads to better efficiency than the satisficing approach previously explained. However, the group size during the conflict resolution phase is not limited and could potentially lead to communication bandwidth saturation in very dense environments.

The main drawback of such approaches is the scalability with respect to computation and negotiation time. The total communication flow scales quadratically with the number of agents [55] as well as the number of iteration needed to find a solution [56]. Therefore, strategies that need to cooperate in real time will break down sooner than non real time cooperative strategies for increasing densities.

1.2.3.3 Fields strategies

Fields strategies are inspired by particle physics. Agents are modeled as positive charges which are "attracted" by their goal modeled as negative charge. Agents are "repulsed" by other agents. Each agent builds a local artificial potential field induced by other agents and its goal. The controller tends to move the agent towards the minimum of potential. In [64], the authors use an ellipsoid centered in front of the aircraft to generate a higher force in front of the aircraft to better protect the front of the aircraft. Rules are added to solve special cases such as head-on encounters. To overcome the limited sensing capability of each agent and the resulting incomplete view of the environment, [65] designed a control strategy based on magnetic fields. Instead of using artificial potential fields, the author consider each agent as a magnet. Agents measure the total magnetic field locally and use the gradient of the magnetic field to steer towards its goal while avoiding collisions. Agents will sense the global field and thus have a global view of the environment. As a magnetic field decay with the distance, only near neighbors have a significant influence and can be used to perform collision avoidance. Such strategies need to know the environmental magnetic field in order to consider only the influence of neighbors and goal and therefore is not suitable for real-world application as the cost of mapping 3D real-time magnetic field all around the world is prohibitively high.

In general, Fields strategies are non-collaborative strategies where each agent computes full avoidance maneuver. In addition, the two main drawbacks of Fields strategies are the difficulty to take into account acceleration constraints of the agents and the incompleteness of the method i.e. reaching the goal can not be guaranteed. Indeed, the potential field could require to perform infinite acceleration in order to follow the minimum and avoid a collision.

1.2.3.4 Geometric models

Geometric models compute the solution based on the geometric relationship between two agents or more. In this section, the focus will be given to velocity based approaches. In velocity based approaches, the problem is solved in the 3D velocity space instead of considering the 3D space. In [66], Fox introduces the Dynamic Window approach. The space of velocity is constraint to the set of all velocities that can be reached under the dynamics of the agent in a given time window. This set is later constrained to the velocities that are safe with respect to obstacles. A velocity is considered as safe if the agent can stop before the obstacle. In [67], the authors extend this work to dynamic obstacles.

In [68], Fiorini introduced the concept of Velocity Obstacle. The velocity obstacle defines the set of all velocities of the agent that would lead to a collision with another moving obstacle. To avoid collision, the agent should choose a velocity outside this set. The set of non colliding velocity is reduced to all dynamically reachable velocities. By ignoring the decision capability of the other agent, velocity obstacle collision avoidance lead to oscillations. In [69], the authors propose a Reciprocal Velocity Obstacle approach to overcome this issue. They assume that all agents have the same collision avoidance strategy where each of the possibly colliding agents are responsible for half of the avoidance maneuver. In [70], they show a sufficient condition to ensure collision free navigation between multi-agents and called their approach Optimal Reciprocal Collision Avoidance (ORCA). They later improve their approach by considering acceleration constraints [71] and uncertainties in agent's dynamics and sensing [72]. They showed collision avoidance in simulation for 128 quadrotors in 3D [72]. This strategy has since been extended to deal with accelerations limits [71], arbitrary linear dynamics [72], non-holonomic robots [73] or enforcing C^n continuous control sequences [74].

Such approaches are appealing even if these strategies assume collaboration of other vehicles and could fail to provide a collision-free trajectory if a vehicle do not collaborate following a sensor failure for instance. This particular case could be handled separately.

1.2.3.5 Numerical optimization

In [75], a neural network is used to perform collision avoidance with dynamic environment without learning. Their approach is purely reactive and is therefore computationally efficient. This approach seems appealing at first sight but the hidden mechanism of the controller where it is difficult to understand exactly what is happening, makes it indefensible in front of eventual passengers. In addition, if the problem becomes more complicated such as in the myCopter scenario, the convergence of the optimization process of the neural network is not guaranteed. If no solution is found, there is no way to understand why and choose a new direction of search. Therefore, neural network approaches have too many drawbacks to be used in the myCopter scenario.

In [76], the authors compare three optimization algorithms: a genetic algorithm (GA), a

differential evolution algorithm (DE) and a particle swarm optimization algorithm (PSO). Each algorithm is performed as online planner, minimizing the deviations from the initial path and avoiding loss of separation. They test the three approaches on two different scenarios. The algorithms were stopped after 120′000 evaluations of the cost function. In these scenarios, the PSO performances are below the two others especially when the number of agents increases.

The main drawback of these strategies is the need to perform a large number of evaluation of the cost function. In a dynamic environment, the constant situation changes force to restart very often the computation. In addition, the time to react is small and does not give the time to the strategy to converge and give a result. Therefore, such strategies will fail to give a solution before reactive strategies.

1.2.4 Collective behavior

To alleviate the challenge of reactive decentralized collision avoidance strategy, we take inspiration from Nature. Evolution came up with a simple solution to navigate safely in dense environments. In Nature, flocks of birds [77], school of fishes [78], swarms of insects [79], human crowds [80] and bacteria [81] are examples of collective behaviors emerging from local control and local sensing. As stated in [82], it can be observed that large-scale behaviors coming from self-organization process emerge spontaneously from only local interaction and local sensing.

In this section, flying collective behaviors is first presented, then human collective behavior, i.e. crowds.

1.2.4.1 Flying collective behaviors

In flocks of birds, emerging global behaviors can be observed from local rules and local knowledge of the environment without any central entity deciding for the whole.

Flying in groups by decentralized control strategies can be divided in three main formation structures: the Leader-Follower (or Leader-Wingman) approach [83], [84], the Virtual leader approach [85] and the Behavior approach [86]–[88].

In the Leader-Wingman approach, one agent is defined as being the leader of the formation and all other agents are followers and are required to maintain a given distance and position with respect to the leader and neighbor agents. In such approach, each agent should be able to track precisely their own position as well as leader position and intent in order to maintain a rigid formation. This pseudo-rigidity makes collision avoidance a hard task especially in very dense environments as the formation has to react as a whole and error propagation makes rear agents react poorly to new dynamic threats.

In the Virtual leader approach, each agent tracks the same virtual leader that can either

be a real agent or a virtual point. Agents are not required to maintain a relative distance to neighbor agents and thus overcome the drawback of error propagation of the Leader-Wingman approach. Nevertheless, as agents do not track neighbors, they might not be able to avoid collisions. Furthermore, a central entity should designate the leader as well as provide the trajectory of the Virtual Leader leading to communication issues as well as decision processes. The needs of central entity and communication are major drawbacks of such approaches.

Behavior approaches are inspired from Nature where flocks of birds tend to position themselves only by "sensing" local neighbors. Agents tend to position themselves with respect to local neighbors and recreate a given formation geometry. This approach doesn't suffer from single point of failure and is able to cope with dynamic changes of the environment.

Swarm intelligence has been widely used in Robotics. Thanks to scalability and redundancy, low cost platforms can be used together to perform higher level tasks such as establishing communication network [89], [90], formation flying (leader-follower approach) [91], migration [92] or sensor network [93].

However, all approaches tend to fulfill common higher level tasks for the whole swarm and do not tackle the problem of agents with individual goal as this is the case with PAVs flying each to its own defined destination.

1.2.4.2 Crowds

Humans moving in a crowd is a good example of navigation in very dense environments with individual goals.

Crowd simulation models are separated in two categories: macroscopic and microscopic point of view. In the macroscopic point of view, the agents are aggregated and controlled by a higher central entity. Such approaches are designed for simulation purposes only and are therefore not suitable for our problem. Microscopic point of view computes trajectories and collision avoidance in a decentralized manner. State-of-the-art in microscopic crowd simulation is presented in this section.

In [94], a concept of social force is introduced for crowd simulation. Agents are modeled as Newtonian particles navigating towards their own goal. Each agent is under the influence of two kinds of virtual forces named social forces: an attractive force tends to move the agent toward its goal and a repulsive force from other agents located below a maximal influence distance. Experimental psychological studies proved that potential based approaches can modeled human collision avoidance with good fidelity [95], [96]. Upgraded versions of this work exist [80], [97], [98]. However, this approach suffer from the same drawback as the Fields strategies presented in Section 1.2.3.3 i.e. it is an incomplete approach that deals poorly with dynamic constraints of the agents.

In [99], the Principle of Least Effort is used to compute movement of agents in large crowds.

The Principle of Least Effort states that individuals tend to move towards their goal choosing the mean that lead to the least amount of perceived effort. This Principle is modeled as an optimization method to compute a biomechanically energy-efficient path. Each agent tries to minimize its total biomechanical energy which is related to the agent's velocity i.e. each agent tries to keep the velocity that has the least cost.

In [100], navigation of agents in a crowd is divided in two steps: first each agent computes their path to their goal by using a global planner building a roadmap of the environment, then at each time step, agents compute their preferred velocity in order to move towards their goal and Reciprocal Velocity Obstacle (RVO) [69] strategy is used to avoid collisions with neighbor agents.

RVO is a geometric based model, which computes the solution based on the geometric relationship between the agents. The interested reader can find a detailed analysis of geometric based models in Section 1.2.3.4. RVO solves the problem in the 3D velocity space instead of considering the 3D space. It first computes the velocity obstacle set, defined by all velocities of the agent that would lead to a collision with another moving obstacle. To avoid collision, the vehicle should choose a velocity outside this set. By ignoring the decision capability of other agents, velocity obstacle collision avoidance would lead to oscillations. Therefore, RVO assumes that each of the agents uses the same strategy and split the avoidance maneuver equally between the agents. Later, in [70], they show a sufficient condition to ensure collision free navigation between multi-agents and called their approach Optimal Reciprocal Collision Avoidance (ORCA).

Such approaches can be adapted to our scenario where agents want to optimize their comfort i.e. minimize the amplitude and the number of accelerations.

1.2.5 Human-swarm interaction

Studies in Human-Swarm interaction aimed to decrease the workload of the operator, focuses on the accomplishment of a particular task (e.g. radiation source localization [101] or foraging). However, they offer little discussion of more general requirements (e.g. monitor, control and safety) that are not task specific. Furthermore, these promising schemes were always tested in controlled environments such as indoor experiment rooms or in simulation [102], [103].

This drives the need for an analysis of the requirements for a Ground Control Interface (GCI) in order to be used to monitor, control and guarantee safety in experiments with multiple drones in an uncontrolled environment.

Building and exploiting a swarm of drones is a challenging task. One require a drone that was specially designed such that it require a minimal time to mount, (re-)program it and repair it. The design should encompass not only the mechanical parts but also the software part. In this thesis, we focus on the software development side and propose an autopilot framework that is tailored for multi-drone research experiments.

1.3 Main contribution and thesis organization

The main novelty of this thesis is the use of a reactive decentralized collision avoidance strategy to incorporate the physiological comfort for passengers traveling in Personal Aerial Vehicles instead of a path planning strategy. Our approach is inspired by crowd modeling, where multiple decision-making entities travel in dense environments, having each its individual goal. In addition, a framework was developed to enable outdoor experiments with multiple small drones. This framework is composed of an autopilot and a ground control interface, both tailored for easy multi-robot operations. The collision-free strategy was validated through extensive simulations as well as realistic outdoor multi-drone flight tests using the proposed framework.

The thesis is organized as follows:

- Chapter 2 presents the design of an autopilot framework that was developed around the robot used to perform mid-air collision avoidance in outdoor environments. The framework tends to speed up the theory to experiment iterations by using an architecture that can be rapidly and easily customized to the user/developer needs. Design choices are validated using different application scenarios. Several projects using the framework are finally presented.
- Chapter 3 presents first the set of features that a Ground Control Interface (GCI) must incorporate to allow monitoring, control and safety of outdoor missions with a swarm of Small Drones (drones of less than $1kg$). Then, the extension of a widely used GCI is presented, incorporating those features and its usage is demonstrated on a swarm of 10 Small Drones flying outdoor. We compare the influence of deployment and landing strategies on the maximal size of a swarm of Small Drones, given their battery endurance and the expected mission duration. Similarly, we compare the maximal mission time, given a fixed number of robots and their battery endurance. In both cases, we show that a better performance can be achieved using our GCI.
- Chapter 4 presents a reactive decentralized collision avoidance strategy that incorporates passenger physiological comfort based on the Optimal Reciprocal Collision Avoidance strategy [70]. We study in simulation the effects of increasing PAV densities on the level of comfort, on the relative flight time and on the number of collisions per flight hour and demonstrate that our strategy reduces collision risk for platforms with limited dynamic range. Finally, we validate our strategy with a swarm of 10 quadcopters flying outdoors using the autopilot and the ground control interface described in the previous chapters.

1.4 Publications

Part of the work presented in this thesis was published in the following publications:

- N. Dousse, G. Heitz and D. Floreano, "Extension of a Ground Control Interface for Swarms of Small Drones", in *Artificial Life and Robotics*, vol. 21, pp. 308–316, 2016.
- N. Dousse, G. Heitz, F. Schill and D. Floreano, "Human-Comfortable Collision Free Navigation for Personal Aerial Vehicles", in *IEEE Robotics and Automation Letters*. vol. 2, no. 1, pp. 358–365, 2017.

Following paper is under submission for possible publication:

- N. Dousse, J. Lecoeur, B. Huber ,G. Heitz, F. Schill and D. Floreano, "MAV'RIC, a modular framework for research on drones", in *International Journal of Micro Air Vehicles. Under submission.*

2 MAV'RIC, a modular framework for research on drones

WE present MAV'RIC, an open-software and open-hardware framework aimed at fast prototyping for research application on Small Drones. We present the software architecture, which allows fast theory to experiment iterations. The framework uses a modular approach that allows easy and fast configurability. We provide multiple validation examples of our design choices. We explain how the framework was used successfully in several research projects. This framework will be then used in Chapter 4 to validate the proposed collision avoidance strategy.

This chapter is based on the following publication:

N. Dousse, J. Lecoeur, B. Huber, G. Heitz, F. Schill and D. Floreano. "MAV'RIC, a modular framework for research on drones". *Under submission.*

2.1 Introduction

Small drones are becoming more and more widespread in commercial applications such as aerial mapping, cinematography, inspection of buildings, search and rescue, package delivery or leisure flight.[104]

With an increasing ease of use, small size, smaller prize, and high flight speed, drones are often preferred in situations that previously required manned aircraft. For aerial mapping and filming, they can be deployed over remote places for a fraction of the cost of a piloted aircraft. In the context of search and rescue, drones extend the field of view of rescuers, and quickly provide them with situational awareness in emergency scenarios. This allows to detect victims to rescue in avenues where it would be dangerous to send people. Researchers provide drones with new capabilities on a regular basis. These new capabilities are on the one hand enhanced autonomy [105] and on the other hand clever mechanical design. [106], [107] This leads to increasing commercial applications for these small drones. For instance, industrial inspection of pipes and chimney is now done by drones, whose mechanical design was specially adapted to flight in clutter environments.

The enhanced autonomy allows now a single operator to monitor, control and operate a swam of robots, leading to a more efficient and a faster completion of tasks [108] such as mapping, establishment of ad-hoc communication network [109] or following ground user for filming.

However important challenges are still to be tackled. As drones become more autonomous and perform actions without explicit manual commands from the operator, more autonomous features need to be developed in order to increase the safety of drones. Drones should be capable of navigating in unknown environments and detecting and avoiding static and dynamic obstacles autonomously.

In order to operate any type of drone, a minimal set of features is required. In addition to the mechanical and electronic parts, this set is composed of an autopilot, a ground control station and a communication protocol to monitor and control the drone. In this work, we understand the autopilot as being solely the software running on an electronic board. In contrary to the ground control station and the communication protocol, the autopilot is highly dependent on the type of drone and on its application and is the main focus of this work. If the user wants to adapt the code, going for instance from a quadrotor to a fixed wing or to a multi-modal drone capable of performing multiple different locomotion modes, he will have to change entirely the flight controller. Therefore, having an autopilot where the flight controller can be easily replaced by changing a single block of code is an advantage that shorten the time to adapt the autopilot to its new drone.

Different operational requirements imply different software features. It can be challenging to accommodate every potential requirement in a single, ready-to-use, monolithic autopilot even if most of the code will be reused For instance, the requirements for a vision based collision avoidance strategy differ much from the ones for a autonomous outdoor navigation

strategy. In the first case, the autopilot will require a large computing power as in the second case, data fusion between different sensors will be the main focus of the code. Therefore, having an autopilot software that can run both on a lightweight micro-controller or a more powerful Linux-based computer is an advantage that shorten the time required to prepare an experiment and collect data. The autopilot should allow fast and easily customization depending on the user/developer needs.

Code sharing between users and developers is one of the tools that allows faster theory to experiment iterations. By sharing code, researchers avoid multiple implementations of the same feature, and multiple corrections of the same bugs.

In conclusion, there is a strong need for reliable and ad-hoc autopilots that allow fast prototyping and reliable experimentation.

In this work, we present the MAV'RIC framework, an open-source software library to build autopilots for small drones. MAV'RIC framework is designed with a focus on modularity, configurability and explicitness of the code. It is aimed for researchers requiring a large freedom to meet their specific requirements. Even if MAV'RIC is primarily a software library, it provides ready-to-use sample drone projects. It guaranties a clear separation between library code and project code. Library code is shared between research projects, and project code is specific to a particular application. As a consequence, the researcher can write his or her own application by importing parts of the library in the project code without interfering with other researchers. In addition, he or she can develop new features and incorporate them into the library such that the whole group benefits from the development.

This chapter is organized as follows. In Section 2.2, the potential design choices are presented. Section 2.3 presents the design principles of the MAV'RIC framework. In Section 2.4, selected modules of the MAV'RIC framework are presented, which follow the design principles. In Section 2.5, we show multiple validation examples of our design choices. User guidance to use the framework are given in Section 2.6. In Section 2.7, a quadrotor designed for multi-drone experiments is presented. Finally, concluding remarks are given in Section 2.8.

2.2 State-of-the-art

Many autopilot softwares are now available both in form of open source projects and commercial software. Here, we will focus on open source software, since access to the code is an important feature for research platforms as well as education projects. Being able to view and modify the code allows the user to adapt the project to unconventional platforms and to fully control the vehicle beyond basic functionalities. In this section, we first present the history behind this project. As the aim of this work is to propose a different new autopilot framework, we deliberately avoid the comparison with existing frameworks as choices in comparison criteria would be subjective. We will list requirements of an autopilot for research and education platforms and present the solutions implemented by existing projects.

2.2.1 History of the project

This project was started at the beginning of the myCopter project, which was presented in Section 1.1.1. At that time, open-source autopilot frameworks for Small Drones were not as popular and reliable as today. Therefore, it was decided at that time to go for our own solution. After some time, there were some strong needs in the lab for an autopilot that everyone could use. The framework was therefore extended and refined to meet all the different requirements of the lab members. This lead to the development of the framework presented here. Some of the projects that used this framework are presented here below, in Section 2.5.5.

2.2.2 Supported Vehicle Types and Configurations

Many different conventional and unconventional autonomous mobile robots are developed in academia. To facilitate the reuse and exchange of source code between different robots and scenarios, an autopilot should be easily adaptable to different configurations. This includes different ways of locomotion, different methods of control and different electronic hardware, including sensors, actuators as well as the computational unit. Many existing platforms focus on fixed-wing and multicopter platforms, while others keep their software as generic as possible.

The Paparazzi UAV autopilot software [110] allows the user to configure the autopilot to his platform through a XML file describing the robot. This files contains information about the electronic components as well as their configuration, allowing the user to adapt to a specific platform without modifying code.

The ArduPilot framework has four different projects for different: ArduCopter for multicopter and helicopters, ArduPlane for fixed wing aircrafts, APMrover2 for ground vehicles and boats and AntennaTracker for antenna trackers. While there is an effort to share elements between the project, there are considerable differences between vehicle types. Dividing the framework in different projects has the advantage of the code being separated for different vehicles and hence being less complex.

The PX4 autopilot framework has a unified code base allowing to seamlessly port code between platforms. A new airframe can easily be adding a new configuration file which can contain changes such as using different modules, parameter values or sensors.

2.2.3 Supported Computational Hardware

Autopilot boards with different levels of complexity are available. Boards range from powerful multi-core computers capable of performing complex high level tasks such as image processing to simple micro controller boards. Many autopilot software were originally designed for specific boards. However, many of them were ported to autopilot boards and even to off the shelf end products such as Parrot's Bebop quadrocopter. While most boards share the

same functionalities and similar components, a main difference is the size of the board as well as the number of connectors and types of communication channels. In addition to the board running the stabilization, many autopilots offer the possibility to connect companion computers where high-level code is executed. Running high-level and low-level software on separate chips increases the safety since in case of a problem in the high-level software, the low-level software can continue to run.

2.2.4 Operating Systems

On one hand there are autopilots that run on a real-time operating system (RTOS) while others run on a bare metal, i.e., without an operating system. An RTOS can handle multiple concurrent threads, allowing different tasks to be executed concurrently. Other typical features of RTOSes include advanced file management and convenient means of remote interaction during runtime and separation of tasks into standalone applications. RTOS features typically come at the cost of increased code size and complexity. The PX4 [111] autopilot runs on a NuttX RTOS, giving it convenient features, such as readily available drivers, scheduling and resource sharing.

2.2.5 Scheduler

Many tasks need to be executed on a autopilot, such as stabilizing the vehicle, communicating with a ground control station or a remote control, and handling a mission. While some tasks need to be executed occasionally, many tasks need to be run periodically. A scheduler is managing when to run which tasks. Schedulers are either cooperative or preemptive. In cooperative scheduling, a task is run until itself decides to stop. In preemptive scheduling, the execution of a task can be interrupted either by an external signal, such as a sensor reading or by the scheduler. Hence, preemptive scheduling allows for longer tasks to be run without the risk of delaying time critical tasks such as the stabilization loop. However, this adds complexity to the system and requires careful implementation of resource sharing to avoid problems due to concurrent access to data from different tasks. RTOS based autopilots typically take advantage of the systems underlying preemptive scheduler.

2.2.6 Abstraction and Modularity

When modifying an autopilot software, it is important for developers to contain the scope of the modifications. Modular software design approaches allow to exchange or rewrite only the necessary modules and leave the rest of the code with as little changes as possible. In addition to allowing faster modifications, this also means that the customized code and the original code will remain similar. Hence, porting new features or bug fixes from the original code to the customized code is simpler. A modular design allows for easier porting of the code to different platforms.

An important abstraction principle is the separation between hardware dependent and independent code. The hardware dependent code containing drivers for different components such as sensors, actuators and buses is commonly referred to as Hardware Abstraction Layer (HAL). Having a common interface for drivers allows to keep most of the code hardware independent. Autopilots that rely on OS or RTOS may use an HAL that is provided by the operating system, it is the case for PX4 which uses the HAL of NuttX. Other projects provide their own HAL implementations, for example Paparazzi.

In addition to separate hardware dependent and independent code, separation of the software into different modules such as attitude control, mission planning and communication. Having predefined interfaces for such modules allows to replace them with custom modules. A standard way to exchange modules is by directly modifying the code that uses the module. The Paparazzi autopilot allows exchanging modules by modifying a configuration XML file. The chosen modules are then incorporated into the source code. The PX4 autopilot runs modules as separate applications. This allows to start, stop and exchange modules even in flight. The different modules use an asynchronous publish and subscribe communication protocol called uORB to exchange information such as sensor measurements.

Using software paradigms such as inheritance and templates allow to reuse a maximum of code for different modules.

2.2.7 Communication

Communication with a user or a software on the ground is an important feature for almost all UAV projects. It may be used to send commands such as waypoints from the ground to the drone or for the drone to send information about its state to the ground. A commonly used protocol is the MAVLINK protocol, a lightweight marshaling library for micro aerial vehicles, developed in relation with PX4.[112] This communication protocol offers a large number of commands and messages. It is supported by many autopilots and ground control stations. Using a widely used protocol allows autopilots to use various existing tools, such as ground control stations and message relaying tools. While many autopilots are based on MAVLINK to benefit of various tools developed for this protocol, some autopilots have defined their own protocol. This allows them to have communication tailored to their needs. Many autopilots can be connected to an onboard computer that is used for high-level tasks such as image-processing or mapping of the environment. Such companion computers typically use the same protocol to communicate with the autopilot. In a multi drone scenario, additional communication might be required between drones. Interdrone communication might use the same channel and protocol as the communication to the ground.

2.2.8 Simulation

Simulations allow features or settings to be tested without the risk of damage. Furthermore, preliminary tests can be executed for many settings and under many environmental conditions. Different simulation setups have been proposed and implemented. Most autopilots provide a software simulation where both the autopilot code as well as the physics simulation are run on a desktop computer.

To test the code directly on the target hardware, many available autopilots offer a Hardware In the Loop (HIL) simulation. In this configuration, the physics simulation is run on a computer while the autopilot code is run on the target hardware. Measurements of simulated sensors are sent from the simulation to the autopilot and motor commands are sent in the other direction.

2.2.9 Support & Collaboration

Many autopilot frameworks have an active community of researchers, companies, and hobbyists that contribute improvements, bug fixes and new features. To fully benefit from their communities, large projects require a good organizational structure as well as suitable collaboration tools. A common collaboration platform is Github. It allows to exchange source code, discuss problem and features and to track changes. Having many supporters and developers allows for fast feature development as well as fast debugging. ArduPilot, for instance, has over 250 registered contributors on Github.

