UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

MASTER THESIS

TITLE: Design of an UAV Swarm

DEGREE: Master of Science in Aerospace Engineering

AUTHORS: Eduard Graells Pina

ADVISOR: Antoni Barlabé Dalmau

DATE: April 25, 2022

Title : Design of an UAV Swarm
Authors: Eduard Graells Pina

Advisor: Antoni Barlabé Dalmau

Date: April 25, 2022

Overview

This master thesis tries to give an overview on the general aspects involved in the design of an UAV swarm. UAV swarms are continuoulsy gaining popularity amongst researchers and UAV manufacturers, since they allow greater success rates in task accomplishing with reduced times. Appart from this, multiple UAVs cooperating between them opens a new field of missions that can only be carried in this way.
All the topics explained within this master thesis will explain all the agents involved in the design of an UAV swarm, from the communication protocols between them, navigation and trajectory analysis and task allocation.

CONTENTS

LIST OF FIGURES

PREFACE

The main objective of this project is to introduce the reader on the main aspects concerning the design of UAV swarms. Although this topic includes a wide field of study, the author of the present work only covers in detail the most essential parts.

First part of the project is focused on general description of UAV swarms. This includes an introduction to the concept of swarms, and how this concepts are used to create robotic swarms consisting on multiple agents. In this case those agents will be treated as unmanned aircraft configured and working in such a way to work cooperatively.

Second part of the project introduces the concept of multiple UAV architectures. This involves not only the swarm architecture, but also others where the hierarchy on which decisions and actions amongst the coalition members and how they are made, are their essential features.

The third part of the project focuses on communications between the members of a swarm. All the topics covered in this field include the protocols of communication of mobile flying elements. Those protocols will have different implementations depending on the type of architecture that the UAV swarm has.

The fourth part of the project relies on some of the algorithms that must be applied in order to simulate the most essential parts of an UAV swarm. This includes navigation, trajectory analysis and task assignments. Despite the fact that those 3 topics are heavily dense by themselves, this project only brings into discussion the most essential ones.

The fifth part will consist on a list of open source tools that must be used in order to perform in deep calculations within this field.

Due to time constraints, some aspects that should be covered within this project have been skipped. Instead, the author tries to describe in a meaningful way which additional topics should be considered to perform a full study of a UAV swarm.

Part I

UAV SWARMS

CHAPTER 1. INTRODUCTION

1.1. Swarms in nature

In biology a common definition of swarming behavior is an aggregation often combined with collective motion, which mostly agrees with the examples depicted in figure 1.1. The importance of swarms as a concept is also communicated in the language itself. For example, in English there are words for a number of swarm behaviors, such as flocking in the case of birds, shoaling or schooling in the case of fish, and herding in the case of quadrupeds. Consequently, we owe a definition of a swarm and conclude for now that a swarm is defined via its behavior.

Figure 1.1: Examples of swarms in nature [1]

1.2. Robotic swarms

1.2.1. Scenarios in swarm robotics

The purpose of this section is to review the main scenarios present on swarm robotics. Those scenarios are given according to its complexity. The least complex consists on aggregation and dispersion. It is followed by a discussion of pattern formation, object clustering, sorting, and self-assembly follows. Collective construction is a more complex scenario compared to the previous ones since it has to combine several small tasks, such as collective decision making and collective transport. We take the example of collective manipulation to discuss the interesting phenomenon of super-linear performance increase. Not only the swarm performance increases with increasing swarm size but even the individual robot's efficiency. Flocking, collective motion, foraging, shepherding are discussed

as typical examples of swarm behaviors [1].

1.2.1.1. Aggregation

Aggregation is one of the most fundamental swarm behaviors frequently observed also in natural swarms. This is probably one of the most essential features of the swarms, since it highlights the importance of staying together. Otherwise the swarm would separate into several parts, decrease in size, and eventually its survival may be endangered. Aggregation behaviors are observed in many natural swarms, such as young honeybees, ladybugs, and monarch butterflies. In aggregation the task for the robots is to position themselves close to each other in one spot. This is achieved by minimizing the distances between robots. The actual position of the aggregation spot may be specified or unspecified. If it is unspecified, then the robot swarm self-organizes to find a consensus on its aggregation position. Hence, aggregation at an unspecified spot has an inherent collective decision-making component. The aggregation spot can be specified, for example, by saying the swarm should gather at the brightest or warmest spot. Then each robot, possibly in collaboration with others, has to find that position and stop there.