2.3 Design principles

In this section, we describe the architecture of the MAV'RIC software framework. This framework is a library which provides the user with modules that can be linked together and configured in the project specific user code. This allows to quickly create an autopilot tailored to the user's needs without modifying the library. In addition, any further development and improvement can be directly inserted in the code by every developer via e.g. a version control software such as git.

2.3.1 Modules

The software is organized in different modules. Every part of the code is a module implemented as a class, from low level drivers (e.g. UART, I^2C) to high level logics (e.g. state estimation, PID controller). The code of each module is consistent and is composed of four standard elements:

- A class constructor with explicit dependencies and a configuration structure given with a default configuration.
- An initialization function, initializing all variables values of the module.

- A configuration structure, encapsulating all the configuration values of the module.
- An update function: being the core of the module and performing the aim of the module (e.g. reading sensor value, estimating the position of the robot).

In order to use a specific module, the user follows three steps.

1. First, the constructor is called in the constructor of the user specific code with the default or a user defined configuration structure and with the explicit dependencies of the module.
2. Then, the initialization function is called.
3. Finally, the update function is added as task in the scheduler.

The modularity provides easiness of use. It is indeed fast to modify and/or to replace any module. One just has to repeat the three steps previously explained in order to replace any module by another one. The configuration of any module can be modified in the user specific code and does not interfere with other users configurations. Furthermore, explicit dependencies between modules and predefined module code structure facilitate the understanding, the cleanness and the safety of the code.

2.3.2 Abstraction Layers

The architecture of the MAV'RIC framework is presented here below. It is composed of five layers (Fig. 2.1), which are presented from top to bottom.

The autopilot layer is responsible for all the high level logical components such as state estimation or navigation.

The sensor and actuator interface layer is an abstract interface for sensors and actuators. This layer is therefore not specific to the sensor or the hardware used.

The driver layer implements different sensors from its abstract interface. This layer is sensor specific but the drivers could be used on different hardware architectures.

The Hardware Abstraction Layer (HAL) layer is an abstract interface for low-level drivers linking the drivers with the peripheral layer. It is hardware specific.

The peripheral driver layer implements the low level protocols (e.g. I^2C, PWM) to communicate with sensors and actuators. For each peripheral driver, there is a dummy version of it that can be used during an earlier development phase to test different parts with the whole code compiling.

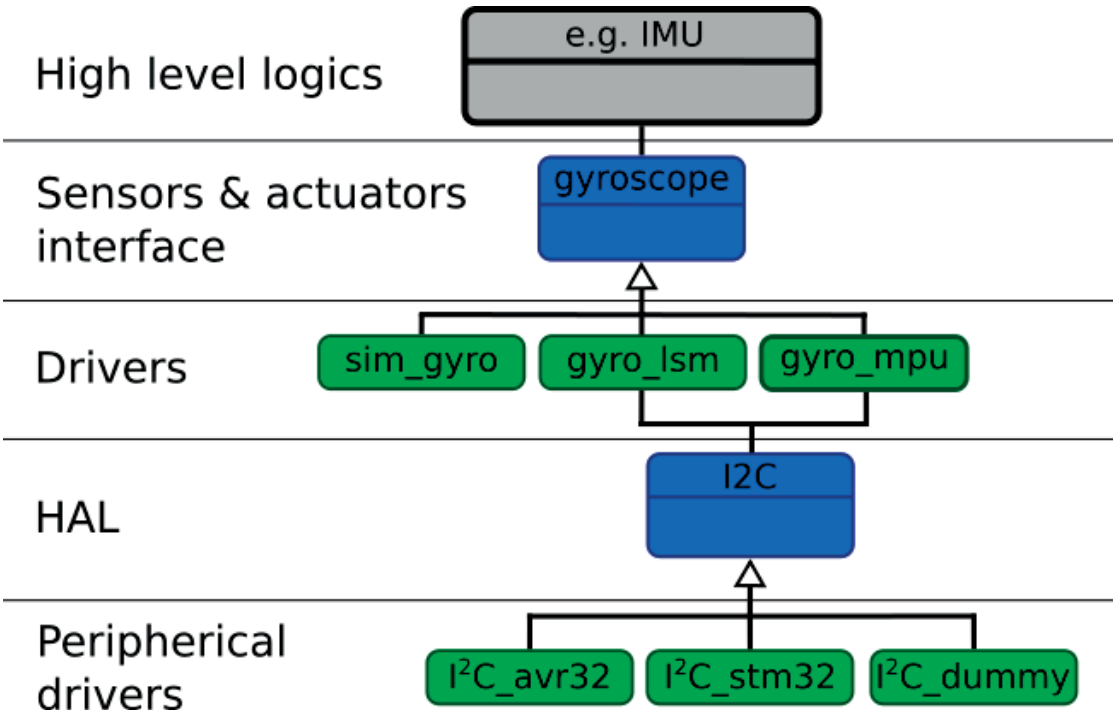


Figure 2.1 – MAV'RIC five layer architecture, example for the gyroscope driver. The autopilot has a gyroscope interface through its Inertial Measurement Unit (IMU) structure. Depending on the board, this interface is linked to one of the implementation of a gyroscope sensor (simulated, LSM330DLC or MPC6050). The real sensors require the I^2C communication protocol to communicate with the microprocessor. This I^2C protocol is implemented as an interface. Each microcontroller type has its own I^2C implementation. We provide an implementation for the AVR32 and STM32 architecture as well as a dummy implementation for earlier development phases.

This layered architecture ensures a high code reusability as the high level code is invariant of the hardware used. In addition, development for new architecture is eased with the use of dummy versions for peripheral drivers.

2.3.3 Architecture of MAV'RIC based projects

As the framework is mainly a library, the user needs to link the desired modules together to create a project. A project is composed of two main components, namely a board and a vehicle (Fig. 2.2).

First, a board is created with all its hardware components (peripherals & sensors) that expose high level interfaces for sensors and actuators. Second, the vehicle is created by assembling modules for high level functionality (state estimation, communication & telemetry, navigation & control, task scheduler). Modules are created by the constructor of the board/vehicle. Then, the initialization function of the board and of the vehicle initialize every module with their

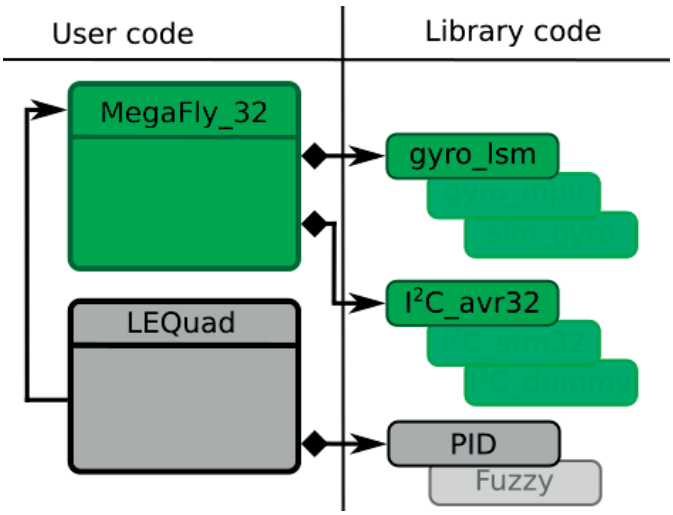


Figure 2.2 – Link between library code and user code. The module “MegaFly_32” is the board component and the module “LEQuad” is the vehicle component of this project. The board module is using the LSM330DLC gyroscope and the I^2C driver for the AVR32 architecture. The vehicle module is using a PID controller and is linked to the board module via its interfaces.

desired configuration. Then, they add the downlink telemetry callback functions as well as the variables that will be recorded by the onboard logger. Finally, the module's task is added to the scheduler.

After the initialization, the main loop is only composed of the update function of the scheduler. The scheduler is then solely responsible to call every module at the correct frequency.

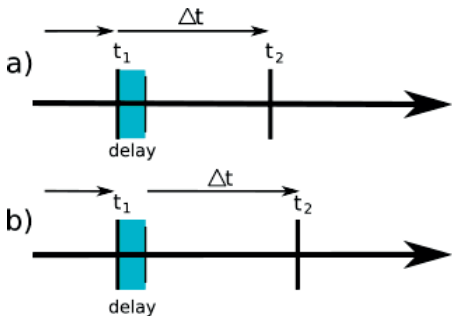


Figure 2.3 – The two timing modes. *a*) absolute timing mode, the next execution of the task is due starting from the previous scheduled time. *b*) relative timing mode, the next execution of the task is due from the effective starting execution time.

2.4 MAV'RIC Software Framework

We present here a list of selected modules that are available in the library and describe our design choices as well as configuration options for the user.

2.4.1 Scheduler

Instead of relying on a complex real-time operating system, the MAV'RIC framework uses a simple task scheduler with two possible scheduling schemes: a priority-based and a round Robin scheme. The priority-based scheme will execute first the tasks with a higher priority. The round Robin scheme will simply execute one task after the other regardless of the priority.

The scheduler loop runs as fast as possible and executes each task at a specific time interval. If multiple tasks should be executed at the same time, the order depends on the selected scheduling scheme.

In addition to the priority scheme, the user can choose between two different task timing modes. It can have an absolute period or a relative period timing mode (Fig. 2.3).

The absolute period timing mode computes the next time at which the task should be executed from the previous requested execution time. The delay in the execution of the task has no influence on the next execution time. However, if two tasks are due at the same time but one has higher priority or is scheduled before in the tasks list, it will always be executed first and the second one will always be delayed from at least the execution time of the first task.

The relative period timing mode computes the next execution time by taking the actual time at which the task started to be executed the last time. With this timing mode, the execution time of these tasks tend to scatter in time to minimize the overlap between two tasks. Indeed, if two tasks are scheduled at the same time, one will be delayed from at least the execution time of the other and the next execution time will be shifted accordingly. The main drawback of this timing mode is that it cannot be guaranteed that a task with low priority will not be

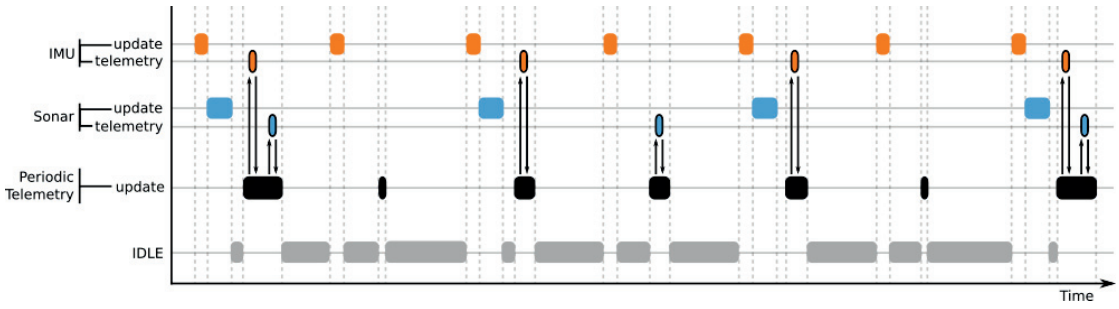


Figure 2.4 – The periodic telemetry task is called at constant time interval. It polls in its registered messages the ones that are due at that time and sends the specified message. The execution time of this task can vary a lot depending on the different update frequencies of the messages.

delayed often and that its actual frequency will not be lower than the expected one.

We present the performance of this scheduler in the Validation section. Choosing the right scheduler depends on the objectives of the user and is always a trade-off between complexity and performance. We show that this simple scheduling system can reach satisfying results while keeping the complexity quite low.

2.4.2 Communication

The open-source protocol MAVLink is used to exchange messages with the ground control station, other drones, or with an embedded companion computer. Using MAVLink as communication protocol offers compatibility of the MAV'RIC platform with widely used ground control station software such as QGroundControl and Mission Planner, as well as with scripting tools like DroneKit. MAVLink protocol also allows the creation of new message types to address custom needs, which fits perfectly our design objectives.

The MAV'RIC software framework provides users with a set of tools built on top the MAVLink protocol to handle communication. The communication subsystem is divided into three main modules: Periodic Telemetry, Message Callbacks, and Onboard Parameters.

The Periodic Telemetry module deals with messages sent periodically by the drone. Every module of the autopilot can use the Periodic Telemetry module to register a message. This is mostly used to inform an operator on the ground about the state of the UAV. For instance, as shown on Fig. 2.4, a module implementing an attitude estimation filter provides the Periodic Telemetry module with a function that fills a MAVLink message of type ATTITUDE_QUATERNION, along with the period at which the message will be sent. After this initialization step, the message is registered and configured to send telemetry messages with specific periodicity.

Nevertheless, the telemetry & communication part of each module is separated from the base module such that MAV'RIC framework could be interfaced with any communication protocol

or even without any communication at all.

We use callbacks features to send and receive messages from other entities of the system (e.g. the ground control station or other robots). The autopilot can register a callback to read as well as to send any MAVLink message. These callback can also be configured in order to filter out message that are for other robots or to read every incoming messages. Registered messages can be activated or deactivated on the go such that they could be either activated in a later phase of the flight or in case of problems or deactivated to reduce the throughput on the communication bandwidth. The update frequency of each message can also be adapted in flight. It is therefore not required to reprogram the autopilot each time the telemetry settings needs to be modified.

Onboard parameters are parameters which value can be modified in real-time. These parameters are registered for each module in the user code. Their current value can be gathered or changed by the operator by sending a MAVLink message.

To conclude, this module is a generic tool for communication used throughout the library and configured through user code.

2.4.3 State Estimation

The state estimation module estimates the 6 degrees of freedom of the robot, namely the attitude (i.e. orientation around the center of mass) and the position (i.e. location of the center of mass in space). The state estimation is separated into two different estimators, one for the attitude estimation and one for the position estimation. Different mathematical filters can be used to estimate the state of the robot, some of which are already available in the library code and briefly described here below.

The attitude of the robot is represented as a quaternion.[113] To estimate the attitude, the filter can fuse sensor values from the gyroscope, the accelerometer and the magnetometer. Two filters are currently available in the library, namely a complementary filter and an extended Kalman filter. As for all modules, these two modules can be replaced by any other attitude filter (e.g. Madgwick attitude filter [114] also available in the library code but not extensively tested).

In the complementary filter, gyroscope values are integrated over time and corrected by accelerometer and magnetometer measurements using a constant gain. In addition, the drift of the gyroscope is compensated by estimating the gyroscope's bias using the accelerometer and magnetometer measurements. The complementary filter is computationally inexpensive. It provides a good attitude estimation for static applications as well as for slowly moving robots. Indeed, it assumes that the acceleration vector is always equal to gravity. Therefore, a drawback for this filter is that if the measure of the acceleration is too far from gravity, a bad correction could be applied and the filter can drift away from the real value as it uses a constant gain in the correction step. In addition, in the case of a constant lasting centrifugal acceleration, the

acceleration vector will not be aligned with gravity anymore and the orientation estimation will diverge accordingly.

An extended Kalman filter (EKF) is a non-linear version of the standard Kalman filter.[115] Instead of using matrices for state transition, it uses non-linear functions. A 7 dimensional state vector is used in this EKF (i.e. the four-dimensional attitude quaternion and the three-dimensional gyroscope bias). An EKF is computationally more expensive as matrices have to be inverted. In order to avoid a 7x7 matrix inversion, the corrective step is applied sequentially for the accelerometer and the magnetometer such that only a 3x3 matrix inversion is performed at a time. It also allows for different update time for the two sensors. As the Kalman gain is computed for each iteration depending on the measurements, outliers have less impact on the estimation than with the complementary filter.

The module responsible for position estimation estimates both the 3D position and the translational 3D velocity of the robot. Measurements from the accelerometer are interpolated and are then corrected by the measurements of a GPS, a barometer, a down-facing sonar or a motion capture system. An optic-flow based sensor can also be fused to estimate the velocity.

In this module, a complementary filter as well a Kalman filter are available in the library code for position and velocity estimation.

2.4.4 Stabilisation

The stabilisation module is responsible for the control of the 6 degrees of freedom of the platform, namely the orientation and the position. We provide a cascade of controllers composed of multiple layers. All the controllers use the same data structure such that the input and output to every controller are compatible. Therefore, in the cascade controller, the output of one controller is the input of the lower level controller. Depending on the control mode, the cascade can be entered at a different control level (e.g. velocity control, attitude control). Once the cascade is entered, all the lower level controllers are run one after the other.

The cascade controllers provided in the library are both composed of three layers: a velocity controller, an attitude controller and a angular rate controller. One can control a multi-rotor type of drone and the other a fixed wing type of drone.

A working example of the cascade controller can be seen on Fig. 2.5. If the robot is operating in attitude control mode, the input to the cascade will be an attitude command encompassed in a more global common data structure. Then, the attitude closed loop control will output a rate command that will be fed into the angular rate controller. Finally, the angular rate controller will output torque and thrust commands.

Depending on the type of robot, the thrust and torque commands are mapped into motor commands by a matrix. It is therefore rather straightforward to adapt the code to any kind of control scheme (e.g. multi-rotor, fixed wing). The user has only to change the matrix mapping

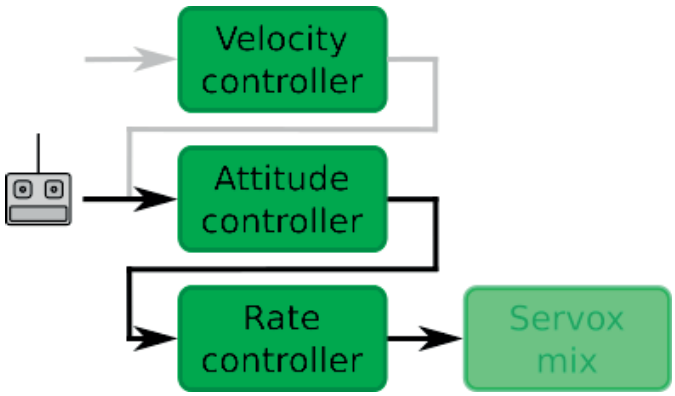


Figure 2.5 – A three layer cascade controller. The current input is an attitude command, coming from a remote controller for instance. The output of the attitude controller is then the input of the rate controller. The output of the rate controller is sent to the servos mix module that takes thrust and torque command and transform it to servos (motors) command depending on the mechanical design of the platform.

the torque and thrust commands to motor commands.

Each controller was implemented using a PID controller. The gain values are defined in the configuration structure of the controller module and can be modified for each type of robot. Users can add, modify or replace any layer of the cascade controller, the type of controller used or even the whole cascade controller.

2.4.5 Mission Planner

The mission planner module is responsible for high level navigation logics. In this work, we understand navigation as being the strategy that drives a robot from its current 3D location to its 3D goal position. Two navigation strategies are available in the library code. First, a "direct to" navigation strategy that drives the robot directly from its current location to its goal location. This strategy is suited for a multi-rotor configuration as it sets a zero velocity when the waypoint is reached. The second strategy is suited for a fixed-wing. It creates 2D Dubin's path to navigate between waypoints while ensuring a maximal turning rate at all time.[116] A Dubin curve is characterized by a starting and an ending position with given directions. Under certain conditions, these two points can be linked by intersecting at most 3 parts that can be either a straight line segment or an arc of circle of given curvature.[117] As for other modules, these two strategies can be replaced by any other navigation or path planning strategies.

In addition, different collision avoidance strategies are available in the library code. All strategies were tested on 10 quadrotors flying autonomously in an outdoor environment. The first strategy is a geometry based method, solving potential collision treats in velocity space.[71] The second strategy is a human-friendly collision avoidance strategy suited for dynamic environments where robots and human share a common airspace.[118] It is based

on a heuristic model originally proposed to model pedestrian behavior. The third strategy is based on modified Reynolds flocking rules, which is a model to perform flocking of a group of robots.[119] The focus here was made in the collision avoidance part as well as robot with individual goals. Finally, the last strategy is a potential-based approach where the shape of the repulsive field is an ellipsoid centered in front of the vehicle in order to better anticipate the future position of the vehicle.[64] The three last strategies were implemented during a Master project.[120]

The framework is compatible with MAVLink waypoint protocol. The user can send, receive and modify any waypoint in a flight plan.

The high level logic encompasses different modules enhancing safety to operate drones.

- *Geographical fences*: two level of geographical fences can be set such that a safety behavior is triggered if the fence is violated. When the robot violates the first fence, it will fly back to its takeoff location and wait for further orders from the drone operator. When it violates the second fence, it will land at its current location.
- *Low voltage*: a threshold on the battery can be set such that below a given voltage, the robot will land at its current location before running out of battery.
- *Loss of connectivity with the base station*: In case of loss of connectivity with the base station during a predefined time interval, the robot will fly back to its takeoff location.
- *Loss of connectivity with the remote*: the robot will fly back to its takeoff location and land there.
- *GPS loss*: if the connection breach is too long, the robot will land at its current location while keeping a leveled attitude as without GPS, the horizontal position estimation will drift apart relatively fast and thus cannot be used to control the robot anymore.

Behaviors when safety is endangered can be specific to the application, the environment and the drone itself. The aim of this section was only to present our design choices based on our experience and needs. They were successfully tested in a multi-rotor configuration as well as in a fixed-wing configuration. These behaviors can be adapted to the user's specific needs.

2.4.6 Sensors

Drivers for multiple sensors are available in the library folder. Each of this driver is implemented using the corresponding abstract interface such that other drivers can be implemented and rapidly integrated in the code. We present here below a list of chips and boards currently supported by MAV'RIC framework.

Inertial Measurement Unit (IMU) is composed of an accelerometer, a gyroscope and a magnetometer. Drivers for the following chips are available: LSM330DLC (accelerometer and gyro-

scope), HMC58831 (magnetometer), MPU6050 (accelerometer and gyroscope) and MPU9250 (accelerometer, gyroscope and magnetometer), The pitot tube driver for the MPXV7002 chip is also available. The GPS driver for U-blox GPS are provided. It is compatible with the UBX communication protocol. The motion capture system OptiTrack can also be used as mean of absolute positioning system. Two barometer chips are supported, the BMP085 and the MS5611. Optic-flow measurements can be gathered from the PX4Flow board. Distance from ground can be measured by a MB1242 I2CXL sonar. Command from the ground can be received via a Spektrum satellite receiver in 10 or 11 bits DSM2 configuration mode. Simulated sensors are also available for the following sensors: accelerometer, gyroscope, magnetometer, GPS, barometer and optic flow. Details of the implementation of the simulation is described in a further section.

2.4.7 Hardware Abstraction Layer

The Hardware Abstraction Layer (HAL) is an interface linking drivers (e.g. accelerometer) with a specific implementation of a hardware feature (e.g. I^2C communication). This interface allows to use the same sensors driver on multiple hardware architectures.

Two physical boards are currently fully supported by the MAV'RIC framework: A first autopilot board called MegaFly_32 (Fig. 2.6) based on an Atmel AVR 32bit processor with 512KB of flash memory and 64kb RAM, floating point unit and DSP instructions, and the commercially available board Sparky[2] based on a STM32 processor. The files required to produce the MegaFly_32 board are available on the GitHub repository. Therefore, low-level interfaces are available for AVR32 and STM32 architectures (serial communication (UART and USB), I^2C, SPI as well as PWM, etc.).

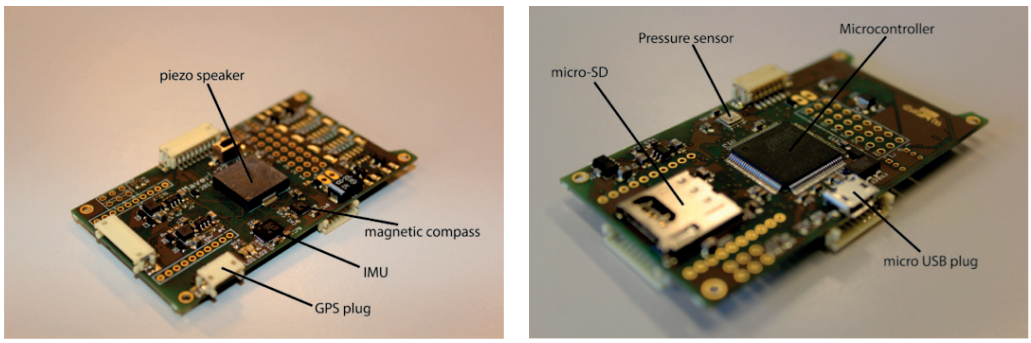
In a future version of the autopilot or if distance between chips increases, UAVCAN will be used as reliability of I^2C communication becomes insufficient at distance over 25cm. Implementation of UAVCAN exists but is not yet open-source available.

2.4.8 Simulated dynamics and sensors

Development and testing of new code is a challenging task. One can insert bugs in the new code, in the interaction between different parts of the code or by neglecting involuntary some combination of factors leading to a unspecified behavior. Simulation can speed up the validation process as it can remove environmental factors such as wind, meteorological conditions or hardware failures. It allows therefore faster and more convenient testing.

In a simulation, simulated sensors can replace physical components, allowing to control environmental factors through a dynamic model. A dynamic model allows to simulate the behavior of a vehicle by applying a set of dynamic equations. The state of the vehicle is

[2] https://github.com/TauLabs/TauLabs/wiki/Sparky2



(a) Top view of the MegaFly_32 board.

(b) Bottom view of the MegaFly_32 board.

Figure 2.6 – The MegaFly_32 autopilot board.

incrementally updated based on the actuator commands given by the autopilot. Simulated sensors convert the simulation state to simulated measurements that are then fed into the autopilot's logic.

The user can conveniently replace one or multiple sensors with its simulated version in the low level user code. Before feeding the autopilot with their measurements, these simulated sensors make sure that the last simulation update step was performed and therefore, that the simulation state is up to date. As a consequence the user does not need to specifically add a task to update the simulation. Therefore, the high level user code as well as the library code will not be affected by a transition from physical to simulated components.

The simulated sensors access the dynamic model through an interface, allowing the user to choose between different dynamic models. The user can select one of the existing models (currently fixed wing or quadrotor vehicles), or implement a new model tailored to the dynamics of his vehicle.

Having an interface for the dynamic model also allows to choose whether the computation of the model is performed within the autopilot application binary, or the performed by an external simulator. In the first case, starting a simulation requires only running a single application, hence not requiring any additional communication between applications or devices. Furthermore, it allows for the simulation to be run directly onboard of the vehicle, as explained in the following section. In the latter case, the implementation of the dynamic model interface only performs communication with an external simulator. This allows to run more complex models at the cost of additional communication.

2.4.9 Simulation on Hardware and on Emulation

The Autopilot can be run either onboard or be emulated on a computer. On Fig. 2.7, we represent the three ways of running the MAV'RIC autopilot. First, the autopilot runs on the robot itself, drives the real motors/servos and reads the measurements of the real sensors.

Second, the Simulation on Hardware (SOH) is represented, where the autopilot runs also on the robot itself, but drives simulated motors/servos and reads the measurements from simulated sensors. SOH is presented in details here below. Finally, the Simulation on Emulation (SOE) is represented, where the autopilot runs on a desktop computer where it drives again simulated motors/servos and reads the measurements from simulated sensors. SOE is presented in details here below as well.

2.4.9.1 Simulation on Hardware

Typically, autopilots implement a hardware in the loop simulation (HIL), where the high level code is run on the board itself and the dynamic simulator is run on a separate desktop computer (e.g. on Gazebo simulator). We propose instead to use Simulation on Hardware (SOH), where the dynamic engine is running directly on the hardware. As opposed to standard HIL, no additional data are required to be sent between the ground control station and the robot. The telemetry module sends the same amount of information from the robot to the ground control station as when it is operating in reality. Hence, no overload of the communication channel occurs and no communication delay are added for the simulation of the dynamics. Only the sensors are replaced by simulated sensors (e.g. accelerometer, gyroscope, GPS) and the motors' effects are simulated in the dynamic engine. It is better to run the code on the hardware itself to ensure equivalence between the simulated and the real behavior in terms of communication.

2.4.9.2 Simulation on Emulation

In the Simulation on Emulation (SOE), the whole code runs on a computer. The communication between the ground control station and the robot is done internally (e.g. via UDP protocol). On computers, a lot of debugging features are available that can speed up the testing of high level modules. The simulation on emulation allows to test these modules faster (no need to go to a test field) and with more debugging features. This simulation mode is merely thought to test high level code such as navigation strategies.

2.5 Validation

2.5.1 Evaluation of real-time scheduler

We measure on the real hardware the performance of the two scheduling schemes as well as the two timing modes on one of the tasks, namely the stabilization task. In a flying robot, the stabilization task is the most important task. If the stabilization loop has too much delay, it cannot be guaranteed that the robot will be stable in the air.

This task is therefore inserted first in the tasks list with the priority set to the highest value. It is thus the first task to be executed in both scheduling schemes. In addition, the mission

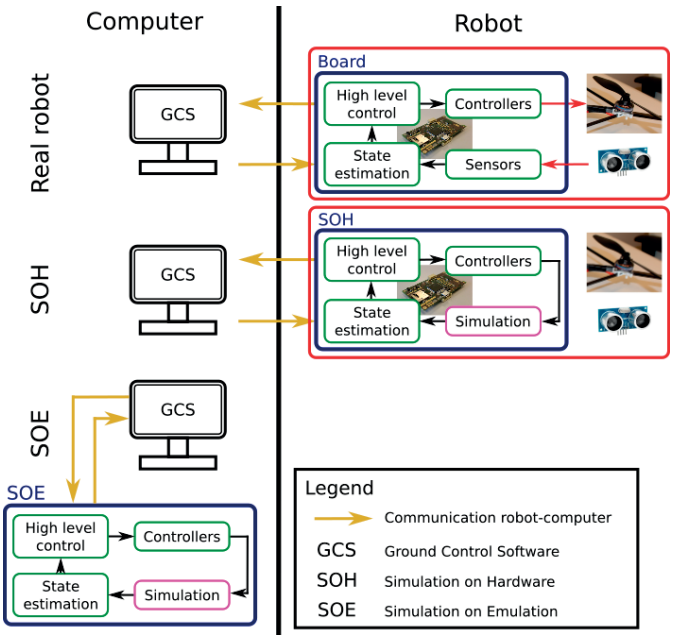


Figure 2.7 – Representation of three different boards. On the top, the robots reads the sensors values and sends command to the motors following some logics. The robot communicates with the Ground Control Station by wireless communication. In the middle, the Simulation on Hardware (SOH) is represented. the simulation modules simulates onboard the dynamic model of the robot and the sensors of the robots. The robot reads these simulated sensors values and sends command to the simulated motors. The robots still communicates with the Ground Control Station by wireless communication. On the bottom, the Simulation on Emulation (SOE) is represented. The whole software runs on the computer. The simulation is still part of the code and simulates the dynamic model as well as the sensors inputs. The simulated robot communicates with the Ground Control Station by UDP packets.

planner task, the state machine update task and the communication task were simultaneously running.

On Fig. 2.8, the median and standard deviation of the execution time for the stabilization task is shown for the two scheduling schemes and the two timing modes. The execution time is not affected by the scheduling scheme. As no concurrent task are running as it would be the case with a RTOS system, tasks are executed until the end. No interruption occurs that would lengthen the execution time of the task.

On Fig. 2.9, the median and standard deviation of the execution delay for the stabilization task is shown for the two scheduling schemes and the two timing modes. The execution delay is small compared to the period of the task, which was set to $4000\mu s$ (i.e. 250 Hz). The task is therefore executed at a rather constant frequency, which is better for the stabilization of the robot.

In all schemes, the performance of the stabilization task are sufficient to have a stable behavior

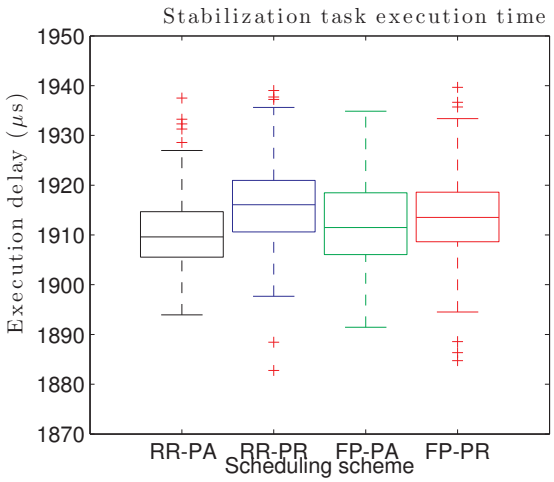


Figure 2.8 – Mean and standard deviation of the execution time of the stabilization task in μs. RR stands for Round robin scheme, FP for fixed priority, PA for periodic absolute and PR for periodic relative.

for the robot, This holds obviously only if the controller gains are tuned properly.

2.5.2 Callback-based Inter-vehicle Communication

To demonstrate the ease of use of the callback-based communication module, we describe in this section the procedure to implement mid-air collision avoidance between multiple drones using the MAV'RIC framework and using wireless data exchange. The procedure of avoiding collisions can be split into two tasks, namely collision detection and collision avoidance. First, the robot should gather information from its neighbors in order to detect whether there is a (or multiple) collision threat(s) or not. Then, if required, the robot should compute a change of trajectory to avoid the potential collision(s).

Using the callback-based communication module previously described, the procedure to enable wireless data exchange is fast. One should register two callbacks: one callback to broadcast the required information (e.g. position, velocity) in a specific message and one callback to listen to the incoming messages from neighbors with the same information, in order to determine whether the neighbor is a threat or not. Then, if required, the strategy can modify the goal velocity accordingly.

In this project, the goal was to implement the Optimal Reciprocal Collision avoidance (ORCA) strategy [71] and perform experiments with 10 quadrotors flying outdoors. ORCA is a velocity based collision avoidance strategy for multiple robots, that requires for the robot to sense periodically the position and the velocity of each neighbor. The trajectory traces of 10 quadrotors perfoming ORCA can be seen in Fig. 2.10.

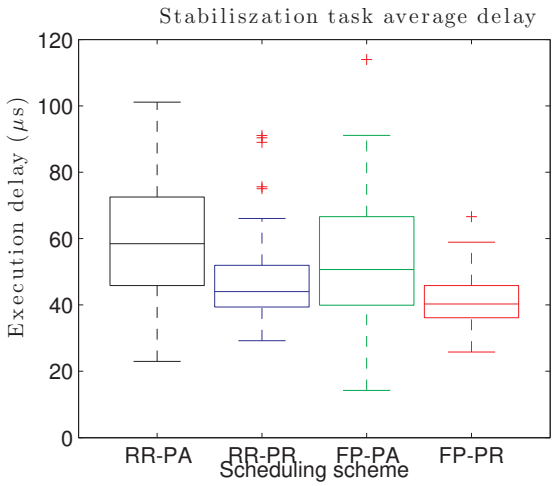


Figure 2.9 – Mean and standard deviation of the execution delay of the stabilization task in μs. RR stands for Round robin scheme, FP for fixed priority, PA for periodic absolute and PR for periodic relative.

2.5.3 Abstraction

We demonstrate here one of the advantages of our modularity by presenting the simple procedure to go from an outdoor flying drone to a drone flying in an indoor environment with a motion capture system (MOCAP). A motion capture system is a set of infra-red cameras that provide fast and precise 3D positioning in a given indoor space.

Using the MAV'RIC framework, the procedure to go from a drone flying outdoor with a GPS to a drone flying indoor with a Motion Capture system has only one step. One should only replace in the user code the real GPS by a GPS_mocap. The GPS module is an abstract class for any kind of GPS. A GPS_mocap module is simply one implementation of this abstract module. The Motion capture systems sends MAVLink messages that are then decoded and used as if it received a real GPS message to estimate its position. The position estimation filter correction gained are adapted accordingly as the precision of a MOCAP is in the order of magnitude of at most the centimeter, opposed to a few meters for a standard GPS.

2.5.4 Safety

Research on drone is currently leaving the unrealistic controlled environment of laboratories and move towards realistic scenarios in outdoor environments. Outdoor multi-drone experiments is a reality.[12], [13], [121] To operate multiple robots in an uncontrolled environment, one should be able to monitor, control and operate the swarm safely.

In [122] and in Chapter 3, we present an extension to an existing Ground Control Interface that allow monitoring, control and safety of outdoor missions with a swarm of Small Drones.

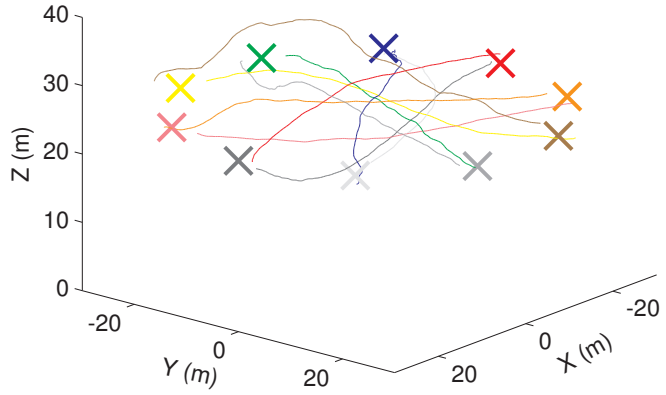


Figure 2.10 – Trajectories of 10 real quadrotors flying outdoor with GPS, crossing a circle while avoiding each other following the ORCA strategy.

Using this interface together with the MAV'RIC framework, we were able to obtain a flight authorization from the Federal Office for Civil Aviation of Germany (LBA) to fly a swarm of robots for a demonstration on a controlled airport in Braunschweig, Germany, with the collaboration, and under the supervision of the Air Traffic Controller. This authorization shows that our approach meets the expectation of the rule makers and that it could be applied in real missions with Small Drones.

2.5.5 Research & Educational Projects

In this Section, we present different research projects using the MAV'RIC framework.

MAV'RIC framework was originally developed for the myCopter project. This European project aimed to enable the technologies for Personal Aerial Transportation Systems. One of the challenges is to ensure safe and comfortable navigation of flying vehicles in dense environments. In order to validate collision avoidance strategies in real world conditions and under real-time constraints, a quadrotor was developed aimed at multi drone outdoor experiments (Fig. 2.11a). We were able to achieve collision avoidance for 10 quadrotors flying outdoors and presented our results in [121]. As presented previously, the callback-based inter-vehicle communication proved to be an easy tool to enable data exchange between drones during the collision detection phase of the collision avoidance.

MAV'RIC framework is also currently used for education. The practicals for the Mobile Robots class at EPFL are using the quadrotor platform. Students learn the steps needed to develop an autopilot, going from sensors calibration, to an attitude controller and finally performing a month long mini-project where they have to design an altitude controller or implement new kind of controllers such as a fuzzy controller. The modularity of the code permitted to

separate the required information for the student during each of the practical from the rest of the code, allowing them to dig into the interesting code more easily and therefore easing the teaching support.

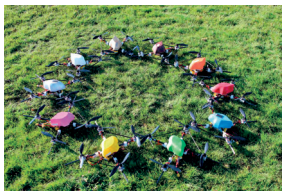
Unconventional flying robots need an easy adaptable autopilot to meet the requirements of their specific platform. The DALER project [106] aimed at developing a multi-modal flying robot capable of flying like a plane, hovering like an multi-rotor and crawling on the ground, using a minimal number of actuators (Fig. 2.11b). The robot used successfully the MAV'RIC framework to performed closed-loop attitude control in hover mode, forward flying mode as well as crawling mode. The major modification of the framework was the mapping of rate control command to actuators command, going from a multi-rotor mapping to a servo mapping.

The framework was then extended to a flying wing performing autonomous flight (Fig. 2.11c). Therefore, the framework can now be used in a multi-rotor configuration as well as in a fixed-wing configuration.

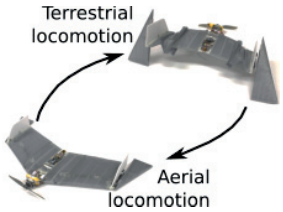
More and more companies are investigating drone carrying goods (e.g. Amazon, Google) as delivery systems. One challenge of such application is to guarantee the safety of the persons interacting with the drone as well as the good itself. The flying pumpkin (Fig. 2.11d) propose an innovative design to alleviate this challenge. It uses a foldable cage that protect on the one hand humans from the propeller and on the other hand, the good by bending and absorbing the energy of the shock. This design leaded to a patent.[123] Once folded, the volume of the drone is reduced by 92% such that it could be carried in a backpack.

Collision avoidance in cluttered and unknown environments is still a challenging task. One can take inspiration from insects, which use optic-flow to avoid collisions, regulate speed and center along corridors. There exist many ways on how optic flow is computed and linked with directional commands. Such methods can be implemented and eventually validated on flying robots. A Master thesis [124] implemented one of this model [125] using the MAV'RIC framework. Two 180 degrees PX4-flow cameras were installed one on each side of the robot. Using optic-flow measurements from these cameras, the robot was able to navigate autonomously in a cluttered and unknown environment, while avoiding collisions with static obstacles[3]. This project raised some interesting issues regarding the use of quadrotors combined with optic flow approaches. As a quadrotor has to tilt in order to move in the horizontal plane, additional optic-flow is measured corresponding to this maneuver. This decreases the interesting optic-flow from nearby obstacles. Therefore, the Hexhog, a hexacopter with inclined motors, was developed to overcome this issue. The inclination of the motors gives the robot the possibility to move along its 3 translational degrees of freedom without modifying the attitude of the robot. Since this platform can navigate without rotational movement, the perceived optic-flow is only due to translational movements. Using this platform, the optic-flow induced by nearby obstacles can be more easily extracted from the measured optic-flow.

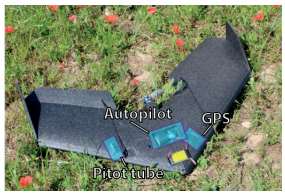
[3] http://actu.epfl.ch/news/a-drone-that-gets-around-obstacles-like-an-insect/



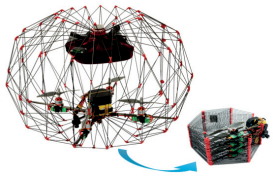
(a) A swarm of 10 quadrotors using the MAV'RIC framework in an outdoor environment.



(b) The multi-modal DALER platform, a robot capable of flying, hovering and crawling using a minimal set of actuators.



(c) A wing using the MAV'RIC framework.



(d) The flying pumpkin. The volume of the folded robot is reduced by 92%.

(e) The quadrotor for optic-flow collision avoidance. It uses two PX4 flow cameras, one facing each side of the robot.

Figure 2.11 – Some of the projects using the MAV'RIC framework.

2.6 User guidance

In this section, we give some guidance for the interested user to start with MAV'RIC framework. The framework is available to the community on GitHub[4]. Developers can clone the repository and can merge their code to the main repository by creating pull requests. Information on how and what to install are available on the project webpage[5]. Steps to set up the framework (e.g. bind your remote controller, connect XBee modules together) are also available at that location. Compiling, flashing and running the software were successfully tested on Linux, Windows and Mac OS X. On Windows, the user can compile and flash the code via Cygwin[6] or Atmel Studio for the MegaFly32 board. On Linux, the user can directly compile and flash the code by the command line and a Makefile specific to his project. On Mac OS X, the user can similarly compile and flash the code by the command line.

2.7 LEQuad

LEQuad is the name of the quadrotor developed at the Laboratory of Intelligent Systems (LIS) of EPFL. It uses the MegaFly_32 autopilot board previously described (Fig. 2.6) and is based on

[4]https://github.com/lis-epfl/MAVRIC
[5]http://lis-epfl.github.io/MAVRIC_Library/
[6]https://www.cygwin.com/

the MAV'RIC framework.

The robot estimates its attitude by fusing data from an Inertial Measurement unit (IMU) composed of a three axis gyroscope, three axis accelerometer and three axis magnetic compass. The 3D position is estimated by fusing data from a barometric pressure sensor, an external GPS receiver as well as a down-facing ultrasonic sensor. The main control loop operates at 250 Hz. Higher-level tasks are executed at lower frequencies depending on requirements.

A small RC receiver is plugged in to allow manual control and safety override (arming and disarming the motors, switching control modes).

The airframe was specifically designed to be simple to build and easy to repair – this was deemed important to be able to construct and maintain a fleet of 10 robots. The design uses standard extruded carbon rods, connected by custom-designed parts that were 3D-printed (selective laser sintering). Two carbon beams are composing the main structure of the robot. They are linked to the autopilot's support via a central piece of compressed open-cell foam to dampen vibrations. Four additional carbon beams link the four motors giving a lateral rigidity to the whole structure. Most parts connect by tight push-fit, except the battery cage and main baseplate which are held together by 8 self-tapping screws. As there are no glued or permanent connections, any part can be replaced quickly if damaged. The ease of construction allowed us to build 10 airframes in 1 day with 2 people, after all parts had arrived.

The quadrotor can be operated under different modes, ranging from little to full autonomy. The first mode is an attitude control mode where the pilot controls the attitude of the quadrotor via a remote controller. When GPS is available, it can be switched to a velocity mode, where the pilot controls the 3D velocity of the quadrotor (translational rate command mode). The pilot can then activate a position hold mode, in which the autopilot will maintain its current 3D position. The last mode is a fully autonomous mode, where the autopilot performs waypoint navigation.

The MAV has a thrust-to-weight ratio exceeding 2:1 and is capable of fast multi-g accelerations and high cruise velocities, which makes it suitable for outdoor operation. For autonomous navigation in our experiments, we limited the horizontal cruise velocity to 3 meters/second and the vertical rate to 1.5 meters/second. Note that the MAV will maintain the direction of the commanded 3D velocity vector but adjust the magnitude to stay within these limits.

2.8 Conclusion

We presented the MAV'RIC framework, an open-source framework designed for fast theory to experiments iterations.

We presented and motivated our architecture choices. MAV'RIC is based on a modular approach, where standard modules are linked together in project specific files. This modular approach allows easy configurability as well as high percentage of code sharing between

different users/developers. It rely on a simple scheduler scheme that proved to be sufficient to obtain performances allowing high frequency attitude control.

Several projects currently using this framework were presented, each of which has its own set of specific constraints. The design choices allowed all these projects to be based on the same code with only minimal code that are specific to each of the projects.

2.8.1 Contribution

Contributions for this work come from different persons. This sections aims at describing the contribution of each of them.

The electronic design of the MegaFly_32 board, the mechanical design of the LEQuad platform, the HIL simulator, the scheduler was performed by Felix Schill. The simulation on emulation was developed by Julien Lecoeur.

Development of low level drivers, the establishment of the design principles, flight testing and the development of educational material were a common work between all developers.

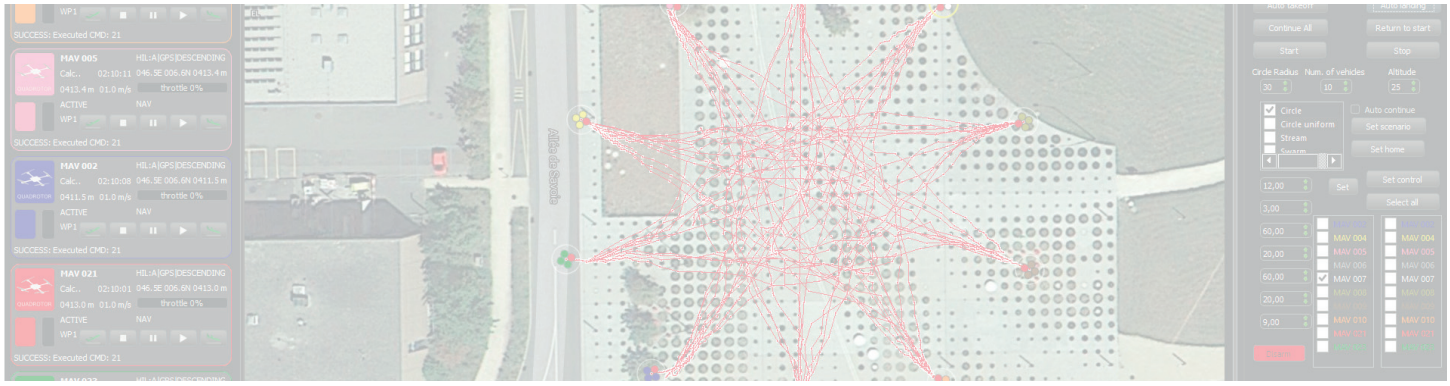
I developed the safety features (e.g. geofences, GPS failure procedure, low battery procedure), the extended Kalman filter used for attitude estimation, the tuning of the stabilization module parameters the inter-vehicle communication, the navigation strategies and the automation process (e.g. automatic takeoff and landing).

3 Extension of a Ground Control Interface for Swarms of Small Drones

ALTHOUGH the technology for fully autonomous swarms of robots is rapidly progressing, the human operator will continue to play an important role during any swarming mission due to safety, monitoring and control constraints.

In this chapter, we present the set of features that a Ground Control Interface (GCI) must incorporate to allow monitoring, control and guarantee safety of outdoor missions with a swarm of Small Drones (drones of less than $1kg$). A GCI is a wireless connected human-robot interface aimed for the supervision and the control of Small Drones. In addition, we present an extension of a widely used GCI for multi-robot handling, incorporating those features and we demonstrate its usage on a swarm of 10 Small Drones flying outdoor. This extension aims at decreasing the workload of the operator.

The chapter is organized as follows: in Section 3.1, motivations for this work are given. In Section 3.2, we describe the important features of a GCI for simple and safe operation of swarms. In Section 3.3, we present the most popular existing open-source GCIs for Small Drones. In Section 3.4, we discuss their limitations and present an extension to one of the GCIs aimed for multi-robot handling. Finally, in Section 3.5, we demonstrate its benefits and usage on a swarm of 10 Small Drones flying outdoor (Fig. 3.1). Finally, we conclude our work in Section 3.6.



This chapter is based on the following publication:

N. Dousse, G. Heitz and D. Floreano, "Extension of a Ground Control Interface for Swarms of Small Drones", *Artificial Life and Robotics*, 2016, vol. 21, pp. 308-316.

3.1 Introduction

For realistic swarming missions, safety management is a key factor in the success and acceptance of Small Drones by the general public and authorities. The regulations ruling Unmanned Aerial Vehicle have gained much attention lately. The United States Federation Aviation Authorities (FAA) have published a Notice of Proposed Rulemaking [126]. The current proposition of rules states that:

- Autonomous flights should be allowed,
- The Small Drones should remain at any time in visual line-of-sight with the pilot (i.e. to confine the area of operation),
- The pilot should be able to regain control over the Small Drone at any time,
- There should be as many pilots as Small Drones.

This last point would be the major limitation for wide use of swarms of Small Drones. The constraint of having as many pilots as Small Drones would cancel all the benefits of having a swarm of autonomous flying vehicles. We believe that in a later phase, the regulations will shift towards an operator together with a safety pilot being allowed to operate many Small Drones, if it can be proven that all the other safety requirements are still met. A possible task repartition between the operator and the safety pilot could be as follow. The operator would be responsible to monitor and control the whole swarm, while the safety pilot could remotely pilot endangered robot(s) to ensure safety (e.g. if they behave in an uncontrolled way, after a sensor failure, for instance). For this to happen, the Ground Control Interface (GCI) should be tailored to meet these other requirements.