The naive control approach to aggregation in swarm robotics is to move in a random direction, once you see another robot you stop. Although initially robots may reduce the average distance between robots, the swarm system gets caught in a local optimum immediately. Most clusters of robots are small, maybe two or three robots mostly. Even in this simple problem of aggregation, we face already a challenging problem of exploitation of local knowledge and exploration of the environment. Robots should not greedily exploit knowledge about their environment by staying stopped close to the first robot they meet. Instead they should also explore their surrounding sometimes and check whether bigger clusters exist that they should join.

Also note that a generic optimal solution to this exploration–exploitation trade-off does not exist because it depends on the respective scenario. How often an exploratory behavior is useful depends on the swarm density (robots per area), the swarm size, and the robot's speed. In addition, the robots are usually not synchronized, certain behaviors may create deadlocks, and the robots can hardly know when the exploration phase in the swarm is over. Finally, in the case of aggregation at a specified spot, the environment may be dynamic. Hence, the robots should always stay exploratory and send out scout robots that check whether a better aggregation spot has emerged at a different location [1].

The literature about aggregation in swarms is rich. The most popular algorithm is, as mentioned before, *BEECLUST* [7]. The swarm has to find consensus about options A and B. Both options have identical utility. A human being faced with such a problem randomly picks one. The swarm, however, has to collectively decide about which one of the two to choose based on local information only. This problem corresponds directly to the problem of having two clusters of robots with the aim of merging them. This is an even more difficult question if the other cluster is out of the robots' perception range. This is again an exploitation–exploration tradeoff that cannot easily be solved. Depending on the environmental conditions and the robot's capabilities in the given scenario, appropriate probabilities of leaving and joining a cluster need to be defined. Joining and staying at

bigger clusters should be more desirable, hence, creating a positive feedback effect.

1.2.1.2. Dispersion

Dispersion is the opposite scenario of aggregation. A robot is supposed to keep a safe distance between other robots, but not too far as to stay in contact with them. The distances between robots should be maximized but the robots should stay within communication range. This can be extremely useful if large areas of land need to be scanned or if the swarm must minimize its density. The primarily expected spatial distribution of robots is a uniform distribution, that is, the robots position themselves in a regular pattern. The task can, however, also be approximated by a random uniform distribution, or even clusters may be acceptable depending on the underlying task. This corresponds well to the three types of population distributions in ecology: clumped, random, and regular.

One implementation of dispersion is based on the biologic concept of pheromones [8]. In this approach, the robots communicate by infrared or radio messages and measure distances between robots by measurements of signal strength only.

1.2.1.3. Pattern Formation

Taking aggregation concept one step further means asking robots to form certain patterns, to sort themselves, to self-assemble into a super-robot, or to sort objects. Self-organized pattern formation is also found in many natural systems, predominantly in biology but even in self-organizing stone and salt patterns.

Patterns can be seen everywhere in nature and in our civilization. One of the most notorious cases in pattern formation can be seen in pedestrian crosses, which is an example of self-organized pattern formation. Another example of pattern formation found in nature are the trail networks formed by ants. Ants use stigmergy to organize their foraging behavior, or in other words, they mark trails with pheromones. The navigation within these trail networks seems to be based on the specific geometry of the trail system. In particular, defined angles of trail bifurcations can be leveraged by the ants to know which direction leads to food and which to the nest. This approach can also be used in swarm robotic systems for orientation in a trail network [9].

1.2.1.4. Clustering

Clustering of objects was often investigated in simulation but there are also many hardware implementations. The termite simulation model studies the behavior of termites that collect wood chips and cluster them according to 2 predefined rules [1]:

- In the case of not having yet a wood chip, then pick one up randomly.
- In the case of having a wood chip already, place it next to a randomly chosen wood chip

As refinement one can introduce probabilities, such that it is less likely to pick up a wood chip positioned close to other wood chips and more likely to place a wood chip close to many other wood chips.

Another interesting alternative work on object clustering is the approach of *anti-agents* which is based on probabilities. The standard approach to object clustering is probabilistic with a probability P_{pick} to pick up an object and P_{drop} to drop it. An anti-agent is a robot that behaves differently than the majority of robots. There has been other approaches to investigate different choices of *anti-agents*, such as random behavior, inverted neighborhood and reverse anti-agents.

1.2.1.5. Sorting

Sorting is required when multiple types of objects need to be clustered. An study carried out about multi-object sorting of annular structures [10] consist in forming a central cluster of one class of objects, and surrounding it with annular bands of the other classes, each band containing objects of only one type. This research is inspired by ants that build a perimeter wall around their nest.

A similar task is patch sorting [11]. Patch sorting is defined as grouping two or more classes of objects so that each is both clustered and segregated, and each lies outside the boundary of the other.

There are also other ways to perform sorting of any robots in a more straight way. In the case of 2 type of robots in a swarm, where each one must belong to an aggregation area, the swarm is supposed to collectively decide about which area belongs each type of robot and sort themselves [12].