While operating outdoors, robots are faced with uncontrolled environmental factors (e.g. wind) and human presence (e.g. buildings, isolated walkers, injured persons in a disaster

Figure 3.1 – In flight view of 5 out of the 10 Small Drones used for the experiments.

scene). They may also suffer from sudden sensor failures. Small Drones generally fly fast (e.g. a flying wing can fly up to $20 m/s$) and therefore, the time to react to any unexpected event is relatively short. This calls for a fast and easy tool to regain control by the human operator to ensure the safety of the operations. However, existing swarm interfaces of GCI do not consider safety at a swarm level.

The battery endurance of flying robots is still a limiting factor for real applications. Maximization of the operational time of the swarm should be a main criteria in the design of a swarm interface.

In [127], Bashyal showed how the performance of the swarm can be improved if the human operator can insert knowledge (e.g. *à priori* information known by the operator about the mission) in the system by taking control of an avatar (i.e. an operator-controlled robot) and driving it towards the completion of a task. In addition, in [128], Lewis analyzed the Human-Robot interaction using the command complexity. He defines the control of a swarm of n robots, by sending a task to one robot after another, as a task of complexity O(n), which scales poorly with the number of robots. In opposite, the simultaneous control of a group of robots is defined as a task of complexity of O(1), which is of constant time, independent from the number of robots. The author states that a mix between O(n) and O(1) tasks is necessary to supervise multi-robot performing realistic tasks. Depending on the situation (i.e. monitoring, control and managing safety of the mission), the operator should be able to give commands to the whole group, to a sub-part of it or to a single robot.

In [129], Drury proposes different schemes for situation awareness in Unmanned Aerial Vehicles (UAV) operations. They present four kinds of awareness: namely, human-UAV awareness, UAV-human awareness, human-human awareness and UAV-UAV awareness. Developing the three last kinds of awareness strongly depends on the mission and on the type of Small Drone used. In opposite, human-UAV awareness (i.e. the awareness of the operator about the state of the UAVs) concerns Drone swarm operations independently of the type of Drones and of the mission. In this thesis, we focus only on this awareness.

3.2 GCI features for safe and easy swarm operations

In this section, we identify and motivate the minimal set of features to be included in a GCI to monitor, control and maintain safety during a mission with multiple Small Drones. We consider not only safety for humans but also the one for onboard goods and for the drones themselves. “Within the context of aviation, safety is the state in which the possibility of harm to persons or of property damage is reduced to, and maintained at or below, an acceptable level” [130].

In Section 3.2.1, we present the features that are common to monitoring, control and safety. We then split these features into more specific features: the monitoring and control specific features in Section 3.2.2 and the safety specific features in Section 3.2.3.

3.2.1 GCI features to monitor, control and maintain safety in swarm missions

- *Identify each robot*:

  For both control and safety, the operator should be able to associate efficiently the robot in the sky with its virtual representation on the interface. Small Drones fly relatively fast and therefore require an immediate and reliable way to identify uniquely each robot of the swarm. Indeed, the time to recover from a dangerous situation or to take influence on the swarm, is inversely proportional to the speed of the robot.

- *Recover/gain manual control of one, few or all robots*:

  The performance of a swarm can be improved if the operator can take manual control of a robot to influence the swarm towards the completion of the task [127].

  In addition, if one or many robots suffer from a sensor failure (e.g. GPS failure), are subjected to strong wind gusts, approach obstacles (e.g. trees, buildings) or run low on battery, the operator should be able to take manual control back and resolve the issue (e.g. land the robot(s) or move it away from obstacles).

- *Automatic takeoff/landing of one, few or all robots*:

  The operator should be able to trigger the autonomous takeoff and landing of one, few or all robots in the shortest amount of time possible. If possible, performing these maneuvers simultaneously for all robots would increase the operational time and allow fast landing of robots when safety cannot be guaranteed anymore.

3.2.2 GCI features specific for swarm monitoring and control

- *Data logging*:

  When running experiments, the operator usually aims to record numerous flight data. To avoid loss of data packets due to high use of the wireless communication channel, the data logging should be performed on-board. The operator should be able to control the start/stop time of the logging for synchronization purposes between the robots.

- *Switch between different behaviors/modes*:

  Depending on the evolution of the mission, the operator should be able to change and to adapt the current behavior of the swarm (e.g. switch from an exploration phase to a maintenance phase).

- *Setting mission, setting parameters*:

  To decrease the deployment time, pre-programmed missions are usually loaded on the firmware of the robot. The operator should be able to set these missions and to adapt their parameters simultaneously on all robots rather than reprogram every robot individually each time. He should even be able to start and adapt these missions while flying.

- *Acknowledgment message to the operator*:

  When the operator sends a command, he should receive a feedback from every robot to inform him that it has received the command and whether it will perform it or not.

3.2.3 GCI features specific for safe swarm missions

- *Stop & Restart autonomous high-level tasks*:

  If anything may endanger a safe operation or be endangered by the swarm, the operator should be able to stop the global behavior of the swarm. All robots should then adopt a safe behavior (e.g. hover, loiter). In addition, when the operator decides that it is safe to go on with the mission, he should be able to restart the high-level task.

- *Emergency Procedure*:

  In extreme and as last resort cases, the operator should be able to rapidly command an emergency procedure (e.g. switch off the motors) on one, few or all robots. The aim of this feature is to terminate the flight of a dangerously behaving robot as quick as possible, when trying to adopt a safe behavior was not sufficient. In this case and in opposite to the previous case, the integrity of the robots may not be guaranteed.

3.3 Existing GCIs

Many open-source GCIs for Small Drones are available. We present here the most popular open-source ground control software for Small Drones:

- *OpenPilot GCI*[7]:

  This GCI was developed for the OpenPilot autopilot. This cross-platform GCI is written in C++ and uses the Qt framework[8] for the user interface. It uses its own UAVTalk communication protocol specially developed for the OpenPilot autopilot.

- *QGroundControl*[9]:

  This GCI was originally developed for the PIXHAWK autopilot[10]. It is now actively developed by a large community. This cross-platform GCI is written in C++ and uses the Qt framework for the user interface. It uses the MAVLink communication protocol[11]. The standard control interface is displayed on the left in Fig. 3.2.

- *APM Planner 2.0*[12]:

[7] https://wiki.openpilot.org/display/WIKI/OpenPilot+User+Manual
[8] http://www.qt.io/developers/
[9] http://qgroundcontrol.org/
[10] https://pixhawk.ethz.ch/
[11] http://qgroundcontrol.org/mavlink/start
[12] http://planner2.ardupilot.com/

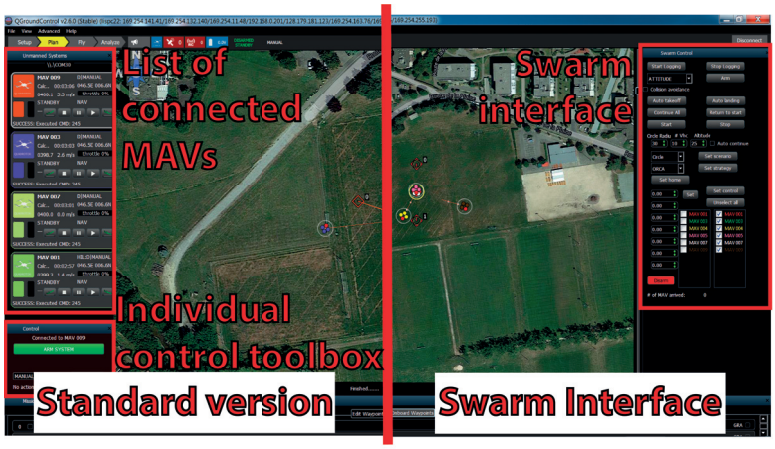


Figure 3.2 – *On the left:* QGroundControl standard interface. If a fifth robot is connected, it will appear below the control toolbox, with no possibility to scroll down in the list. Thus, the operator would not be able to interact with it. *On the right:* The proposed Swarm extension. Note up to 15 robots can be simultaneously visualized on the interface. If more robots are connected, the operator can scroll down the list to see them. This interface is presented in details in Fig. 3.6.

This GCI is based on QGroundControl and is optimized for the ArduPilot autopilot. It has replaced Mission Planner, the previously used GCI available for Windows platforms only.

- *Paparazzi GCI*[13]:

  This GCI was originally developed at ENAC, a French aerospace university. This cross-platform GCI is written in C and OCaml. It uses its own communication protocol (Ivy/PPRZ).

OpenPilot is able to control only one robot and therefore is not suited for multi-robot handling. The three others GCIs can control multiple robots. However, as the number of flying robots increases, their interface rapidly show some limitations on how the operator can control a swarm, a sub-part of it or a single robot. We discuss in detail the limitations in the following section.

3.4 Extension of a GCI for Swarms of Small Drones

In this section, we propose an open-source extension to QGroundContol GCI for allowing multi-robot handling. QGroundControl uses the MAVLink communication protocol, which is already used in more than 12 different autopilots. “MAVLink is a very lightweight, header-only message marshaling library for micro air vehicles”[14]. QGroundControl and MAVLink follow

[13] https://wiki.paparazziuav.org/wiki/GCS
[14] http://qgroundcontrol.org/mavlink/start

open standards charter, which means they are not limited to the particular hardware/software they were originally designed for. Therefore, any Small Drone using MAVLink as communication protocol could easily benefit from our extension of QGroundControl for swarming missions.

For each of the features presented in Section 3.2, we discuss here below the limitations of existing GCIs and present our solution implemented in our extension (see Table 3.1 for a summarized comparison).

In Section 3.4.5, we compare the influence of deployment and landing strategies on the maximal size of a swarm of Small Drones, given their battery endurance and the expected mission duration. Similarly, we compare the maximal mission time, given a fixed number of robots and their battery endurance.

3.4.1 Maximum number of controllable Small Drones

Existing GCIs are thought for one to few robots. Each time a new robot is connected, it appears on the interface. To control a robot, the operator has to click on it. When the number of robots increases, the lastly connected robots are not displayed on the interface anymore. Therefore, the operator is unable to select and control some of these robots.

We propose a minimalist solution that separates the selection and the control of the robots. In Fig. 3.4b, the list of connected robots is displayed. The operator can click on any of these robots in order to select it. The control of the robot is then performed in a standard separate toolbox common to all robots (e.g. MAV009 is currently the selected robot).

The first row of Table 3.1 compares the maximal number of controllable robots for each GCI. This number comes from development or a graphical limitations. These maximal numbers were obtained by tests done on a standard laptop. Note that this number may slightly vary, depending on the screen resolution.

3.4.2 Monitoring, Control & Safety features

In existing GCIs, the operator can only interact and control one robot at a time. There is no way to send commands to the whole swarm simultaneously. For every task the operator wants the swarm to accomplish, he has to select the robot he wants to control, emits the command and repeat this process for each of the robots. Thus, the tasks complexity is O(n). This issue slows down the swarm operation and affects the scalability of the missions in terms of the maximum number of robots that the operator can monitor and control.

Our extension improves the task complexity to O(1) time by emitting commands simultaneously to all robots. The insertion in QGroundControl is shown on the right in Fig. 3.2. The detailed graphical implementation of these features is furthermore shown in long dash in

Table 3.1 – Comparison of different existing GCIs for swarm control. Numbers were measured from direct tests.

GCIs	OpenPilot	QGroundControl	APM Planner 2.0	Paparazzi	Proposed GCI extension
Max # of controllable Small Drones	1	4-5	4-5	4-11	>15
Monitoring, Control & Safety features					
- Identify each robot	None	At start-up	At start-up	At start-up	ID-based
- Recover/gain manual control	No	O(n)	O(n)	O(n)	O(n) ~ O(1)
- Auto takeoff/landing	No	O(n)	O(n)	O(n)	O(1)
Monitoring & Control specific features					
- Start/Stop data logging	No	O(n)	O(n)	O(n)	O(1)
- Switch between behaviors/modes	No	O(n)	O(n)	O(n)	O(1)
- Settings Mission/parameters	No	O(n)	O(n)	O(n)	O(1)
- Acknowledgment message to operator	No	O(n)	O(n)	O(1)	O(1)
Safety specific features					
- Stop & Restart autonomous high-level tasks	No	O(n)	O(n)	O(n)	O(1)
- Emergency procedure	No	O(n)	O(n)	O(n)	O(1)

Fig. 3.6. Our extension is focused on multi robots interactions. Nevertheless, note that every operation (e.g. setting parameters, issuing commands to a particular robot) that was done with the standard GCI can still be executed with the same performance by our extension.

The second row of Table 3.1 compares the features that are common for monitoring, control and safety purposes for multi-robot handling.

3.4.2.1 Identify each robot

In existing GCIs that support multi-robot handling, there is a list of colors to distinguish the robots on the interface. It is attributed depending on the connection order (Fig. 3.3). Thus, if you don't connect the robots always in the same order, they will be displayed in different colors. Therefore, the operator cannot consistently link a color to a robot and it may take too much time to interact with it.

We present our solution in Fig. 3.4: the real robots are painted in the same colors as in the GCI. These colors depend on the robots ID. The list of robots in the GCI is sorted by these IDs. Interaction only requires to detect the color of the robot and click on the corresponding virtual representation. The speed of the interaction is increased and the risk of misidentification is decreased. However, the limitation of this method appears when the number of robots is so large that it becomes hard to distinguish between the different colors.

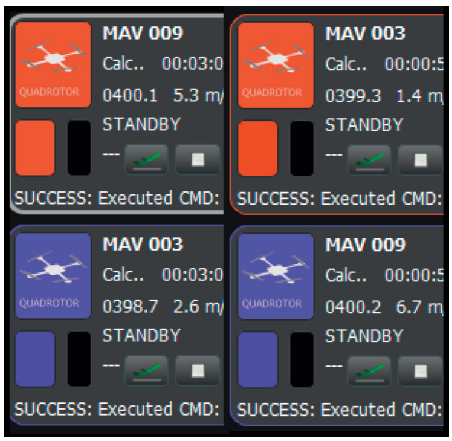
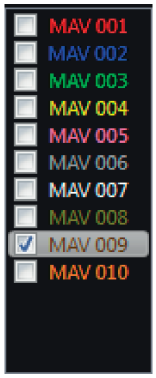


Figure 3.3 – Two different startup orders. As the color is based on the startup order, MAV 009 is once represented in red, once in blue, as for MAV 003, it is the opposite.

(a) Swarm of 10 hovering platforms used in the experiments



(b) Virtual representation of the Small Drones on the interface

Figure 3.4 – Painted robots with their associated colored virtual representation on the interface.

3.4.2.2 Recover/gain manual control of one, few or all robots

On the existing GCIs, in order to recover manual control of one or several robots, the operator has to go sequentially through every concerned robot to set manual control or restore autonomous flight.

With our solution presented in Fig. 3.5, the swarm is represented by a two lists of selectable robots. On the list on the right, the selected robots are manually controlled and the unselected ones are autonomous. Even if the selection requires as many clicks as robots, there is only one command to send to the robot. The list on the left allows to select one particular robot and interact with it with standard toolbox, as explained in Section 3.4.1.

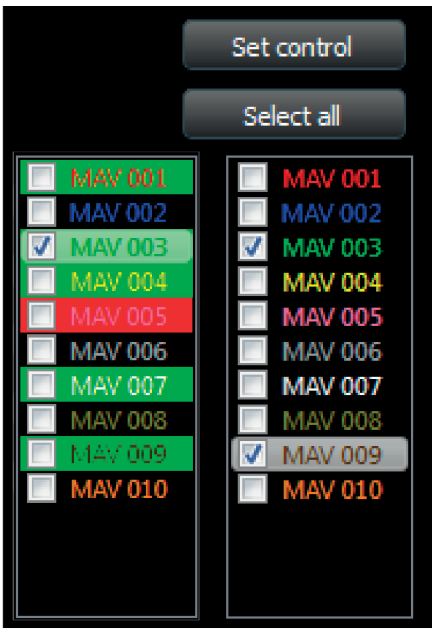


Figure 3.5 – *List on the left*: Temporarily shown visual feedback of acknowledgment messages (i.e. MAV 001, 003, 004, 007 and 009 received and accepted the command (in green), while MAV 005 received but rejected it (in red), and MAV 002, 006, 008 and 010 didn't received the command at all). In addition, this list allows the operator to select a single robot to interact with, using the standard control toolbox (e.g. MAV 003 is the currently selected robot). *List on the right*: ID-sorted list of robots for manual control (i.e. MAV 003 and MAV 009 are manually controlled and the others are autonomous).

3.4.2.3 Automatic takeoff and landing

Our extension improves the task complexity to O(1) time by emitting commands simultaneously to all robots. The advantage of performing automatic takeoff and landing is twofold. On the one hand, it allows to maximize the operational time and, on the other hand, it allows to minimize the remaining time in the air in presence of an endangering situation. Indeed, it takes as much time to takeoff or land a single robot as for the whole swarm.

3.4.3 Monitoring and control specific features

The monitoring and control specific features are

- Start/Stop data logging,
- Switch between different behaviors/modes,
- Setting mission/parameters,
- Acknowledgment message to the operator.

The detailed graphical implementation of these features is shown in short dash in Fig. 3.6. Going to a task complexity of O(1) allows to maximize the operational time. Furthermore, by setting the mission simultaneously on every robot, they will start the demanded high-level behavior simultaneously and issues due to asynchronous start are avoided.

On existing GCIs, when a command is emitted, the robot generally sends an acknowledgment back to the GCI, which is then shown on the control panel individually for each robot. In order to assert that every robot has received the command, the operator needs to select each robot one after the other and check whether the command was received or not.

Our solution is shown in Fig. 3.5. The acknowledgment is directly and temporarily shown in green (if accepted) or in red (if rejected) on the list of connected Small Drones. The operator can see directly which robot has received the command. The third row of Table 3.1 compare the monitoring and control specific features for multi-robot.

3.4.4 Safety specific features

Existing GCIs do not consider safety at a swarm level. These safety features are

- Stop & Restart autonomous high-level tasks,
- Emergency procedure.

The detailed graphical implementation of these features is shown in plain in Fig. 3.6. Going from a task complexity of O(n) to O(1) allows to minimize the time needed to perform safety related tasks. This improvement increases the overall safety. The fourth row of Table 3.1 compare the safety features for multi-robot.

3.4.5 Maximal size of a swarm of Small Drones

In this section, we compare the influence of deployment and landing strategies on the maximal size of a swarm of Small Drones, given their battery endurance and the expected mission duration. Similarly, we compare the maximal mission time, given a fixed number of robots and their battery endurance.

The deployment and landing procedure is composed of the launching command of the robots and the actual takeoff procedure of every robot. For n robots, the takeoff strategy can be performed in three different manners. First, the safety pilot can remotely takeoff one robot after the other (*manual*), second, the operator can send the takeoff command sequentially to the autonomous robots via the GCI (*sequential*) or third, the operator can send the takeoff command simultaneously to all the autonomous robots (*simultaneous*).

Let's consider a swarm of n hovering Small Drones that has to perform a mission of duration of $t_{mission}$ with a battery endurance of t_{endur} for each robot. Let t_{deploy} and t_{ld} respectively

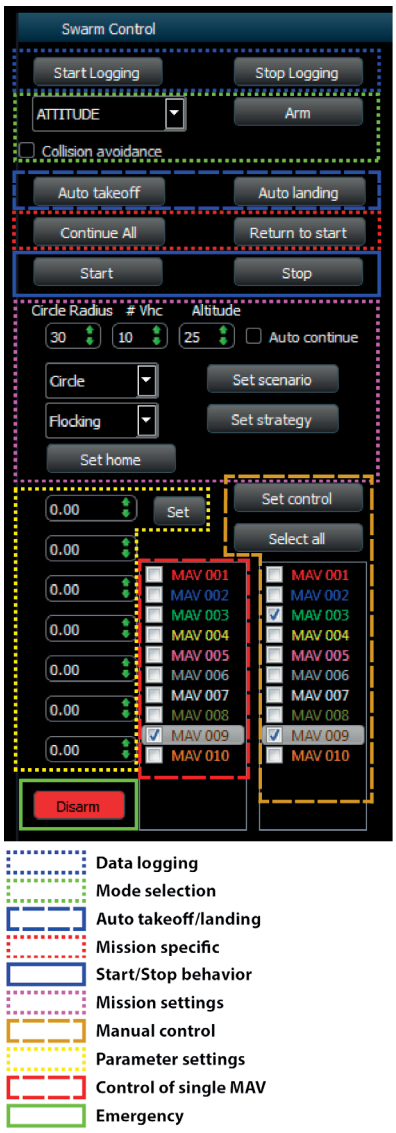


Figure 3.6 – Swarm interface of the GCI: in long dash, the features that are useful for monitoring, control and safety, in short dash, the monitoring and control specific features, in plain, the safety specific features.

be the time to deploy and the time to land all the robots at the beginning and at the end of that mission. In order to be able to accomplish the mission, we should have

$$t_{endur} \geq t_{deploy} + t_{mission} + t_{ld} \tag{3.1}$$

Let $t_{cmd_{to}}$ and t_{to} respectively be the time required to emit the takeoff command to one robot and the time required by this robot to perform the take-off. Then, for n robots, the deployment

time is equal to

$$t_{deploy} = \begin{cases} n \cdot (t_{cmd_{to}} + t_{to}), & \text{for the } \textit{manual} \text{ procedure.} \\ n \cdot t_{cmd_{to}} + t_{to}, & \text{for the } \textit{sequential} \text{ procedure.} \\ t_{cmd_{to}} + t_{to}, & \text{for the } \textit{simultaneous} \text{ procedure.} \end{cases} \tag{3.2}$$

Similarly for the landing procedure, let t_{cmd_l} and t_{land} be respectively the time required to emit the landing command and the time required by one robot to perform the landing. The landing time is equal to

$$t_{ld} = \begin{cases} n \cdot (t_{cmd_l} + t_{land}), & \text{for the } \textit{manual} \text{ procedure.} \\ n \cdot t_{cmd_l} + t_{land}, & \text{for the } \textit{sequential} \text{ procedure.} \\ t_{cmd_l} + t_{land}, & \text{for the } \textit{simultaneous} \text{ procedure.} \end{cases} \tag{3.3}$$

From Eqs. (3.1), (3.2) and (3.3), we can get the maximal size of the swarm for both the *manual* and *sequential* strategies.

$$n_{max} = \begin{cases} \frac{t_{endur} - t_{mission}}{t_{cmd_{to}} + t_{to} + t_{cmd_l} + t_{land}}, & \text{for the } \textit{manual} \text{ procedure.} \\ \frac{t_{endur} - t_{mission} - t_{to} - t_{land}}{t_{cmd_{to}} + t_{cmd_l}}, & \text{for the } \textit{sequential} \text{ procedure.} \end{cases} \tag{3.4}$$

Similarly, from Eqs. (3.1), (3.2) and (3.3), we get the maximal mission duration for the three different strategies

$$t_{mission_{max}} = t_{endur} - \begin{cases} n \cdot (t_{cmd_{to}} + t_{to} + t_{cmd_l} + t_{land}), & \text{for the } \textit{manual} \text{ procedure.} \\ n \cdot (t_{cmd_{to}} + t_{cmd_l}) + t_{to} + t_{land}, & \text{for the } \textit{sequential} \text{ procedure.} \\ t_{cmd_{to}} + t_{to} + t_{cmd_l} + t_{land}, & \text{for the } \textit{simultaneous} \text{ procedure.} \end{cases} \tag{3.5}$$

In the *manual* and *sequential* strategies, the deployment and landing time depend on the number of robots n. Then, from Eq. (3.4), we can see that the maximal size of the swarm is bounded by the battery endurance. Therefore, it exists a n_{fail} from which the size of the swarm is too large to ensure that the required mission time is reached. Then, the maximal size of the swarm is less than this n_{fail}. Similarly, from Eq. (3.1), given a fixed number of robots, the total mission time is bounded by the deployment and landing time as well as by the battery endurance. Therefore, it exists a t_{fail} from which the mission time is too long to be fulfilled by this number of robots. Then, the maximal mission time is smaller than this t_{fail}.

In opposite, in the *simultaneous* strategy, the deployment and landing time does not depend on the number of robots. Therefore the size of the swarm is not bounded. In addition, the total mission time depends only on the battery endurance, thus this strategy is easily scalable.

Although standard GCIs could be used for small number of robots (i.e. less than 5), only our interface can be used for larger swarms and/or for using the *simultaneous* strategy, as presented in Section 3.4.2.

3.5 Outdoor flying experiments and results

We demonstrate the results of Section 3.4.5 on the deployment sequence of a swarm of 10 quadrotors flying outdoors using our swarm interface compared to the basic version of QGroundControl.

The Small Drones used in our experiments have a battery endurance t_{endur} of 15 minutes. We aimed for a mission duration $t_{mission}$ of 12 minutes. While flying outdoor with our 10 Small Drones, we measured the following mean values:

- Time to send the takeoff command: $t_{cmd_{to}} = 5$ seconds,
- Time to perform the takeoff: $t_{to} = 7$ seconds,
- Time to send the landing command: $t_{cmd_l} = 4$ seconds,
- Time to perform the landing: $t_{land} = 21$ seconds.

From Eq. (3.4), we obtain the maximal size of the swarm able to accomplish the mission

$$n_{max} \begin{cases} = 5, & \text{for the } \textit{manual} \text{ procedure.} \\ = 17, & \text{for the } \textit{sequential} \text{ procedure.} \\ > 17, & \text{for the } \textit{simultaneous} \text{ procedure.} \end{cases} \tag{3.6}$$

In the *manual* and *sequential* strategies, the swam size is bounded, while in the *simultaneous* strategy, it can be arbitrarily large.

Now, we want to address the maximal mission time for a swarm of 10 Small Drones. For that, we release the condition on the duration of the mission. From Eq. (3.5), we get the following maximal mission time:

$$t_{mission_{max}} = \begin{cases} 9min, & \text{for the } \textit{manual} \text{ procedure.} \\ 13min, & \text{for the } \textit{sequential} \text{ procedure.} \\ 14min, & \text{for the } \textit{simultaneous} \text{ procedure.} \end{cases} \tag{3.7}$$

In Fig. 3.7, we represent the mission time as function of the number of Small Drones for the three different strategies. We see that given the size of the swarm, the maximal mission time is smaller in the *manual* and *sequential* strategies than in the *simultaneous* strategy. The maximal mission time for the *simultaneous* strategy does not dependent on the size of the swarm and therefore is bounded only by the battery endurance.

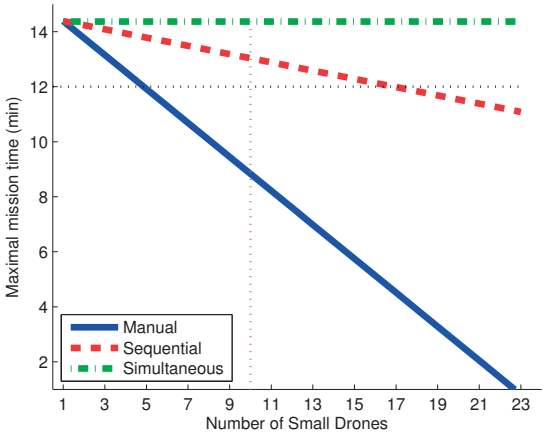


Figure 3.7 – The maximum mission time as function of the number of Small Drones for the three strategies. The intersection with the black line gives the maximal size of the swarm for a mission time of 12 minutes. The intersection with the magenta line gives the maximal mission time for a swarm of 10 Small Drones.

From Eq. 3.2 and from the measured values, we get as deployment time for 10 robots

$$t_{deploy} = \begin{cases} 120sec, & \text{for the } \textit{manual} \text{ procedure.} \\ 57sec, & \text{for the } \textit{sequential} \text{ procedure.} \\ 12sec, & \text{for the } \textit{simultaneous} \text{ procedure.} \end{cases} \tag{3.8}$$

An illustrating video of the three different deployment strategies applied to a swarm of 10 flying robots can be seen at http:/ / youtu.be/ AE0KQHYT2MI. Note that all deployment strategies were performed with our interface. As not only the standard QGroundControl can only perform the *manual* and the *sequential* strategies but also it is limited to a maximum number of 5-6 robots.

The time of deployment for the *manual* strategy is even longer than what was expected (35 seconds more). This delay can be explained by the displacement of the pilot from robot to robot in order to ease the piloting of the robots as well as the coordination between the operator and the pilot that increase the total deployment time. Indeed, the operator has to give and remove the control to and from the safety pilot after each takeoff. Coordination is required only in this strategy as the safety pilot has to know when he has control on a given robot.

In conclusion, we showed that using our interface for simultaneous and automatic deployment, the mission time of the swarm as well as the size of the swarm can be increased compared with the two other strategies.

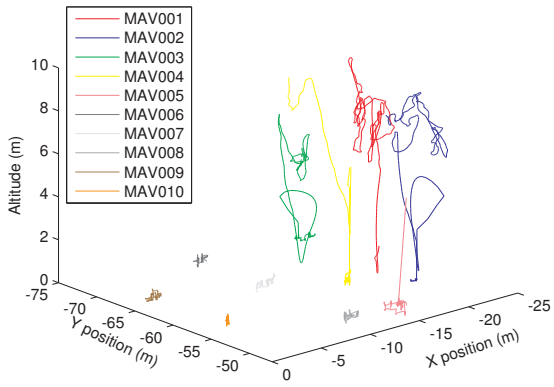


Figure 3.8 – Freeze-frame shot of the GPS tracks during the *manual* deployment strategy, MAV 001 to 004 are airborne, MAV 005 is currently taking off, while the other MAVs are still on the ground

3.6 Conclusion

In this chapter, we listed the features that should be available in a swarm interface of a Ground Control Software in order to monitor, control and guarantee the safety of a swarm experiment with Small Drones. We proposed an extension for multi-robot experiments to QGroundControl, an existing, widely-used (in more than 12 autopilots), open-source GCI for Small Drones and showed how it includes the required features.

A second video demonstrating the usage of our extension on a swarm of 10 Small Drones flying outdoor is available at https://youtu.be/wsiyYRFHrSg.

We showed that using our GCI extension to fulfill a mission allows to use a larger swarm size than using standard GCIs. Indeed, our GCI extension allows to use quicker deployment and landing strategies than standard GCIs. Similarly, we show that, given a fixed number of robots, the maximal mission time is higher using our GCI extension compared with using standard GCIs. Our GCI extension provides thus scalability to the system.

We showed the time of deployment for three deployment strategies. The *simultaneous* strategy that is only possible using our GCI extension decrease the deployment time compared to the two other strategies that can be used with standard GCIs. Thus, the maximal mission time can be closer to the battery endurance, which remains a major challenge in Aerial Robotics.

After a successful demonstration of our swarm interface, we received from the Federal Office for Civil Aviation of Germany (LBA) the authorization to fly a swarm of robots for a demonstration on a controlled airport in Braunschweig, Germany, with the collaboration, and under the supervision of the Air Traffic Controller. This authorization shows that our approach meets the expectation of the rule makers and that it could be applied in real missions with Small Drones.

The extension is freely available on GitHub[15]. The features presented here are mission independent and the interface could be easily adapted to any type of mission or platform.

3.6.1 Contribution

Contributions for this work come from different persons. This sections aims at describing the contribution of each of them.

I performed the development of the Ground Control Interface, as the flight tests were performed with the help Grégoire Heitz.

[15] https://github.com/ndousse/qgroundcontrol.git

4 Human-Comfortable Collision Free Navigation

In this chapter, we propose a reactive decentralized collision avoidance strategy that incorporates passenger physiological comfort based on the Optimal Reciprocal Collision Avoidance strategy [70]. We study in simulation the effects of increasing PAV densities on the level of comfort, on the relative flight time and on the number of collisions per flight hour and demonstrate that our strategy reduces collision risk for platforms with limited dynamic range. Finally, we validate our strategy with a swarm of 10 quadcopters flying outdoors.

The chapter is organized as follows: in Section 4.1, motivations to perform this work are given. In Section 4.2.1, we summarize the Optimal Reciprocal Collision Avoidance (ORCA) strategy from the literature, then in Section 4.2.2, we present our extension of the ORCA strategy to include passenger comfort. In Section 4.3, we apply our strategy in a real-time dynamics simulation on two challenging scenarios and discuss the effects of the main parameter and the scalability of our approach. The real-world implementation is presented in Section 4.4. Finally, in Section 4.5, we give concluding remarks.

This chapter is based on the following publication:

N. Dousse, G. Heitz, F. Schill and D. Floreano, "Human-Comfortable Collision Free Navigation for Personal Aerial Vehicles", *IEEE Robotics and Automation Letters*, 2017, vol. 2, num. 1, pp. 358-365.

4.1 Introduction

In highly decentralized, dense, and dynamic environments, future PAVs could benefit from autonomous navigation capabilities. Autonomous navigation has received a lot of attention for many years. With state-of-the-art techniques, robots can now autonomously navigate safely in known and partially unknown environments [131], in the presence of other dynamic obstacles or vehicles [132], and even in the presence of humans [118]. However, the comfort of the trajectories has rarely been the focus of these studies. Indeed, in order to be widely accepted as a new mean of transportation, PAVs automation capabilities should not only ensure safety but also tackle the problem of the passengers comfort. Even if some people do appreciate the adrenaline rushes coming from roller coaster rides, the vast majority would prefer a quiet ride. If the automation in PAVs cannot guarantee a certain level of comfort, the general public will not adopt this new transportation system [32].

Having comfortable collision-free trajectories is not only important with humans on board but also for package delivery [133] (the payload can be far from the center of mass and every acceleration provokes additional lever arm to counteract) and fire fighting [134] (water moves back and forth in the tank provoking oscillatory movements that can destabilize the robot.)

Some studies about comfort in commercial aviation focus on environmental factors such as in-cabin factors, such as noise, seat size, mobility, cabin pressure and humidity [34]. Other aviation studies measure the effect of roll oscillation frequency [35], combinations of roll and pitch motion, or combination of roll and vertical oscillations [36], [33] on physiological discomfort. Although these studies have highlighted some of the conditions that lead to discomfort, they do not offer a solution to minimize discomfort. In this thesis, we focus only on the motion quantity component, also named the physiological component of comfort [38].

It has been shown that the level of physiological discomfort for passengers rises with the magnitude of jerk [37], [38], [41] which is the time derivative of acceleration. Previous work on ground vehicles suggested to improve physiological comfort by minimizing the time integral of the square of the jerk in path planning strategies [42], [43]. In addition, the ISO norm 2631-4 recommends the use of the jerk as mean to measure comfort in fixed-guideway system (e.g. trains) [44].

Patents held by aircraft manufacturer companies have been published where jerk is used as design criteria to tackle passenger comfort in a path planning strategy for aircraft taxiing on the ground [47], in order to have a trajectory with constant ground gradient segments [45] or a trajectory following lateral and vertical constraints [46]. However, path planning strategies are only effective as long as vehicles have a global knowledge of their environment or at low density, as they have an exponential complexity in terms of number of vehicles [48]. In a dynamic environment with local and possibly incomplete knowledge, such as in a dense PATS, these approaches could fail to provide safe and comfortable trajectories for each personal aerial vehicle because the situation might change faster than the time needed to obtain global information and to compute a new path. Therefore, it is worth considering the use of reactive

collision avoidance strategies.

Here, we propose a method to include physiological comfort in the previously-published Optimal Reciprocal Collision Avoidance (ORCA) strategy [70]. We use simulations of a PATS to show the effects of increasing PAV densities on the level of comfort, on the relative flight time and on the number of collisions and we validate the main results with a group of ten real quadcopters in an outdoor environment.

ORCA has been extended to deal with accelerations limits [71], arbitrary linear dynamics [72], non-holonomic robots [73] or enforcing C^n continuous control sequences [74]. However, none of these approaches consider the variation of acceleration between consecutive time steps and therefore do not limit the jerk. Alone, these methods would not suffice to maximize passengers comfort and would require some modifications in order to minimize the jerk of the flown trajectories. In this work, we take the original strategy and show how the jerk can be minimized.

4.2 Comfortable extension of ORCA

ORCA is a reactive decentralized collision avoidance strategy that provides sufficient conditions to ensure collision-free navigation of multiple robots [70]. It is based on the Velocity Obstacles paradigm [68]: ORCA is a geometrically based strategy that solves trajectory conflicts in the velocity space.

4.2.1 Standard ORCA

It is assumed that all vehicles follow a 3D single-integrator kinematic model (i.e. the control input is directly equal to the velocity of the vehicle [135]) with a maximum speed of v_{max}, that they are using the same strategy and that they can sense the relative position and the relative velocity of their neighbors. At each time step, each vehicle performs a cycle of sensing and acting, ensuring that their trajectory will remain collision free for at least a given amount of time τ.

For a situation with two vehicles A and B, vehicle A computes the Velocity Obstacle set, which is the set of all relative velocities that will lead to a collision between vehicles A and B within a time window τ, noted $VO^{\tau}_{A|B}$. τ is also called look-ahead time, which is the amount of time during which we consider potential collisions. If the relative velocity is outside $VO^{\tau}_{A|B}$, the two vehicles are guaranteed to be collision free for at least τ amount of time (i.e. $\boldsymbol{v}_A - \boldsymbol{v}_B \notin VO^{\tau}_{A|B}$).

If the relative velocity is in $VO^{\tau}_{A|B}$, we define $\boldsymbol{u}$ as the vector from the relative velocity to the closest point outside $VO^{\tau}_{A|B}$. If vehicles A and B alter their relative velocity by at least $\boldsymbol{u}$, they will be collision free for at least a time duration of τ. The way this alteration is done is arbitrary. As vehicles A and B use the same strategy, both vehicles share equally the responsibility to alter their velocity by at least $\frac{1}{2}\boldsymbol{u}$. Therefore, the set of non-colliding velocities for vehicle A

with respect to B following this fair shared responsibility is the half-volume pointing in the direction of $\boldsymbol{u}$ at location $\boldsymbol{v}_A^{opt} + \frac{1}{2}\boldsymbol{u}$ and is called $ORCA^{\tau}_{A|B}$.

With more vehicles, this procedure is repeated for each neighboring vehicle. At the end, the set of non-colliding velocities is created by the intersection of all induced half planes.

In a free space without neighboring vehicles, vehicle A would choose $\boldsymbol{v}_A^{pref}$, its preferred velocity. This velocity would typically be given by a higher order planner (e.g. navigation, dynamic path planner). However, in presence of neighbors, choosing $\boldsymbol{v}_A^{pref}$ might lead to a collision. Therefore, once the intersection of planes is created, the new desired velocity, $\boldsymbol{v}_A^{new}$, is the velocity inside this intersection of half planes minimizing the Euclidean distance with $\boldsymbol{v}_A^{pref}$ and is obtained by

$$\boldsymbol{v}_A^{new} = \arg\min \|\boldsymbol{v} - \boldsymbol{v}_A^{pref}\| \text{ with } \boldsymbol{v} \in ORCA^{\tau}_A \tag{4.1}$$

where $\boldsymbol{v}$ are all the velocities that can be selected in $ORCA^{\tau}_A$.

4.2.2 Comfortable ORCA

We present here below our main contribution in Proposition 1: an extension to ORCA in order to adjust the level of comfort of the passengers.

We work with the following hypothesis. By definition, for the velocity $\boldsymbol{v}(t) \in \mathbb{R}^3$ and the acceleration $\boldsymbol{a}(t) \in \mathbb{R}^3$, minimizing the variation of velocity, $\Delta\boldsymbol{v}(t), \forall t$ implies minimizing the acceleration, $\boldsymbol{a}(t), \forall t$. In addition, for the jerk $\boldsymbol{J}(t) \in \mathbb{R}^3$, minimizing the variation of acceleration, $\Delta\boldsymbol{a}(t), \forall t$ implies minimizing the Jerk $\boldsymbol{J}(t), \forall t$. Thus, a strategy that minimizes the 3D variation of velocity, $\Delta\boldsymbol{v}(t)$ at every time interval will also minimize the jerk, $\boldsymbol{J}(t)$.

If we take the current velocity of the vehicle as preferred velocity, $\boldsymbol{v}_A^{pref}$ in (4.1), $\boldsymbol{v}_A^{new}$ will therefore be the minimal change of velocity ensuring a safe trajectory and will thus minimize the jerk of the trajectory. However, even if the vehicle is pointing towards its goal at the beginning, it is not guaranteed that it will reach its goal after an avoidance maneuver, as it will continue in this new direction. Therefore, we formulate the Proposition 1 to overcome this issue. The strategy was developed for 3D environments but can as well be applied in 2D. For ease of visualization, Fig. 4.1 gives only a 2D graphical interpretation of Proposition 1. In 3D, the intersection of lines defining the collision-free zones becomes an intersection of planes.

Proposition 1. *Let $\boldsymbol{v}_A^{current}$ and $\boldsymbol{v}_A^{pref}$ respectively be the current velocity and the preferred navigation velocity of a vehicle. Let further $ORCA^{\tau}_A$ be the set obtained from the intersection of half planes following the procedure explained in Section 4.2.1. From (4.1), we can then obtain $\boldsymbol{v'}_A^{current}$, the new velocity minimizing the change of velocity while ensuring a safe trajectory and $\boldsymbol{v'}_A^{pref}$, the new velocity minimizing the change of preferred navigation velocity while ensuring a safe trajectory.*

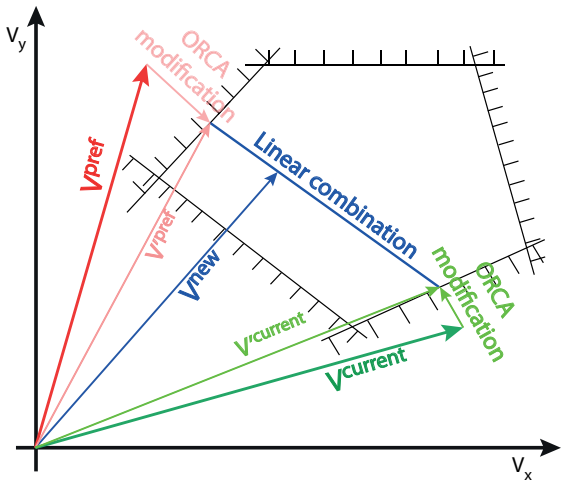


Figure 4.1 – Visual 2D representation of the Comfortable ORCA strategy. The lines become planes in 3D. $\boldsymbol{v}^{current}$ is the current velocity of the vehicle, $\boldsymbol{v}^{pref}$ is the velocity the vehicle would choose if there were no other vehicles. $\boldsymbol{v'}^{current}$ and $\boldsymbol{v'}^{pref}$ are the velocity obtained by applying (4.1) on $\boldsymbol{v}^{current}$ and $\boldsymbol{v}^{pref}$ respectively. The new velocity $\boldsymbol{v}^{new}$ is obtained by a linear combination of these two velocities and always lies within $ORCA^{\tau}_{A}$.

We define $\boldsymbol{v}^{new}_{A}$ *as the linear combination of* $\boldsymbol{v'}^{pref}_{A}$ *and* $\boldsymbol{v'}^{current}_{A}$*:*

$$\boldsymbol{v}^{new}_{A} = (1-\lambda)\cdot \boldsymbol{v'}^{pref}_{A} + \lambda \cdot \boldsymbol{v'}^{current}_{A} \text{ with } \lambda \in [0,1). \tag{4.2}$$

then $\boldsymbol{v}^{new}$ *is in* $ORCA^{\tau}$ *and therefore is safe.*

Proof. From (4.1), we know that $\boldsymbol{v'}^{current}_{A}$ and $\boldsymbol{v'}^{pref}_{A}$ are within $ORCA^{\tau}_{A}$ and therefore are collision avoiding velocities. Further, $ORCA^{\tau}_{A}$ is convex by construction and (4.2) holds by definition of a convex set. Therefore, all velocities obtained by (4.2) are within $ORCA^{\tau}_{A}$ and thus are safe. □

The pseudo-code algorithm 1 explain how $\boldsymbol{v}^{new}_{A}$ is computed for every time step following the procedure of proposition 1.

λ is a parameter allowing to balance between the current velocity and the preferred velocity. On the one hand, the closer λ is to one, the smaller will be the modification of the current velocity and therefore, accordingly the smaller will be the jerk and the trajectory more comfortable. However, the price to pay is a longer trajectory. On the other hand, a λ value close to zero means maximizing the modification of the current velocity in order to move towards the vehicle's goal and therefore the larger will be the jerk but the flight will be quicker and the trajectory more direct. Depending on the user's preferences, there is a trade-off to find between comfort and flight time.

Additionally, each vehicle can define its own λ, the strategy do not require the λ to be equal

for every vehicles. Some user may want a faster and more aggressive trajectories as other may prefer to privilege comfort. In addition, λ is not required to be constant during the whole flight. One user can decide to modify its λ value on the go, during a flight.

The interval of definition of λ is open at 1 to ensure a convergence toward the goal. Indeed, if λ is equal to one, once the vehicle has performed an avoidance maneuver, it cannot be guaranteed that the vehicle will reach its goal.

Algorithm 1 Computation of a new 3D comfortable collision-free velocity

1: **for** $i = 1..N_{neighbors}$ **do**
2: Sense the relative position and relative velocity of neighbor i.
3: Compute the half-volume for neighbor i at location $\boldsymbol{v}_A^{opt} + \frac{1}{2}\boldsymbol{u}_i$, where $\boldsymbol{u}_i \in \mathbb{R}^3$ is defined as in 4.2.1
4: Append the half-volume to the $ORCA_A^\tau$ set.
5: **end for**
6: $\boldsymbol{v'}_A^{pref} = \arg\min \|\boldsymbol{v} - \boldsymbol{v}_A^{pref}\|$ with $\boldsymbol{v} \in ORCA_A^\tau$.
7: $\boldsymbol{v'}_A^{current} = \arg\min \|\boldsymbol{v} - \boldsymbol{v}_A^{current}\|$ with $\boldsymbol{v} \in ORCA_A^\tau$.
8: $\boldsymbol{v}_A^{new} = (1-\lambda) \cdot \boldsymbol{v'}_A^{pref} + \lambda \cdot \boldsymbol{v'}_A^{current}$.
9: Send $\boldsymbol{v}_A^{new}$ to the velocity controller.

4.3 Simulation

The extension of ORCA presented in the previous section was implemented in a real-time flight dynamics simulator (Fig. 4.2) developed in our laboratory. Details of the simulator are given in Appendix A.2. This simulator, written in ADA, is capable of simulating the dynamics of dynamically constrained flying vehicles in 6 degrees of freedom and runs in discrete time steps. The vehicle dynamics are implemented as a hovering, helicopter-like or multi-rotor-like aircraft, with a body-fixed thrust vector and 3-axis rotational rate control, where accelerations and rates are bounded. Aircraft-specific effects such as rotor dynamics and precise aerodynamics were omitted for simplicity and are not required for this study.

A closed-loop velocity controller allows the vehicles to follow any 3D velocity vector, using a simple yaw coordination controller to align the vehicle with the flight direction. The current vehicle velocity may however deviate from the commanded vector due to the vehicle dynamics constraints. The ORCA implementation was adapted from the C++ RVO2-3D library[16].

The vehicles are represented as a sphere. When two spheres intersect (i.e. the distance between two vehicles is smaller than two times the size of one vehicle), we count this as a collision.

Implementing a strategy that is planned for single-integrator dynamics on dynamically constrained robots leads to safety violations as ORCA may ask for collision-free velocities that are

[16]The original ORCA C++ code is available at http://gamma.cs.unc.edu/RVO2/.

Figure 4.2 – Screenshot of our simulator.

dynamically not reachable in a single time step. In [136], the authors propose to extend the radius representing the vehicle used in the ORCA strategy to treat a non-holonomic vehicle as holonomic. Similarly, in our work, we extend the radius of the sphere representing the vehicle to encapsulate partially the dynamic constraints of the vehicles. When the distance between two vehicles is smaller than twice this extended radius and larger than twice the size of the vehicle, it is counted as a near miss.

Sensors noise can be simulated as well. Each sensor can be modeled independently following a particular noise model.

4.3.1 Experimental setup

We tested our strategy against the circle scenario (Fig. 4.3) in which vehicles travel between two waypoints located on the same altitude at opposite position around the circle. In the scenario's original formulation, the vehicles are starting around the circle, all at the same time. In this situation, all vehicles are perfectly synchronized. This leads to an enormous density at the center of the circle. Even if our strategy can be used in this scenario, as shown with the outdoor experiments in Section 4.4, in a realistic PATS, the perfect synchronization is highly unlikely and conclusions drawn from this situation would not have practical meaning. To prevent this synchronization, the vehicles start at a uniformly distributed random position inside the circle and then travel back and forth between two waypoints located at opposite side of the circle. This adaptation can model more realistic scenarios than the original formulation.

During one experiment, each vehicle is asked to perform 60 crossings of the circle. This corresponds to approximately 1 hour of simulated flight. Each set of parameters is repeated

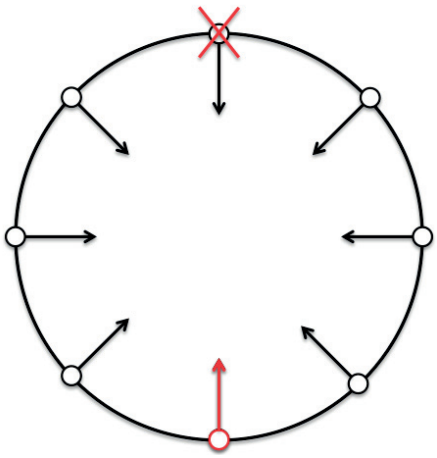


Figure 4.3 – Example of the original circle scenario for 8 vehicles: each vehicle travels back and forth between two waypoints located at opposite side of a circle. The size of the circle is parametrizable and is defined in Table 4.1.

Table 4.1 – Numerical values used in the simulations

Mass	$100kg$
Distance near miss	$5m$
Distance collision	$3m$
Look-ahead time, τ	$11s$
Circle radius	$798m$
Acceleration limit (on every axis)	$3g$

5 times, each time with different random starting positions and averaged. We compare our results relative to one vehicle flying alone with the same set of parameters and for the same number of crossings. The simulation was fully deterministic and sensors noise was not used in this section.

The set of parameters used in the simulations is defined in Table 4.1. The radius of the cicle was defined such that the density of a particular simulation is equal to the number of vehicles used in the simulation.

We then tested our strategy against more complex variations of the circle scenario. First, we added fixed obstacles that can represent for instance tall buildings located at the center of a large city or non-flying zones. Second, we addressed the effects of communication delays by slowing down the update rate of the neighbors position and velocity messages. For this second scenario, deterministic communication delays were used.