1.2.1.6. Self Assembly

Self assembly can be described as the process where discrete components organize into patterns or structures autonomously. According to the previous statement, a type of self assembly behaviour was conceived and named as s-bot [13]. The main property concerning s-bots consists in forming chains of robots that can pull a heavy object collectively by connecting to each other with a gripper, coordinating their motion. In similar studies, the authors show how groups of self-assembled robots can cross gaps and steep hills, that they could not have crossed as single robots [14].

Other state-of-the-art concept of self-assembly consist on reconfigurable modular robotics, which consist on the design of autonomous robot modules that can physically connect and self-assemble into bigger robots [15]. Controlling a group of reconfigurable modular robots is considered to be challenging, especially if also the self-assembled shapes and topologies of the super-bot are allowed to be adaptive. Not all shapes can be anticipated before and the control needs to deal with them at runtime.

1.2.1.7. Collective Construction

Robot swarms can be used to collectively construct buildings or any other kind of artifact. Construction is a task that is easily paralellizable amongst different agents conforming the system, and also incorporates several other basic behaviors of swarm robotics, such as collective decision-making and task partitioning.

When studying how to apply collective construction we must redirect ourselves on how social insects build their structures. They operate based on local information and in a self-organized way. Some studies like [16] have modelled the building behavior as cellular automata, where the world is discretized as a lattice with cells. The major challenge for this cellular-automaton approach but also for any kind of collective construction is to break down the global blueprint to simple local rules. This approach is inherently based on stigmergy and it is unclear whether it is always possible to derive local rules for any given global blueprint. Even more so, it is unclear how that can be done efficiently.

Some research on swarm construction [17] tries to design a dedicated robot platform that picks up flat blocks, puts them on its back, climbs stairs built of these blocks, and places the block at an appropriate position. Even for a single robot the challenge is to ensure that it does not place building blocks at positions that may result in deadlocks. Once we allow multiple robots to construct in parallel, additional deadlock situations may arise because robots block each other's way. Using building blocks also requires relatively high precision in all actions of the robot, ranging from picking up blocks, to placing them, and not falling off the construction.

Another important research more suitable for this project is the one reporting the flight assembled architectures concept [18], where a swarm of quadrotors is used to construct architectural artifacts. The system makes use of global information, a central element, and robots are controlled offboard. The information and the robots' destinations are processed with a complex trajectory planning and trajectory generation algorithm.

1.2.1.8. Collective Transport

Collective transport is a good example for a scenario with potential for an actual swarm effect. A few robots may not be able to move a heavy object even for one centimeter but once a certain threshold in the swarm size is reached they can move it for any distance. Such an extreme steep increase in swarm performance can be observed only in few scenarios.

In the case of quadrotors, an option for collective transport studied in [19], where it reports an approach to grasping and transporting building material in groups of quadrotors. Given that this is one of the earliest works focusing on flying robots for collective transport, this is certainly a big step forward.

A drawback though is that an indoor-GPS system is used and the robots receive control values from a central computer. The research on actual swarms of quadrotors is still in an early stage because it is extremely challenging to control quadrotors with onboard tools (sensors, information processing, control).

1.2.1.9. Flocking and Collective Motion

Flocking is the coordinated motion of bird flocks, where all birds of the flock fly about in the same direction with the same speed. In one of the most classical swarm papers [20], three simple rules that allowed him to simulate a bird flock. The rules require the definition of a neighborhood, for example, by a maximal distance.

- Alignment: adapt your direction to that of your neighbors
- Cohesion: stay close to your neighbors
- Separation: if you are too close to one of your neighbors, then avoid a collision

The approach detailed in [20] is metric because neighborhood is defined on distances. Meanwhile we know that birds probably use a topological approach, that is, they check, for example, their five closest peers and ignore distances when determining their neighborhood. Other studies try to expand the theories proposed before like [21], where it reports a kinetic phase transition, which is interesting from a physics point of view.

Collective motion is a diverse field that has been studied in [22] and put in practice within [23] where collective motion in locusts is investigated. In the experiment being carried out, all the locusts are allowed to move within a ring-shaped arena. Depending on the swarm size and swarm density one observes a synchronization in their walking directions. From the two options of clockwise motion and counterclockwise motion they choose by majority only one.

It is probably not easy to accept that flocking turns out to be rather difficult to be implemented in robotic systems. Probably most problematic is the alignment rule because it assumes that each robot knows the heading of all its neighbors. Working with sensors only and without communication, it is unclear how the heading of neighbors can be perceived. While the birds probably follow visual cues, swarm robots usually do not have a good vision system. Even a single camera may not help because of the wide range that needs to be covered. Once we allow communication, it is still unclear what information should be transmitted.