Fixed obstacles avoidance was implemented as proposed by [70]. The vehicle uses the same strategy as with neighboring vehicles, except that it takes full responsibility in the avoiding

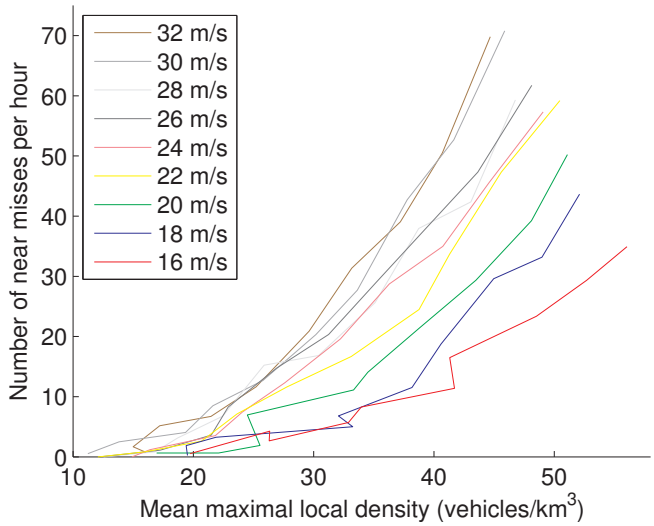


Figure 4.4 – Number of near misses as function of the density for varying cruise speed.

maneuver instead of sharing it. The vehicle therefore adds a new collision avoiding constraint for each static obstacles to the $ORCA^{\tau}_{A}$ set and solves (4.1) as in the case with only moving obstacles.

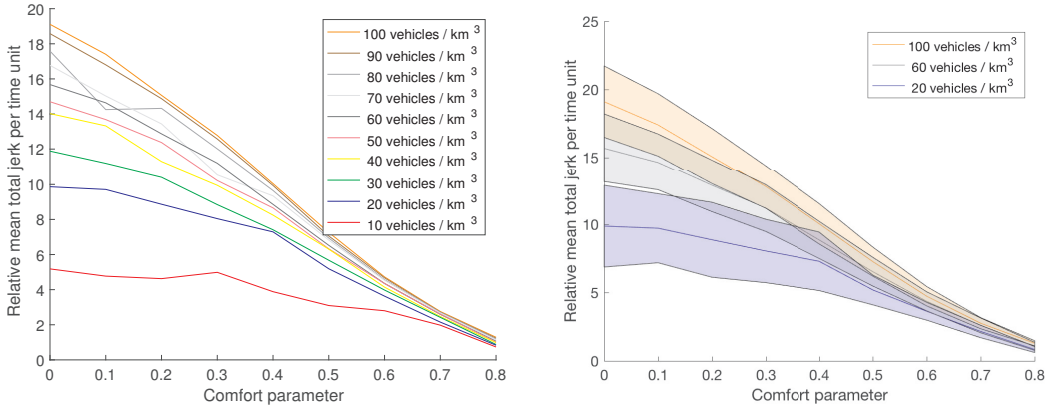
4.3.2 Results

We first address the influence of the cruise speed on the number of near misses for different densities of vehicles. In Fig. 4.4, the quadratic relationship between the density and the number of near misses can be observed for all cruise speeds, as expected from [137]. This is true regardless of the collision avoidance strategy. Therefore, even if changing the strategy could possibly lower the number of near misses, the effect of density should still receive a particular attention.

In addition, at a given density, the number of near misses increases with increasing cruise speed. As the platform is dynamically constrained, at higher cruise speed, the ORCA strategy may require to reach a collision-free velocity that is not feasible by the platform.

The cruise was then set to 26m/s which is about the expected 100km/h for future PAVs [40].

To address the comfort of the parameter sets, we compute the relative mean total jerk per time unit, $\hat{J}$. It is obtained as follow. The time integral of the square of the jerk, $\boldsymbol{J}_i(t)$ is first divided by the travel time, T_{travel}, summed for all n vehicles and for every crossing of the circle R and then averaged per vehicle and per crossing. The same ratio is computed for one vehicle flying alone the same number of crossings. We then divide the mean total jerk per time unit for n vehicles by the mean total jerk per time unit for one vehicle and obtain the relative mean total



(a) For 10 to 100 vehicles.

(b) For selected number of vehicles, with standard deviation.

Figure 4.5 – Relative time integral of the square of the jerk per time unit as function of the comfort parameter λ.

jerk per time unit. Mathematically, the relative mean total jerk per time unit, $\hat{J}$ is obtained by

$$\hat{J} = \frac{\frac{1}{R}\sum_{r=1}^{R}\frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{t=0}^{T_{end}}(\|\boldsymbol{J}_{i_r}(t)\|\Delta t)^2}{T_{travel_{i_r}}}}{\frac{1}{R}\sum_{r=1}^{R}\frac{\sum_{t=0}^{T_{end}}(\|\boldsymbol{J}_{r}(t)\|\Delta t)^2}{T_{travel_{r}}}}. \tag{4.3}$$

where R is the number of repetition of the experiments and T_{end}, the final time of the simulation.

4.3.2.1 Results for the circle scenario

In Fig. 4.5, $\hat{J}$ is represented as function of the comfort parameter for different densities. $\hat{J}$ is decreasing with increasing comfort parameter value: the higher the comfort parameter is, the less jerk per time unit the trajectory will have. This result shows that the comfort parameter is capable of decreasing the total jerk and thus increasing the comfort of the trajectory compared to the standard ORCA strategy.

In Fig. 4.6, the relative mean travel time is represented as function of the comfort parameter for different densities. The relative travel time is increasing quadratically with the comfort parameter. At high value, once the vehicle has diverged from the path to its goal due to an avoidance maneuver, it takes more time to steer it back on track.

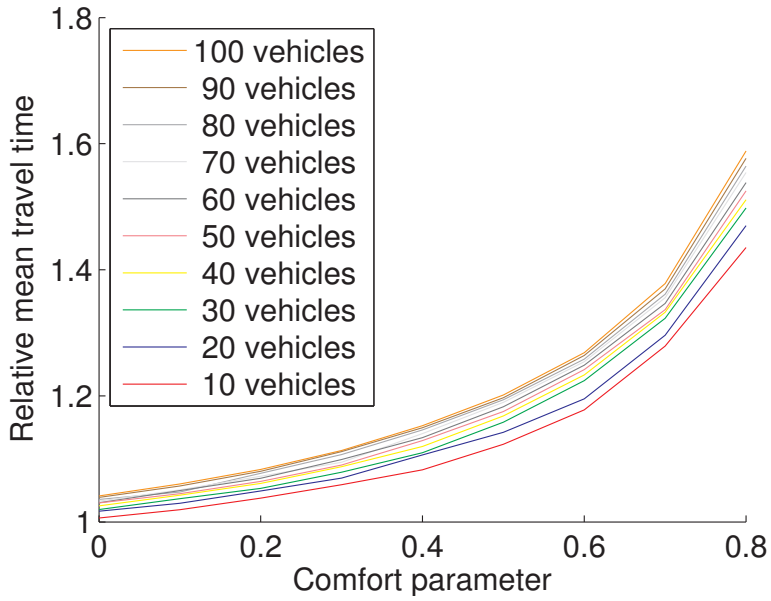


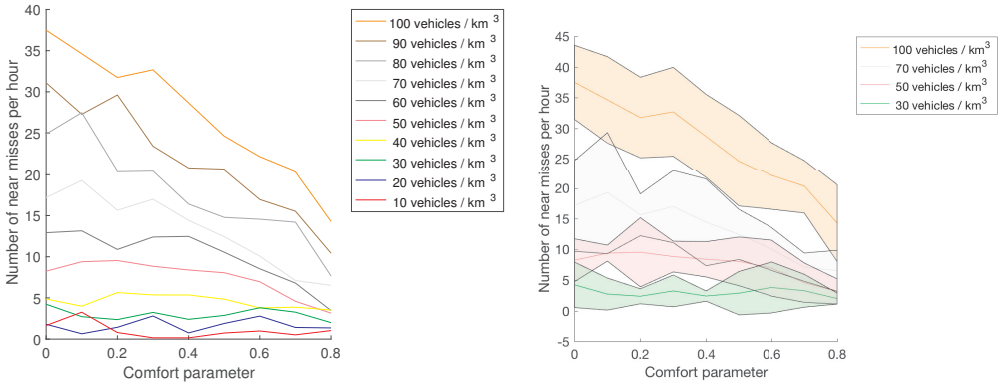
Figure 4.6 – Relative travel time as function of the comfort parameter.

In Fig. 4.7, the number of near misses per hour is represented as function of the comfort parameter for different densities. The number of near misses decreases with increasing comfort parameter values. At first, we would expect the opposite as adding this comfort parameter is slowing down the dynamics of the vehicle and thus making it less reactive for avoidance maneuvers. The vehicle could also take too much time to stop and would not avoid a collision requiring to break intensively. However, according to Proposition 1, the output velocity vector of the Comfortable ORCA strategy is always in the safe set, and limiting the variation in commanded velocity reduces the dynamic requirements from the vehicle. In Fig. 4.8, the variation of the velocity vector is represented for different comfort parameter values. The smaller the comfort parameter value, the larger the variation of velocity is during one time step. For a dynamically constrained vehicle, this variation may be unfeasible and the actually achieved velocity may lie in the unsafe zone leading to collisions or near-misses. A small velocity variation is more likely to stay in the set of dynamically feasible velocities, and therefore the vehicle's velocity is more likely to remain in the safe set.

4.3.2.2 Results for the circle scenario with fixed obstacles

Results for static obstacles show a similar trend as for the original case. The relative mean total jerk, $\hat{J}$ is decreased (Fig. 4.9a) and the relative flight time is increased (Fig. 4.9b) for λ values getting closer to 1.

Fewer near misses were observed for λ values closer to one (Fig. 4.9c). Trajectories traces for 10 vehicles and 4 fixed obstacles can be seen in Fig. 4.10. The alteration of the trajectories in order to avoid the fixed obstacles can be clearly seen.



(a) For 10 to 100 vehicles.

(b) For selected number of vehicles, with standard deviation.

Figure 4.7 – Number of near misses as function of the comfort parameter.

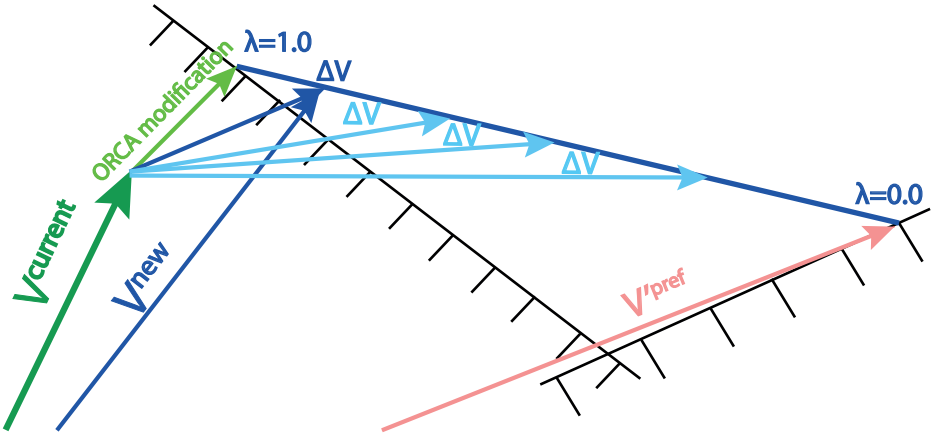


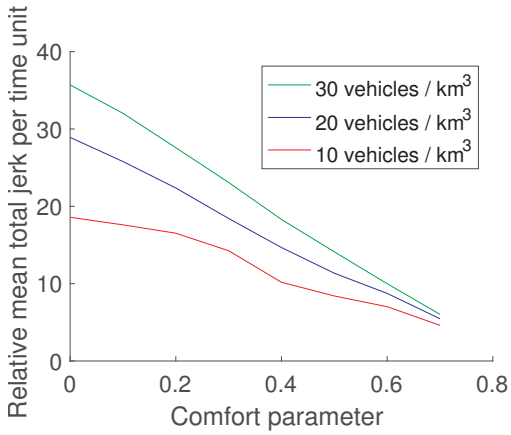
Figure 4.8 – Euclidean distance between the current velocity, $\boldsymbol{v}^{current}$, and the new velocity, $\boldsymbol{v}^{new}$ depending on the comfort parameter. We can observe that a smaller value for the comfort parameter increases the length of Δv and thus may lead to a velocity that is dynamically unreachable.

4.3.2.3 Results for the circle scenario with degraded communication

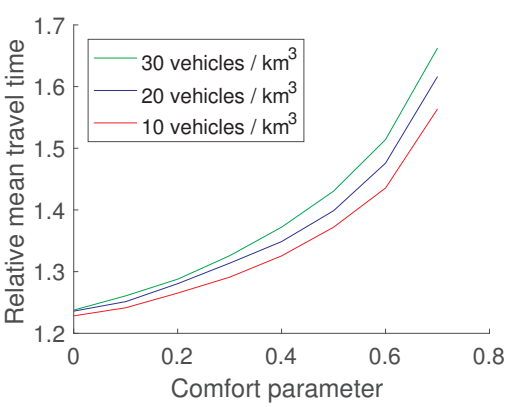
To model degraded communications, the communication update rate is varied from $0.01s$ up to $0.25s$ for a scenario with 20 vehicles. Again, the results show a similar trend as for the simpler case. $\hat{J}$ is not influenced by the update rate but still decreases with λ getting closer to one (Fig. 4.11a), thus even in presence of communication delays, our strategy do improve the comfort of the trajectories.

In addition, the relative flight time is increased with λ values getting closer to one (Fig. 4.11b). The flight time is not influenced by the communication delay.

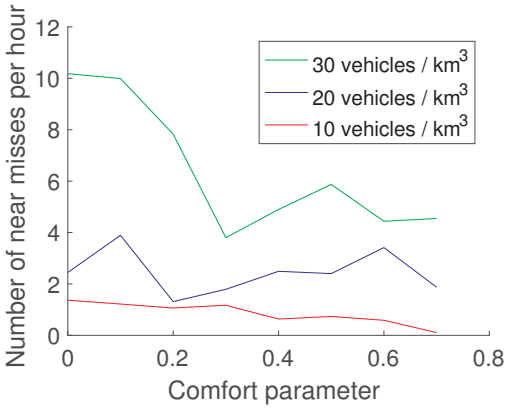
As it could be expected, increasing the communication latency leads to a higher number of



(a) Relative time integral of the square of the jerk per time unit as function of the comfort parameter λ.



(b) Relative travel time as function of the comfort parameter λ.



(c) Number of near misses as function of the comfort parameter λ.

Figure 4.9 – Results for static obstacles.

near misses per flight hour (Fig. 4.11c). The vehicles go closer one to each other as they are not aware rapidly enough of the potential threat from a neighbor. However, the number of near misses per flight hour still decreases for λ values closer to one.

4.4 Real-world implementation

In order to demonstrate the feasibility of this approach under real-time constraints and in real-world conditions, we tested our approach on 10 quadrotors flying outdoors.

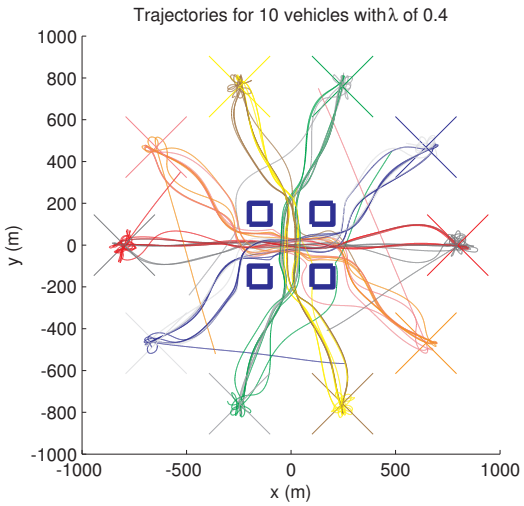


Figure 4.10 – Trajectories traces for 10 vehicles and a λ value of 0.4 with 4 fixed obstacles.

4.4.1 Flying platform

We designed and built 10 quadrotors to demonstrate our approach in real-world conditions (Fig. 4.12). For experiments with up to ten robots it is important to have a flying robot that is safe to use in large numbers, as well as easy to replicate and repair to be able to maintain a large fleet. All technical details about the design of the robot can be found in Section 2.7 and on our wiki[17]. We present here only the important details to understand the experiment.

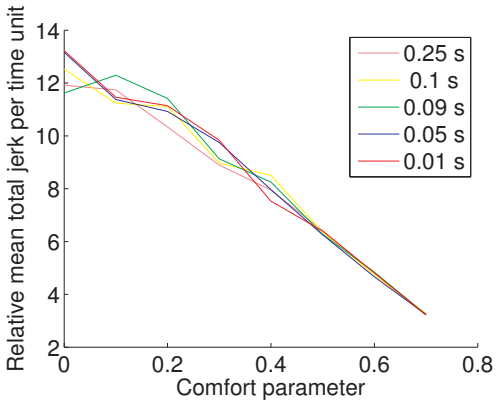
All computation is done locally on the quadrotor autopilot. Communication between the robot and the ground, is for initiating and monitoring the experiments only, and is not required for the on-board collision avoidance strategy. Robots use the same communication channel to broadcast their current state (position and velocity) to their neighbors. The robots use a time-limited nominal state prediction model [49], [51] to estimate the state of their neighbor between two messages: it is assumed that neighbors continue at the same velocity during a given time window.

The software is written in C[18]. ORCA implementation was also adapted from the C++ RVO2-3D library[19].

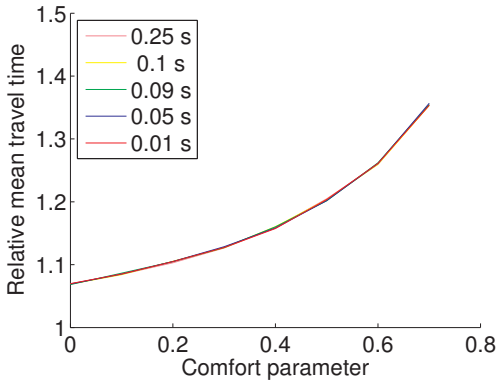
4.4.2 Experimental setup

The quadrotors take off simultaneously. On action of the operator, all the quadrotors start to cross the circle and move towards their destination located at the opposite point of the circle on the same altitude. The ten quadrotors are fully autonomous: an onboard navigation

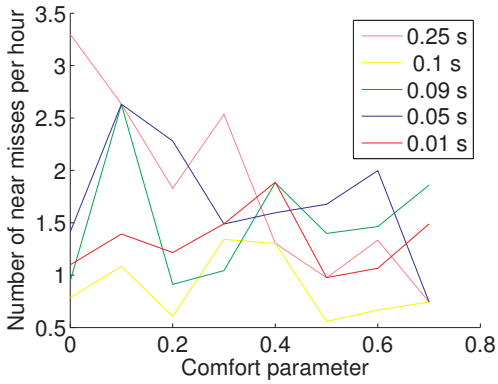
[17] http://lis-epfl.github.io/MAVRIC_Library/
[18] The code is available at https://github.com/lis-epfl/myCopter.git
[19] The original ORCA C++ code is available at http://gamma.cs.unc.edu/RVO2/.



(a) Relative time integral of the square of the jerk per time unit as function of the comfort parameter λ.



(b) Relative travel time as function of the comfort parameter λ.



(c) Number of near misses as function of the comfort parameter λ.

Figure 4.11 – Results for degraded communication.

planner sends 3D velocity commands to drive the robot directly towards its goal. Every robots are connected on the same communication channel and each robot broadcast periodically its position and velocity while listening to messages incoming from its neighbors. No guarantee is given that every robot is aware of all its neighbor at all time. Then, ORCA uses the 3D velocity and position extracted from the messages from the neighbors to compute a safe velocity.

The strategy is tested on the original circle scenario previously described, where all vehicles starts to cross the circle at the same time. We used this extreme case of the scenario in order to ensure the safety of the experiments, as it is harder to recognize a failure in 10 Small Drones randomly flying between two waypoints than having them doing one crossing at a time. Each experiment is composed of 5 crossings of the circle. The robots had a cruise speed of 3 m/s.

We compare the experimental results with the same scenario (e.g. same circle size, number of vehicles, look-ahead time λ and mass) performed in simulation. For the simulation part of this experiment, we modeled the robots by a 6 degrees of freedom multi rotor-like platform,

Figure 4.12 – The quadrotors used in the experiments.

controlled in rate, attitude and velocity. Furthermore, we modeled the noise of the sensors of the robots. The model of the noise for the position and velocity is similar to the one used by [138]. The accelerometer and velocity measurement errors are modeled by a white Gaussian noise of zero mean. The position measurement error is modeled as a Brownian motion in a parabolic potential centered on the real position of the robot.

4.4.3 Results

In this section, we address the effects of the comfort parameter on the relative mean total jerk, $\hat{J}$ on the relative mean travel time and on the number of near misses per flight hour for the circle scenario with 10 quadrotors flying outdoors and compare with the results of the same scenario in simulation.

To address the comfort of the parameter sets, we compute $\hat{J}$ from (4.3). In Fig. 4.13, $\hat{J}$ is represented as function of the comfort parameter. The expected decrease of $\hat{J}$ for increasing comfort parameter value can be observed neither with the outdoor experiment nor in simulation.

Our hypothesis is that the lack of decrease of $\hat{J}$ in Fig. 4.13 is not coming from real accelerations of the vehicle but rather from measurement noise. To investigate further, we followed two approaches: First, we removed the simulated accelerometer noise from the simulation to compare the results. Second, as post-processing step, we computed the fast Fourier transform (FFT) of the jerk from the real robots to separate flight dynamics from other noise sources such as motor vibrations and sensor noise. Note that the FFT is used only as justification of

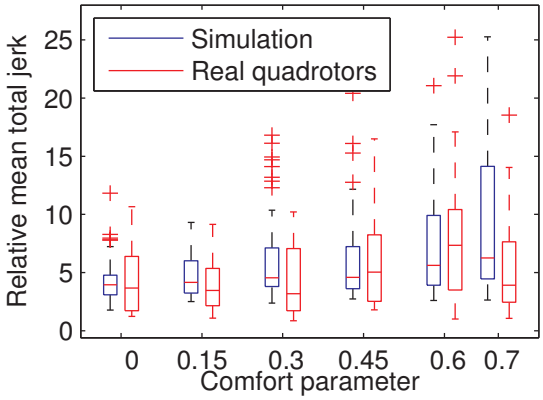


Figure 4.13 – Comparison of relative mean total jerk between simulation and outdoor flights for 10 robots.

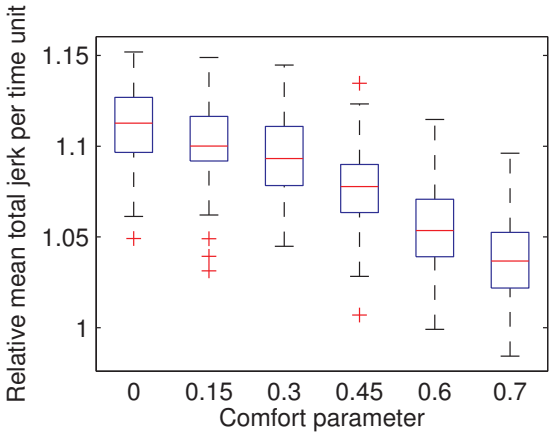


Figure 4.14 – Relative mean total jerk, $\hat{J}$ in simulation for the circle scenario with 10 vehicles with jerk computed with accelerometers without noise.

our method and is not used in the method itself.

We repeated the simulations by computing the jerk with the accelerations without noise. $\hat{J}$ decreases with increasing comfort parameter values (Fig. 4.14)) as expected from the simulation results (Fig. 4.5). This confirms that the lack of decrease of $\hat{J}$ in Fig. 4.13 is not coming from real accelerations of the vehicle but rather from measurement noise. The decrease of jerk shown in Fig. 4.14 is consistent with the previous simulation scenario and validates the use of the comfort parameter in the original circle scenario chosen for the outdoor experiments.

To extract the jerk signal from the noise, we computed the fast Fourier transform (FFT) of the jerk of the recorded data of the outdoor experiments. In Fig. 4.15, we represent this FFT for the two extreme comfort parameter values of 0 and 0.7 for the Y-axis. Similar results are obtained for the two others axes and for the other comfort parameter values. First, above 1.7 Hz, the amplitudes of the signals are in the same order of magnitude for all comfort parameter values

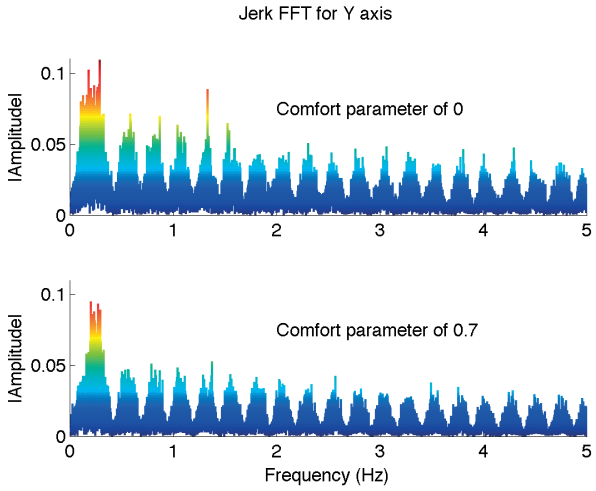


Figure 4.15 – Jerk FFT for Y-axis for comfort parameter values of 0 and 0.7.

and are likely caused by vibrations and other noise sources. Below 1.7 Hz, the amplitudes of the signal are smaller for the lower comfort parameter than for the higher comfort parameter. We consider the frequency interval of 0.15 Hz to 1.7 Hz. Below 0.15 Hz, the time span of the signal is in the order of magnitude of the time of the experiments (i.e. a crossing of a circle) and it makes no sense to associate these frequencies with a real maneuver. This frequency interval represents the frequency bandwidth in which we are expecting the dynamics of a quadrotor to take place. Additionally, it is also similar to the frequencies at which motion sickness generally happens (below 1 Hz [139]). Therefore a decrease of amplitude in this band can be expected to improve passenger comfort.

In Fig. 4.16, we represent the sum of the amplitudes of the jerk FFT divided by the sum of the amplitudes of the jerk for one vehicle flying alone, between 0.15 Hz and 1.7 Hz for different comfort parameter values. The relative sum of jerk amplitudes decreases with increasing comfort parameters for every axis. Thus, the comfort is increased for higher comfort parameter values.

The decrease of the amplitude of the signal at low frequencies as well as the decrease in the relative sum of the amplitudes for increasing comfort parameter values validate again the use of the comfort parameter as tool to increase comfort of the trajectory both in simulation and in outdoor experiments.

In Fig. 4.17, the relative mean travel time between two waypoints is represented for the simulation and the real quadrotors. In both cases, the relative flight time increases with the comfort parameter value. This trend was already observed in Fig. 4.6. Once the robots performed an avoidance maneuver, it takes more time to steer it back on its path.

In Fig. 4.18, the number of near miss per hour is represented as function of the comfort parameter. The number of near misses decreases with increasing comfort parameter similarly

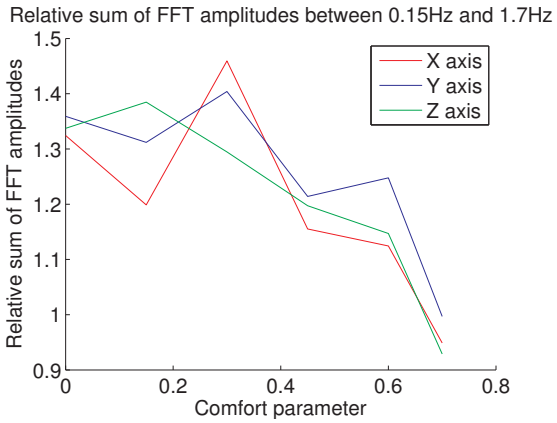


Figure 4.16 – Relative sum of the FFT amplitude for the jerk for all three axis as function of the comfort parameter value.

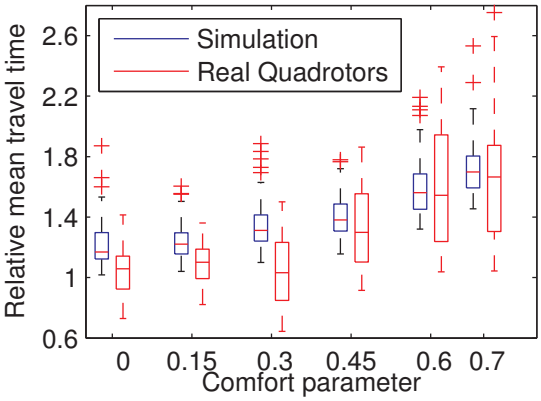


Figure 4.17 – Comparison of relative travel time between the flights with the 10 quadrotors and the simulation.

to the results obtained in Fig. 4.7. The density of this scenario is somehow extreme and even with these near-misses, collisions were never observed during the numerous outdoor flights.

ORCA can vary not only the direction but also the norm of the velocity. During all our tests, the robots velocity varied between 0 m/s (i.e. at some point they had to stop in order to avoid a collision) and 5.5 m/s.

4.5 Conclusions

We demonstrated a Reactive Comfortable Collision Avoidance strategy that is able to deal with very dense environments. An illustrating video of our simulations and experiments can be seen at https://youtu.be/ukWTK_etVzo.

Our approach to increase the comfort could also be applied to other version of ORCA. Indeed,

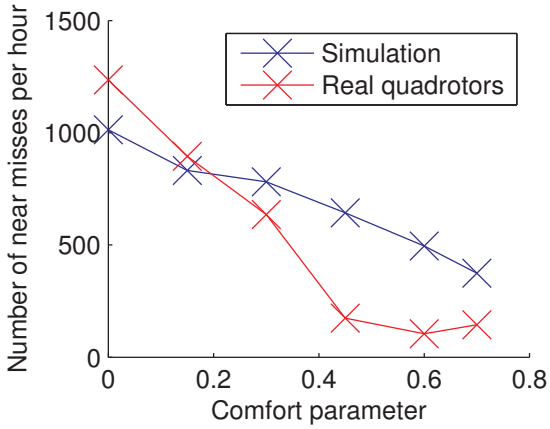


Figure 4.18 – Comparison of near misses per flight hour between the flights with the 10 quadrotors and the simulation.

our approach remains valid as long as the set of collision-free velocities created by ORCA is convex, which is the case for [71], [73], and [74]. However, the reduction in the number of near misses will probably not be observed anymore as the dynamic limitations of the platform are already explicitly taken into account in these works. In addition, in order to take into account uncertainties (e.g. noise, inaccurate sensors), the approach of [140] could be used, where the radius of the robots is increased or decreased depending on the sensing uncertainty.

We showed that using the convexity of the collision-free velocities set created following standard ORCA, defining a single parameter was sufficient to increase the comfort of the trajectory as the time integral of the jerk was decreased. We explained the effects of this comfort parameter λ on the relative mean travel time and on the relative mean total jerk per time unit. We also explained why a larger comfort parameter leads to a lower number of near misses. We also applied our strategy to more complex scenario with static obstacles or degraded communication and showed that our approach was also capable of improving passengers comfort in these more realistic situations.

With dynamics limitations, a different splitting than the one proposed (i.e. 1/2 of the velocity variation of each of the two robots) could avoid potential safety threats. However, as only local information are available, the modification of the splitting would require a negotiation phase between two or even more robots. ORCA would become a cooperative strategy and would probably loose its scalability sooner than with the standard splitting. It could therefore be a potential solution but only at low densities.

With collaborative neighbors, the vehicles share equally the responsibility to modify their relative velocity in order to become collision-free. In contrary, in order to deal with non-collaborative neighbors (e.g. vehicles with sensor or actuators failures), the vehicle could perform the full required modification of its velocity in order to avoid the other vehicle, similarly to the case with static obstacles. Therefore, this strategy could be applied not only

with collaborative vehicles but also in a scenario with non-collaborative vehicles.

ORCA formulation assumes a synchronous cycle of sensing and acting. We showed that Collision Avoidance is achieved in real-time condition even without enforcing synchronization of sensing and acting cycles. Even in an extremely dense environment and under real-time constraints, collisions never occurred between the real robots.

Our strategy can also fit with user-defined levels of comfort: the comfort parameter λ is not required to be identical for all vehicles and could also change in time. In a realistic scenario, the values for the comfort parameter λ will probably not be equal for every vehicle. Users may prefer shorter and more aggressive trajectories while others may rather fly smoothly at the cost of a longer flight time.

4.5.1 Contribution

Contributions for this work come from different persons. This sections aims at describing the contribution of each of them.

The implementation of the Comfortable ORCA strategy was performed by myself, as the flight tests were performed with the help of by Grégoire Heitz.

5 Concluding remarks

THIS chapter summarizes the main accomplishments of the thesis in the topic of human-comfortable collision-free navigation for Personal Aerial Vehicles. A new approach to tackle the comfort of the trajectories for passengers of Personal Aerial Vehicles is proposed and validated through extensive simulations and real-world tests using a framework developed to enable safe outdoor multi-drone operations.

This chapter is organized as follows. The main accomplishments of the thesis are summarized in Section 5.1. Potential direction for future work are given in Section 5.2.

5.1 Main Accomplishments

This section summarizes the main contributions of this thesis, three main contributions can be extracted from this work. The first contribution is an autopilot framework that allow fast and easy deployment and maintenance of swarms of drones. The second contribution is the set of features that must incorporate a Ground Control Interface (GCI) to monitor, control and maintain safety in outdoor experiments with a swarm of drones as well as their implementation in a GCI. Finally, the last contribution is a reactive decentralized collision-free navigation strategy that incorporates passengers comfort.

Personal Aerial Transportation Systems are a promising solution to overcome environmental and financial cost of the current road transportation system. In order to be accepted and used by a vast majority of everyday user, it must be shown that it is safe and comfortable to fly. In this thesis, a reactive decentralized collision-free navigation strategy that incorporates passengers comfort is proposed in order to ensure that Person Aerial Vehicles (PAVs) will travel safely and guaranteeing a certain level of comfort. This strategy is inspired by crowd modeling as being an example of navigation through dense environments with each agent having its own individual goal.

The strategy was tested in simulations on a challenging scenario, where PAVs travel back and forth between two points located at the opposite point of a circle. Comfort was implemented as the time integral of the square of the jerk of the trajectory. The strategy was able to maximize the comfort of the vehicles at a cost of a longer trajectory. Each user can select its own level of comfort. The level of comfort can even be adapted on-the-go during the flight. The strategy was also able to maximize the comfort with static obstacles and with degraded communication.

The strategy was then implemented and validated on a swarm of 10 quadrotors flying autonomously outdoor. In these realistic experiments, the strategy was able to maximize the comfort as well.

In order to be able to perform these experiments, a GCI was developed tailored for the needs of multi-drone experiments. This GCI was based on the analysis of the requirements needed to perform multi-drone experiments, i.e. to monitor, control and guarantee safety. Using our GCI, not only longer mission time could be achieved but also the size of the swarm that can be handled was larger compared to using existing GCIs. The scalability of our system was thus proven.

These experiments were performed using a drone that was specially designed for the ease of building, programming and maintaining a swarm of drones. This development lead to an autopilot framework that allow easy theory to experiment iterations and easy customization. It is now used already in more than six different research projects.

5.2 Future Work

Potential directions for future work are given in this section.

5.2.1 Comfort tests with passengers

On of the next step would be to validate the proposed human-comfortable collision-free strategy on real passengers. In order to control precisely the trajectory, one could use a motion simulator. Motion simulators are moving simulators that use a method known as acceleration onset cueing [141] to simulate the dynamics of a given vehicle (e.g. cars, planes, helicopters or PAVs). This method is able to reproduce the linear and angular accelerations in a way that compensates for the mechanical constraints limiting the range of motion of the simulator. Examples of motion simulator are the CableRobot simulator [142], where eight cables drive a 80kg platform up to 1.5g accelerations, and the CyberMotion simulator [143], where the simulator is based on a commercially available manipulator that can move an enclosed cabin around eight degrees of freedom. Perception cues can be projected on the walls of the cabin to simulate the environment.

As they can reproduce accelerations and therefore jerk, PAVs trajectories could be replayed and passengers could evaluate their perceived comfort for different set of parameters.

5.2.2 Realistic data acquisition

Most collision avoidance strategies do not tackle the problem of data acquisition. It is generally assumed that some information (e.g. position, velocity) is available to the control strategy to perform safely the avoidance maneuver. In low densities and when assuming broadcast of data via a communication link, it is not such an issue. However, with overuse of the communication channel or the use of other sensors, this task is not straightforward.

In the real world, sensors are noisy and give imperfect measurements. From cameras to radar or laser based sensors, measurements do not have infinite precision. Furthermore, the precision of sensors decreases with the distance and give less reliable results. The measures are also not immediate, for instance a radar works with waves traveling at a finite speed such that it takes time for the echo to come back to the emitter. Therefore, the current measurement is available with some delay compared to the real corresponding value. Finally, if an agent is blocking the field of view of a sensor, another agent located in the line of sight of the first one can be missed. All this together leads to a higher risk of collision that would be underestimated when we assume perfect sensing capabilities.

In Section 4.3.2.3, communication bandwidth limitations were introduced and the decrease in performance was measured. Future work could assess the influence on performance of other realistic sensors. Simulations could be performed with sensors with noise, less reliability with increasing distance and having a non zero probability of missing some targets.

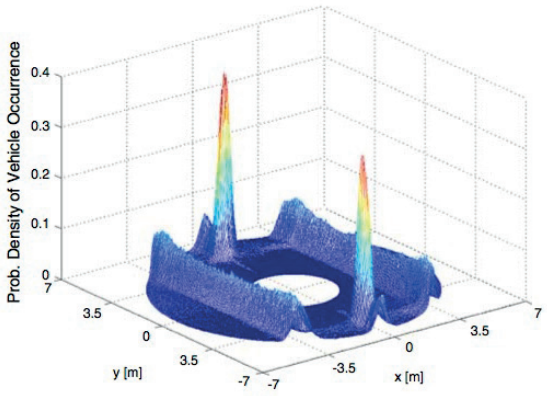


Figure 5.1 – In [144], an occurrence probability density map is obtained in simulation. The sensor suite (i.e. number, position, orientation, etc of the onboard sensors) is then optimized using weightings of the occurence probability map for each detection zone.

In [144], the methodology to optimize the number of sensors, their position, orientation, cone of view and range taking into account the developer or customer preferences is presented. The authors used a Genetic Algorithm to minimize the cost of the sensor suite while maximizing the coverage. They demonstrate their approach on a intelligent vehicle that has to monitor the surroundings traffic. The importance of the detection zone is weighted by the occurrence probability density function obtained previously in simulation (Fig. 5.1). Different ways of selecting the importance of each criteria are presented.

Inspired by this work, the number, the position and the characteristics of the sensors around the vehicle could be optimized in order to minimize the cost while maximizing the coverage for a 3D agent. A large number of encounter geometries can be collected by simulations and a occurrence probability map can be constructed to be used in the optimization process.

Another research axis could assess the influence of sensor failure(s) on the performance starting from the sensor suite found previously. Conclusion on the need of redundancy on exteroceptive sensors in order to guarantee a certain level of safety could be drawn.

A Appendix

A.1 Worst case scenario

In order to address the question of how bad the situation can be when no collision avoidance strategy is implemented, an event-based simulator was developed. The aim of this simulator was to compute what is the number of collision per flight hour that we would get if no collision avoidance strategy was implemented.

Instead of real-time simulators where simulation is performed each user-defined time steps of the order of magnitude of a few milliseconds, an event-based simulator uses a discrete event as time steps. Events can be an arbitrary long flight phase such as acceleration, cruise or deceleration. They are defined by a starting and an ending 4D point. In a regular flight, cruise is usually a long and steady phase. Instead of looping the simulation each tenths of millisecond, one single loop is sufficient to simulate the whole cruise phase. Simulation time is therefore decreased such that statistical analysis on a large number of flight hours can be performed.

Optimal control theory [145] states that the optimal trajectory between two points starting and ending with a zero velocity should have a bang-bang acceleration i.e. the acceleration should be performed at maximal value until reaching cruise velocity, then acceleration should be zero until the beginning of deceleration phase where the deceleration should be maximal.

The acceleration and deceleration phases are thus parabolas and can be approximated by piecewise linear continuous functions. Discretization of these phases leads to a trade-off between precision and number of segments.

Simulations were performed for 200 to 2000 agents flying from and to uniformly distributed 3D points in a $4km$ radius and in a $500m$ flight band in order to stay below controlled airspace above the downtown of a city. The density varies from 8 to 80 vehicles per cubic kilometers. Agents are modeled as spheres and collisions occur when two agents are closer than twice the maximal distance of the vehicle. Two types of simulations were performed, first, where

both agents were taken away from the simulation when a collision occurs, and second, where agents could continue their flights after a collision as if they hadn't have any collision. As soon as an agent landed, it was reactivated with new departing position and destination. The simulation was stopped when the sum of time of flight was equal to 10^6 hours. Each simulation was repeated 5 times.

Simulation results were compared against two analytical models, a gas model and a binomial random variable model.

The number of collisions per time unit of agents uniformly distributed can be analytically described by a gas model. In [146], the author presented an extension to the classical gas model. The agents are represented as cylinders. In [147], the authors introduce a new way to model the deterministic motion of agents opposed to the random motion of gas particles by using a proportional scaling factor k to model the deterministic behavior. This factor is obtained by fitting empirical data. Putting the two models together and adapting for spherical agents, the number of collisions per time unit is obtained by

$$C = k \cdot \frac{N^2}{2B}(\pi g^2 E(V_{rv}) + \pi h^2 E(V_{rh})), \tag{A.1}$$

where k is the scaling factor, N is the number of agents, B is the volume of the airspace, g is the horizontal dimension of the agent, h is the vertical dimension of the agent, $E(V_{rh})$ is the expected horizontal relative velocity and $E(V_{rv})$ is the expected vertical relative velocity.

In [57], the author proposed to represent the number of collisions by a binomial random variable model for agents flying in a 2D space. This work can be extended to deal with 3D agents and the number of collision is obtained by

$$C = p_t \frac{S_{sep} \cdot V \cdot T \cdot A}{2} \rho(\rho - 1/A), \tag{A.2}$$

where p_t is the scaling factor, S_{sep} is the minimum surface around an agent, V is the average aircraft velocity, T is the time interval over which conflict are counted, A is the volume of the airspace and ρ is the density.

The two parameters k and p_t were optimized by minimizing a least squared sum of the difference between the simulation and the analytical results. The results can be seen in Fig.A.1. The simulation results follow only the trend of the analytical curves. This can be explained by two main reasons. First, in both analytical approaches, it is assumed that the agents are moving at a constant velocity, which is not true in our scenario, where agents start with a zero velocity, accelerate until reaching the cruise velocity and finally decelerate to stop at their final position. Second, it is assumed that the agents are uniformly distributed. Even if their starting and ending positions are indeed uniformly distributed, the outer shell do not have uniformly distributed agents as no agents are leaving or entering the sector.

It can also be seen that there are no significant differences between the simulation that

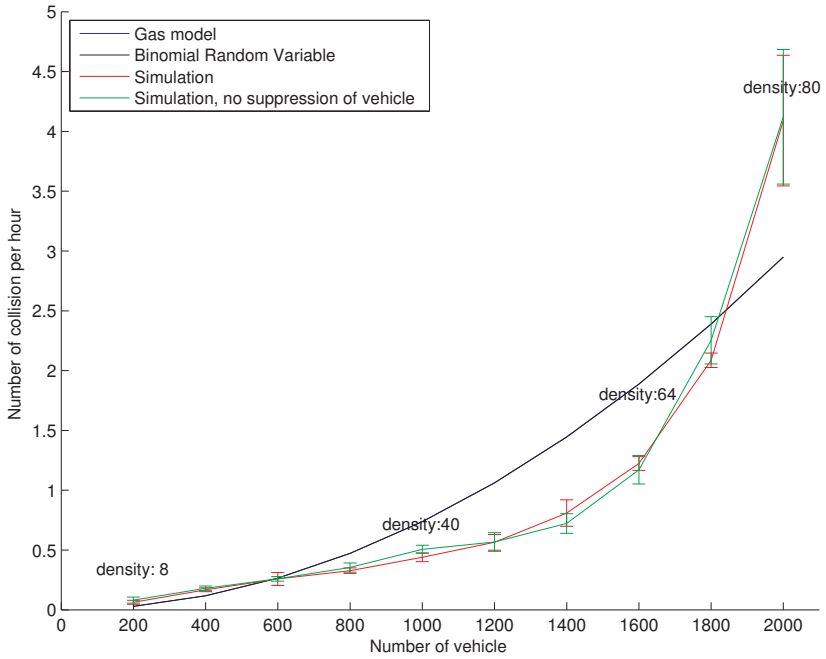


Figure A.1 – Number of collisions as function of the number of vehicles

suppress colliding agents and the one that keeps them. This similarity is explained by the total time of flight. Indeed, the agents are not flying long enough to have a sufficient probability to encounter two agents during the same flight.

These results show the importance of the implementation of a collision avoidance strategy.

A.2 Real-time simulator

A flight dynamics simulator was developed at EPFL-LIS able to simulate large number of flying agents in real-time or faster than real-time[20]. The simulator was written in Ada[21], a language designed for concurrent real-time systems and high-reliability applications. The simulator is capable of simulating the dynamics of dynamically constrained flying vehicles in 6 degrees of freedom and runs in discrete time steps. Aircraft-specific effects such as rotor dynamics and precise aerodynamics were omitted for simplicity and are not required for this study. The source code of the simulator is easily extensible and any type of collision avoidance strategy can be implemented as well as any sensor input can be simulated. Hundreds of PAVs can be simulated in real-time on a current multi-core CPU. In real-time mode, the Graphical User Interface (GUI) permits to either follow the trajectory of a particular flying agent (see Fig.4.2) or to freely navigate in the environment to see the simulation run under different perspectives. At any time, the simulation can be paused and the user can freely navigate around to change his perspective. The ability to go faster than real-time allows running a large number of simulations in order to explore the parameter space of the process and to have statistical meaningful results.

A.2.1 Simulator data flow

A simulation step is split into five different stages (see Fig. A.3) for each vehicle: three stages in the simulated real world (i.e. these stage could be implemented on a real platform) and two in the simulator layer (i.e. a layer that exists only in simulation). The output of each stage is the input of the following one. First, the neighbor selection stage selects the agents that are close enough to be in the sensor range of each other. With this set, the simulated sensors feed the autopilot with various data (e.g. position, velocity, bearing) depending on the set of simulated sensors. With this information, the autopilot computes the command to the actuators, which

[20] The swarm simulator code is available at https://github.com/lis-epfl/SwarmSimulator.git
[21] http://www.adaic.org/

Figure A.2 – Screenshot of our simulator.

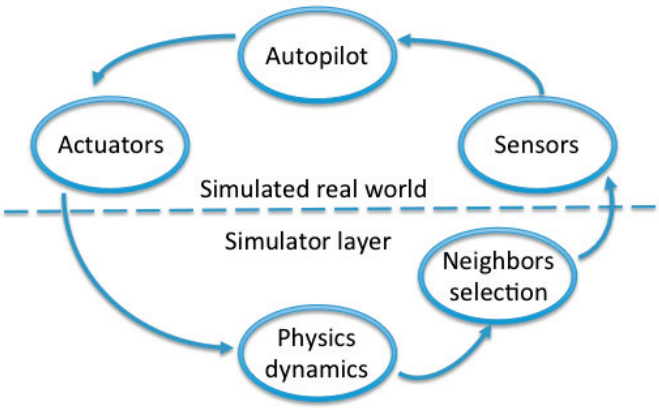


Figure A.3 – Simulator data flow chart.

then creates a command to the control surfaces. The dynamics of the agent are simulated in a last stage, updating the state of the agent.

Parallelization of the architecture of the different tasks allows taking advantage of multi-core computers to increase processing power. However computation for each agent should be synchronized at every step after the computation of the physics dynamics leading to complex synchronization events. Therefore, the simulator parallelizes each step one after the other, ensuring synchronization and efficient use of the computational power available.

A.2.2 Autopilot layer

The autopilot is implemented via a cascade of controllers: High-level orders (e.g. waypoint navigation) drives lower level controllers (e.g. attitude controller). A 3D velocity controller is used for waypoint navigation: a velocity command is given depending on the actual position and on the waypoint position of the agent. This velocity command is then used as the optimal velocity of the collision avoidance strategy. A safe velocity is then sent to an attitude controller, giving finally the corresponding angular rates and thrust. These rates and thrust are finally sent to the simulated actuators.

Bibliography

[1] myCopter. (2016). myCopter - Enabling technologies for Personal Aerial Transportation Systems. [Online]: Available: http://mycopter.eu/ (visited on 10/20/2016).

[2] T. Truman and A. de Graaff, "Out of the box, Ideas about the Future of Air Transport, Part 2", European Commission, 2007.

[3] M. Jump, G. D. Padfield, M. D. White, D. Floreano, P. Fua, J.-C. Zufferey, F. Schill, R. Siegwart, S. Bouabdallah, and M. Decker, "myCopter: Enabling Technologies for Personal Air Transport Systems", *The Future Rotorcraft–Enabling Capability Through the Application of Technology*, 2011.

[4] J. M. Hoekstra, "Metropolis WP5 Results and Simulations and Data Analysis", Delft University of Technology, Deliverable D5.2, 2015.

[5] PPlane. (2016). PPlane Project. [Online]: Available: http://www.pplane-project.org/ (visited on 10/20/2016).

[6] A. Cherpillod, "Pendul'Air Design and Realisation of a flying vehicle", EPFL, Semester Project, 2013.

[7] Terrafugia. (2016). The Transition. [Online]: Available: https://www.terrafugia.com/the-transition/ (visited on 10/20/2016).

[8] P. Moller. (2016). Moller International. [Online]: Available: http://www.moller.com (visited on 10/20/2016).

[9] M. Taylor. (2016). Aerocar. [Online]: Available: http://www.aerocar.com/ (visited on 10/20/2016).

[10] R. Olfati-Saber, "Distributed tracking for mobile sensor networks with information-driven mobility", in *American Control Conference (ACC)*, 2007, pp. 4606–4612.

[11] S. A. Quintero, G. E. Collins, and J. P. Hespanha, "Flocking with fixed-wing UAVs for distributed sensing: A stochastic optimal control approach", in *American Control Conference (ACC)*, 2013, pp. 2025–2031.

[12] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szorenyi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots", in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3866–3873.

[13] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J. C Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: communication range vs. maximum turning rate", in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5015–5020.

[14] H. A. Ruff, S. Narayanan, and M. H. Draper, "Human interaction with levels of automation and decision-aid fidelity in the supervisory control of multiple simulated unmanned air vehicles", *Presence: Teleoperators and virtual environments*, vol. 11, no. 4, pp. 335–351, 2002.

[15] S. R. Dixon, C. D. Wickens, and D. Chang, "Mission control of multiple unmanned aerial vehicles: A workload analysis", *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 47, no. 3, pp. 479–487, 2005.

[16] M. L. Cummings, C. E. Nehme, J. Crandall, and P. Mitchell, "Predicting operator capacity for supervisory control of multiple UAVs", in *Innovations in Intelligent Machines: Studies in Computational Intelligence 70*, 2007, pp. 11–37.

[17] Eurocontrol, "Performance Review Report 2011", 2011.

[18] ——, "Annual Report 2010", 2010.

[19] J. M. Hoekstra, R. C. J. Ruigrok, and R. Van Gent, "Free flight in a crowded airspace?", *Progress in Astronautics and Aeronautics*, vol. 193, pp. 533–546, 2001.

[20] M. S. Clari, R. C. Ruigrok, J. M Hoekstra, and H. G. G. Visser, "Cost-benefit study of free flight with airborne separation assurance", in *Proceegings of the AIAA Guidance, Navigation and Control Conference*, 2000.

[21] J. Krozel, M. Peters, K. D Bilimoria, C. Lee, and J. S Mitchell, "System performance characteristics of centralized and decentralized air traffic separation strategies", in *Fourth USA/Europe Air Traffic Management Research and Development Seminar*, 2001.

[22] G. Dowek, C. A. Munoz, and V. A. Carreno, "Provably Safe Coordinated Strategy for Distributed Conflict Resolution", in *AIAA Guidance, Navigation, and Control Conference*, San Francisco, California, 2005.

[23] J. C Hill, F. R Johnson, J. K Archibald, R. L Frost, and W. C Stirling, "A cooperative multi-agent approach to free flight", in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005, pp. 1083–1090.

[24] M. Pechoucek and D. Sislak, "Agent-based approach to free-flight planning, control, and simulation", *IEEE Intelligent Systems*, vol. 24, no. 1, pp. 14–17, 2009.

[25] D. R. Isaacson and H. Erzberger, "Design of a conflict detection algorithm for the Center/TRACON automation system", in *AIAA/IEEE Digital Avionics Systems Conference*, vol. 2, IEEE, 1997, pp. 9–3.

[26] S.-H. Ji, J.-S. Choi, and B.-H. Lee, "A computational interactive approach to multi-agent motion planning", *International Journal of Control Automation and Systems*, vol. 5, no. 3, p. 295, 2007.

[27] R. S. Committee and others, "Minimum aviation system performance standards for automatic dependent surveillance broadcast (ADS-B)", Technical report, January, 1998.

[28] K. Bilimoria, B. Sridhar, and G. Chatterji, "Effects of conflict resolution maneuvers and traffic density of free flight", in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 1996.

[29] E. Frazzoli, Z. H. Mao, J. H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming", AIAA Journal of Guidance, Control and Dynamics, 1999.

[30] NASA. (2015). Challenge is On to Design Sky for All. [Online]: Available: http://www.nasa.gov/feature/challenge-is-on-to-design-sky-for-all (visited on 10/20/2016).

[31] Herox. (2016). Sky for All: Air Mobility for 2035 and Beyond. [Online]: Available: https://herox.com/SkyForAll (visited on 10/20/2016).

[32] B. B. Myers and B. Marshall, "THE INFLUENCE OF COMFORT ON PASSENGER MODAL CHOICE IN WESTERN CANADA", *HUMAN FACTORS IN TRANSPORT RESEARCH EDITED BY DJ OBORNE, JA LEVIS*, vol. 2, 1980.

[33] M. Turner, M. J. Griffin, and I. Holland, "Airsickness and aircraft motion during short-haul flights", *Aviation, space, and environmental medicine*, vol. 71, no. 12, pp. 1181–1189, 2000.

[34] H. Hinninghofen and P. Enck, "Passenger well-being in airplanes", *Autonomic Neuroscience*, vol. 129, pp. 80–85, 1–2 2006.

[35] H. V. C. Howarth and M. J. Griffin, "Effect of Roll Oscillation Frequency on Motion Sickness", *Aviation, Space, and Environmental Medicine*, vol. 74, no. 4, pp. 326–331, 2003.

[36] M. E. McCauley, J. W. Royal, C. D. Wylie, J. F. O'Hanlon, and R. R. Mackie, "Motion sickness incidence: Exploratory studies of habituation, pitch and roll, and the refinement of a mathematical model", DTIC Document, 1733-2, 1976.

[37] J. Förstberg, "Ride comfort and motion sickness in tilting trains", KTH, 2000.

[38] R. D. Pepler, E. D. Sussman, and L. G. Richards, "PASSENGER COMFORT IN GROUND VEHICLES", *HUMAN FACTORS IN TRANSPORT RESEARCH EDITED BY DJ OBORNE, JA LEVIS*, vol. 2, 1980.

[39] Honeywell, *KFC500 Automatic Flight Control System - Pilot's Guide*. 1999.

[40] B. Schuchardt, P. Lehmann, F. Nieuwenhuizen, and P. Perfect, "Deliverable D6.5 Final list of desirable features/ options for the PAV and supporting systems", 2015.

[41] P. Bevilacqua, M. Frego, E. Bertolazzi, D. Fontanelli, L. Palopoli, and F. Biral, "Path planning maximising human comfort for assistive robots", in *2016 IEEE Conference on Control Applications (CCA)*, 2016, pp. 1421–1427.

[42] S. Gulati, C. Jhurani, B. Kuipers, and R. Longoria, "A framework for planning comfortable and customizable motion of an assistive mobile robot", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 4253–4260.

[43] Y. Morales, N. Kallakuri, K. Shinozawa, T. Miyashita, and N. Hagita, "Human-comfortable navigation for an autonomous robotic wheelchair", in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 2737–2743.

[44] ISO, "Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration — Guidelines for the evaluation of the effects of vibration and rotational motion on passenger and crew comfort in fixed- guideway transport systems", International Organization for Standardization, Standard 2631-4, 2001.

[45] "METHOD OF ADAPTING A SEGMENT OF AN AIRCRAFT TRAJECTORY WITH CONSTANT GROUND GRADIENT SEGMENT ACCORDING TO AT LEAST ONE PERFORMANCE CRITERION", pat. US 2016/0085239 A1, Mar. 24, 2016.

[46] "Method of computing aircraft trajectory subject to lateral and vertical constraints", pat. US 2016/0163201, Classification internationale G08G5/00; Classification coopérative G05D1/0005, G08G5/003, G06Q10/047, Mar. 24, 2016.

[47] "Method and device for generating a speed profile for an aircraft during a taxiing", pat. US 2009/0150011 A1, 6/11/09.

[48] S. M. LaValle, *Planning Algorithms*. Cambridge university press, 2006.

[49] B. Albaker and N. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles", in *Technical Postgraduates (TECHPOS), 2009 International Conference For*, Dec. 2009, pp. 1 –7.

[50] P. Angelov, *Sense and Avoid in UAS: Research and Applications*. Wiley, 2012.

[51] J. K Kuchar and L. C Yang, "A review of conflict detection and resolution modeling methods", *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[52] L. Pallottino, E. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming", *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 3 –11, Mar. 2002.

[53] P. Gupta and P. Kumar, "The capacity of wireless networks", *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388 –404, Mar. 2000.

[54] Z. Wang, H. Sadjadpour, and J. Garcia-Luna-Aceves, "A Unifying Perspective on the Capacity of Wireless Ad Hoc Networks", in *IEEE INFOCOM 2008. The 27th Conference on Computer Communications*, Apr. 2008, pp. 211 –215.

[55] D. Šišlák, J. Samek, and M. Pěchouček, "Decentralized algorithms for collision avoidance in airspace", in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2008, pp. 543–550.

[56] D. Šišlák, P. Volf, and M. Pěchouček, "Agent-Based Cooperative Decentralized Airplane-Collision Avoidance", *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, 2011.

[57] M. R. Jardin, "Air traffic conflict models", in *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, 2004.

[58] V. Duong, E. Hoffmann, L. Floc'hic, and J.-P. Nicolaon, "Extented Flight Rules to apply to the resolution of encounters in autonomous airborne separation", Eurocontrol, 1996.

[59] O. Purwin and R. D'Andrea, "Path Planning by Negotiation for Decentralized Agents", in *American Control Conference (ACC)*, Jul. 2007, pp. 5296 –5301.

[60] F. R. Johnson, J. C. Hill, J. K. Archibald, R. L. Frost, and W. C. Stirling, "A satisficing approach to free flight", in *Networking, Sensing and Control*, 2005, pp. 123–128.

[61] J. C. Hill, J. K. Archibald, W. C. Stirling, and R. L. Frost, "A multi-agent system architecture for distributed air traffic control", in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005.

[62] W. C Stirling, M. A Goodrich, and D. J. Packard, "Satisficing equilibria: a non-classical theory of games and decisions", *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 3, pp. 305–328, 2002.

[63] J. Xiaohui, Z. Xuejun, and G. Xiangmin, "A Collision Avoidance Method Based on Satisfying Game Theory", in *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 2, 2012, pp. 96 –99.

[64] J. Ruchti, R. Senkbeil, J. Carroll, J. Dickinson, J. Holt, and S. Biaz, "UAV Collision Avoidance Using Artificial Potential Fields Technical Report# CSSE11-03", 2011.

[65] K. Sigurd and J. How, "UAV trajectory design using total field collision avoidance", in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[66] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance", *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, 1997.

[67] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach", in *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*, vol. 1, 1999, 341 –346 vol.1.

[68] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles", *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[69] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation", in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference On*, 2008, pp. 1928–1935.

[70] J. van den Berg, J. Snape, S. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles", in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3475 –3482.

[71] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance", *International Symposium on Robotics Research*, pp. 3–19, 2011.

[72] D. F. Bareiss and J. van den Berg, "Reciprocal Collision Avoidance for Quadrotor Helicopters using LQR-Obstacles", in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[73] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots", in *Distributed Autonomous Robotic Systems: The 10th International Symposium*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 203–216.

[74] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal Collision Avoidance With Motion Continuity Constraints", *IEEE Transactions on Robotics*, pp. 1–14, 2013.

[75] S. Yang and M. Meng, "Neural network approaches to dynamic collision-free trajectory generation", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 3, pp. 302 –318, 2001.

[76] C. Vanaret, D. Gianazza, N. Durand, and J. B. Gotteland, "Benchmarking Conflict Resolution Algorithms", presented at the 5th International Conference on Research in Air Transportation, 2012.

[77] E. Sirot and F. Touzalin, "Coordination and Synchronization of Vigilance in Groups of Prey: The Role of Collective Detection and Predators' Preference for Stragglers", *American Naturalist*, vol. 173, no. 1, pp. 47–59, Jan. 2009.

[78] B. L. Partridge, "The structure and function of fish schools", *Scientific American*, vol. 246, no. 6, pp. 114–123, 1982.

[79] W. L. Romey and E. Galbraith, "Optimal group positioning after a predator attack: the influence of speed, sex, and satiation within mobile whirligig swarms", *Behavioral Ecology*, vol. 19, no. 2, pp. 338–343, MAR-APR 2008.

[80] D. Helbing, P. Molnar, I. J. Farkas, and K. Bolay, "Self-organizing pedestrian movement", *Environment and planning B*, vol. 28, no. 3, pp. 361–384, 2001.

[81] H. P. Zhang, A. Be'er, E. L. Florin, and H. L. Swinney, "Collective motion and density fluctuations in bacterial colonies", *Proceedings of the National Academy of Sciences*, vol. 107, no. 31, pp. 13 626–13 630, 2010.

[82] M. Moussaid, S. Garnier, G. Theraulaz, and D. Helbing, "Collective Information Processing and Pattern Formation in Swarms, Flocks, and Crowds", *Topics in Cognitive Science*, vol. 1, no. 3, pp. 469–497, 2009.

[83] Y. Li and X. Chen, "Leader-formation navigation with sensor constraints", in *2005 IEEE International Conference on Information Acquisition*, 2005, pp. 554–559.

[84] M. Saffarian and F. Fahimi, "A novel leader-follower framework for control of helicopter formation", in *Aerospace Conference, 2007 IEEE*, 2007, pp. 1–6.

[85] T. Paul, T. R. Krogstad, and J. T. Gravdahl, "Modelling of UAV formation flight using 3D potential field", *Simulation Modelling Practice and Theory*, vol. 16, no. 9, pp. 1453–1462, 2008.

[86] C. W Reynolds, "Flocks, herds and schools: A distributed behavioral model", in *ACM SIGGRAPH Computer Graphics*, vol. 21, 1987, pp. 25–34.

[87] R. Olfati-Saber, "A unified analytical look at Reynolds flocking rules", in *American Control Conference (ACC)*, 2003.

[88] N. R Watson, N. W John, and W. J Crowther, "Simulation of unmanned air vehicle flocking", in *Theory and Practice of Computer Graphics, Proceedings*, 2003, pp. 130–137.

[89] S. Hauert, S. Leven, J. C Zufferey, and D. Floreano, "Communication-based Swarming for Flying Robots", in *Proc. Intl. Conf. Robotics and Automation Workshop on Network Science and Systems*, 2010.

[90] S. Hauert, L. Winkler, J.-C. Zufferey, and D. Floreano, "Ant-based swarming with positionless micro air vehicles for communication relay", *Swarm Intelligence*, vol. 2, pp. 167–188, 2-4 2008.

[91] F. Giulietti, L. Pollini, and M. Innocenti, "Autonomous formation flight", *IEEE Control Systems Magazine*, vol. 20, no. 6, pp. 34–44, 2000.

[92] E. Justh and P. Krishnaprasad, "A Simple Control Law for UAV Formation Flying", Institute for Systems Research, TR 2002-38, 2002.

[93] C. M. Cianci, J. Pugh, and A. Martinoli, "Exploration of an Incremental Suite of Microscopic Models for Acoustic Event Monitoring Using a Robotic Sensor Network", in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, USA, 2008, pp. 3290–3295.

[94] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics", *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, May 1, 1995.

[95] B. R Fajen and W. H Warren, "Behavioral dynamics of steering, obstable avoidance, and route selection.", *Journal of Experimental Psychology: Human Perception and Performance*, vol. 29, no. 2, p. 343, 2003.

[96] B. R. Fajen and W. H. Warren, "Behavioral dynamics of intercepting a moving target", *Experimental Brain Research*, vol. 180, no. 2, pp. 303–319, Feb. 2, 2007.

[97] T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein, "Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution", *Simulation*, vol. 81, no. 5, pp. 339–352, 2005.

[98] M. Moussaid, D. Helbing, and G. Theraulaz, "How simple rules determine pedestrian behavior and crowd disasters", *Proceedings of the National Academy of Sciences*, vol. 108, no. 17, pp. 6884–6888, Apr. 18, 2011.

[99] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha, "PLEdestrians: a least-effort approach to crowd simulation", in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010, pp. 119–128.

[100] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environments", in *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, ser. I3D '08, New York, NY, USA: ACM, 2008, pp. 139–147.

[101] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt, "Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots.", in *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, 2006, pp. 72–75.

[102] A. Kolling, S. Nunnally, and M. Lewis, "Towards Human Control of Robot Swarms", in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '12, New York, NY, USA: ACM, 2012, pp. 89–96.

[103] C. Vasile, A. Pavel, and C. Buiu, "Integrating human swarm interaction in a distributed robotic control system", in *2011 IEEE Conference on Automation Science and Engineering (CASE)*, IEEE, 2011, pp. 743–748.

[104] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones", *Nature*, vol. 521, no. 7553, pp. 460–466, May 27, 2015.

[105] R. Ritz, M. W. Muller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2012, pp. 4972–4978.

[106] L. Daler, S. Mintchev, C. Stefanini, and D. Floreano, "A bioinspired multi-modal flying and walking robot", *Bioinspiration & biomimetics*, vol. 10, no. 1, p. 016 005, 2015.

[107] M. Itasse, J.-M. Moschetta, Y. Ameho, and R. Carr, "Equilibrium transition study for a hybrid mav", *International Journal of Micro Air Vehicles*, vol. 3, no. 4, pp. 229–246, 2011.

[108] J. Allred, A. B Hasan, S. Panichsakul, W. Pisano, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni, "Sensorflock: an airborne wireless sensor network of micro-air vehicles", in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, 2007, pp. 117–129.

[109] M. Varga, M. Basiri, G. Heitz, and D. Floreano, "Distributed formation control of fixed wing micro aerial vehicles for area coverage", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 669–674.

[110] P. Brisset, A. Drouin, M. Gorraz, P.-S. Huard, and J. Tyler, "The paparazzi solution", in *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*, 2006.

[111] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision", *Autonomous Robots*, vol. 33, pp. 21–39, 1-2 2012.

[112] MAVLink. (2016). MAVLink Micro Air Vehicle Communication Protocol. [Online]: Available: http://qgroundcontrol.org/mavlink/start (visited on 10/06/2016).

[113] M. D. Shuster, "A survey of attitude representations", *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[114] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays", *Report x-io and University of Bristol (UK)*, 2010.

[115] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of nonlinear attitude estimation methods", *AIAA Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[116] Y. Lin and S. Saripalli, "Path planning using 3D Dubins Curve for Unmanned Aerial Vehicles", in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 296–304.

[117] L. E Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[118] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. Di Caro, "Human-friendly Robot Navigation in Dynamic Environments", in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 423–430.

[119] R. Olfati-Saber, "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory", *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[120] G. Burri, "Design of a methodology for comparison of collision avoidance strategies", EPFL, 2015.

[121] N. Dousse, G. Heitz, F. Schill, and D. Floreano, "Human-Comfortable Collision Free Navigation for Personal Aerial Vehicles", *Robotics and Automation Letters*, vol. 2, no. 1, pp. 358–365, 2017.

[122] N. Dousse, G. Heitz, and D. Floreano, "Extension of a ground control interface for swarms of Small Drones", *Artificial Life and Robotics*, vol. 21, pp. 308–316, Sep. 2016.

[123] "FOLDABLE AIRCRAFT WITH PROTECTIVE CAGE FOR TRANSPORTATION AND TRANSPORTABILITY", pat. 25 435 408, 2016.

[124] D. Merk, "Bio-inspired Collision Avoidance in Cluttered Environments", EPFL, Master Thesis, 2016.

[125] O. J. Bertrand, J. P. Lindemann, and M. Egelhaaf, "A Bio-inspired Collision Avoidance Model Based on Spatial Information Derived from Motion Detectors Leads to Common Routes", *PLoS Comput Biol*, vol. 11, no. 11, e1004339, 2015.

[126] FAA, *Summary of small unmanned aircraft rule (Part 107)*, 2016. [Online]: https://www.faa.gov/uas/media/Part_107_Summary.pdf.

[127] S. Bashyal and G. K. Venayagamoorthy, "Human swarm interaction for radiation source search and localization", in *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, IEEE, 2008, pp. 1–8.

[128] M. Lewis, "Human Interaction With Multiple Remote Robots", *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 131–174, 2013.

[129] J. L. Drury and S. D. Scott, "Awareness in unmanned aerial vehicle operations", *The International C2 Journal*, vol. 2, no. 1, pp. 1–10, 2008.

[130] International Civil Aviation Organization, Ed., *Safety Management Manual (SMM)*, 3. ed, ser. International Civil Aviation Organization 9859, Montreal: ICAO, 2013. [Online]:

[131] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma, "Flying Fast and Low Among Obstacles: Methodology and Experiments", *The International Journal of Robotics Research*, vol. 27, no. 5, pp. 549–574, 2008.

[132] P. Nordlund and F. Gustafsson, "Probabilistic Noncooperative Near Mid-Air Collision Avoidance", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1265 –1276, 2011.

[133] A.-a. Agha-mohammadi, N. K. Ure, J. P. How, and J. Vian, "Health aware stochastic planning for persistent package delivery missions using quadrotors", in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3389–3396.

[134] K. Alexis, G. Nikolakopoulos, A. Tzes, and L. Dritsas, "Coordination of helicopter UAVs for aerial forest-fire surveillance", in *Applications of Intelligent Control to Engineering Systems*, Springer, 2009, pp. 169–193.

[135] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic Motion Planning", *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[136] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots", in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 360–366.

[137] M. R. Jardin, "Analytical relationships between conflict counts and air-traffic density", *AIAA Journal of Guidance, Control, and Dynamics*, vol. 28, no. 6, pp. 1150–1156, 2005.

[138] C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, and T. Vicsek, "Flocking algorithm for autonomous flying robots", *Bioinspiration & Biomimetics*, vol. 9, no. 2, p. 025 012, 2014.

[139] A Lawther and M. Griffin, "A survey of the occurrence of motion sickness amongst passengers at sea", *Aviation, space, and environmental medicine*, vol. 59, no. 5, pp. 399–406, 1988.

[140] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty", in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 147–154.

[141] J. Gabbai, "The art of flight simulation", *Emergent Systems, Management and Aerospace-Training*, pp. 1–16, 2001.

[142] M. P. Institute. (Dec. 22, 2016). Cablerobot simulator. [Online]: Available: http://www.cablerobotsimulator.org.

[143] ——, (Dec. 22, 2016). MPI CyberMotionSimulator. [Online]: Available: http:/ / www.cyberneum.de/facilities-research/cms.html.

[144] Y. Zhang, E. K. Antonsson, and A. Martinoli, "Evolutionary engineering design synthesis of on-board traffic monitoring sensors", *Research in Engineering Design*, vol. 19, no. 2, pp. 113–125, 2008.

[145] J. P. LaSalle, *The Bang-Bang Principal*. RIAS, 1959.

[146] S. Endoh, "Aircraft Collision Models", Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1982.

[147] S. Leven, J. C Zufferey, and D. Floreano, "Dealing with midair collisions in dense collective aerial systems", *Journal of Field Robotics*, 2011.

Nicolas Dousse

Route de la Ferme 36
1752 Villars-sur-Glâne
+41 (0)79 376 71 69
ndousse@hotmail.com
nicolas.dousse@isae-alumni.net

Prospective PhD looking for an opportunity to apply his flying robotics knowledge in a dynamic and ambitious company.

Education

2011-02.2017 **PhD, Ecole Polytechnique Fédérale de Lausanne** (Lausanne, Switzerland)

- Research interests: Mid-air collision avoidance between swarms of drones (simulation & outdoor experiments), within the laboratory of intelligent systems.

2009-2011 **Master of Science in Aerospace Mechanics and Avionics, Institut Supérieur de l'Aéronautique et de l'Espace ISAE-SUPAERO** (Toulouse, France)

- Major: Control and Guidance.
- Master Thesis done at the University of Illinois at Urbana-Champaign (Urbana-Champaign, Illinois, USA).
- 1st price at an international drone competition.

2010 **ASC leadership diploma, Level 1, Association Suisse des Cadres ASC** (Suisse)

- Diploma recognizing competences in: group animation and dynamic, adult pedagogy, feedback, stress management, oral communication skills, work technics, sense of ethics and motivation of co-workers.

2006-2009 **Bachelor of Science in Physics, University of Fribourg** (Fribourg, Switzerland)

Classes attended in German and French.

- Minor: Mathematics.

2001-2005 **Secondary Superior School, Collège St-Michel** (Fribourg, Switzerland)

- Option: Physics and applied Mathematics.

Professional experiences

2011-02.2017 **Research assistant**, Ecole Polytechnique Fédérale de Lausanne (Lausanne, Switzerland).

- Development of a drone framework in order to perform outdoor mid-air collision avoidance with multiple drones. The framework is now used in multiple projects in the laboratory. Following my researches on the safety framework, I obtained a flight clearance from the German flight authorities to perform a demo with many of my drones on a civilian airport.
- Management of multiple projects (semester/master) for Micro engineering students, as well as for an interdisciplinary robotic competition. Teacher for multiple mobile robots practicals.
- Representations of the laboratory during multiple educational events.

- Multiple articles published in international conferences and journals (publications list).

2007-2009 **Physics teacher**, High school Ste-Ursule, (Fribourg, Switzerland).

- Physics Teacher for student in last year of compulsory school (15-16 years old).

2008 **Physicist intern**, cantonal hospital of Fribourg, (Fribourg, Switzerland).

- 2 weeks internship in radio-oncology department of the hospital, establishment of a new radiation method.

Technical skills & projects

Informatics Matlab (incl. Simulink), Mathematica
C, C++ (incl. Qt), ADA
Microsoft Office, LaTeX
Catia V5

MAV'RIC Development of an autopilot framework for Small Drones (written in C et C++)
One of the major contributors to the project.

Languages

French Mother tongue

English Fluent, written and spoken (C1)
TOEFL iBT: 92 points (2009)
3rd year of elementary school at Fort Leavenworth (Kansas, USA) 1994-1995

German Fluent, written and spoken (C1)
Goethe Zertifikat C1, Zentrale Mittelstufenprüfung (2010)

Spanish Basic knowledge (A2)

Extra-professional activities

Swiss army Captain (militia). Company commander, cp car 1/1.

Bilan.ch Writing articles on the blog of the magazine Bilan about aerospace.

Aviation Private pilot license (PPL) since 2007, skydiving.

Hockey Coach for teenager from 16 to 18 years old between 2007 and 2009, holder of Jeunesse+Sport level 1 certification. Player in a regional team.

Personal information

30 years old, single, no kids, nationalities: swiss and spanish

Publications

Peer-Reviewed Journal Papers

N. Dousse, G. Heitz, F. Schill and D. Floreano, "Human-Comfortable Collision Free Navigation for Personal Aerial Vehicles", in *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 358–365, 2017.

N. Dousse, G. Heitz and D. Floreano, "Extension of a Ground Control Interface for Swarms of Small Drones", in *Artificial Life and Robotics*, vol. 21, pp. 308–316, 2016.

H. Arneson, N. Dousse and C. Langbort, "A linear programming approach to routing control in networks of constrained nonlinear positive systems with concave flow rates", in *Automatica*, vol. 68, pp. 357–368, 2016.

Peer-Reviewed Conference Proceedings

N. Dousse, G. Heitz and D. Floreano, "Extension of a Ground Control Interface for Swarms of Small Drones", in *SWARM 2015: The First International Symposium on Swarm Behavior and Bio-Inspired Robotics*. 2015.

N. Dousse, H. Arneson and C. Langbort, "Linear programming based routing design for a positive compartmental system with nonlinear flow rates and piecewise constant capacity constraints", in *American Control Conference (ACC)*, pp. 4004—4009, 2012.