If the robot has no global information, it can only try to communicate some information about a relative angle between the two considered robots. If the robots have digital compasses, which is arguably giving them access to global information, then a robot can simply transmit its absolute heading. This is the approach used in [24] with the use a digital compass and communicate the heading via a wireless communication module (IEEE 802.15.4/ZigBee compliant XBee wireless module).

A different approach consists on a simplified flocking algorithm for simple swarm robots that does neither need communication nor global information [25]. The algorithm relies on localization of neighbors according to a sophisticated virtual sector design located around each robot, where every robot can distinguish four directions to its neighbors. This can be implemented with at least four infrared sensors and a set of rules that decide whether the robot should turn or keep moving. Once the sectors are cleverly chosen and the different

rules come with the correct priorities flocking is achieved.

1.2.1.10. Foraging

When studying Foraging behaviors, we only need to take a closer look on the animal world, and specially foraging strategies of social insects. From the modeling perspective of swarm robotics, the studies performed in [26] and [27] regarding to honeybees and ants are the most relevant within this field.

In the case of swarm robotics, we should interpret foraging as a search and retrieval behavior such as search and rescue. All these tasks can be done by a single robot or a robot swarm. An interesting study for modeling swarm robotics is explained in [28], where the effect of interference in a foraging group of robots is investigated. Robots search objects, pick them up, and bring them to a certain target area. Other studies like [29] report a method to emulate pheromones for a foraging behavior similar to ants based on ultraviolet light and glow paint. Robots can draw trails on the ground by lighting their ultraviolet LED, and the trail is then detected by vision. The approach also includes a light beacon to emulate a compass. One of the most sophisticated experiments in swarm robotics can be found in [30]. Here the robots form chains, search for an object, once the object is found they collectively transport it along the chain of robots back to their base station.

1.2.1.11. Task Allocation

One the most important features regarding to robotic swarms, is their ability to perform partitioning and allocation of tasks amongst all the members within the coalition. The proper partitioning of those tasks is a complex process, where lots of algorithmic features are involved. Division of tasks means the work is organized across members while not everyone is doing the same tasks. It is important to say that those tasks should be self-contained and have similar requirements and workload volume. Task allocation is then the problem of assigning tasks to the robots, hopefully in an efficient way. Both task partitioning and task allocation can be done offline or online. An offline approach is simpler but an online approach allows for adaptations to dynamic changes. If the task allocation is dynamic, then a strategy for task switching is required. One wants to avoid frequent switching because each switch may come with a certain overhead.

One of the first studies in task allocation is explained in [31] which is based on task allocation regarding social insects and further expanded in [32]. In the first one it identifies internal factors (body size) and external factors (like the amount of already collected nectar, number of workers engaged in the task) for task allocation. The second study creates the concept of a common stomach, formed by a group of wasps who are generalists and can store water in their crops, to investigate nest construction in social wasps. The common stomach then serves not only as a storage of water but also as an information center about how well the task partitioning is done currently.

The well-known task allocation in honeybees determined by their physiological age is taken as inspiration in [33]. Their focus is on regulating robot's physiological age based on their local information about the current recruitment demand of their task. The approach is

tested in an AUV swarm (autonomous underwater vehicles) with the task of monitoring at different depths [34]. In [35] task allocation in a multi-UAV system is being analyzed with a completely distributed architecture. One of the main features is the avoidance of centralized planning but the system is synchronized by token circulation and all-to-all communication is required.

Other approaches trying to allocate tasks in sequentially interdependent tasks are studied in [36], where robots measure the delay between having finished their task and having to wait for another subtask to finish. They use this local cue to decide about task switching. The approach relies on these measured delays and on a probabilistic task switching behavior. Other methods try to apply artifical intelligence, like the one preseneted in [37] They consider task partitioning, in particular the question of deciding at runtime whether to make use of task partitioning at all. They apply wellknown methods from the multi-armed bandit problem. The considered swarm has to transport objects from A to B and robots can choose to use a cache of objects C. Hence, a robot can do the full unpartitioned service of transporting objects from A to B, or a robot can decide to go for task partitioning and transport objects only from A to C or from C to B. The robots do not know the costs of either of these three options beforehand. Hence, they have to explore and estimate the costs online while working on the task. This corresponds directly to the typical challenge in the multi-armed bandit problem. A gambler can choose to play any of the bandits' arms but initially doesn't know which one gives in average the biggest profit. The gambler, or in our problem the robot, has to explore the different options (exploration), estimate the underlying probability density, and make a choice of which option to go for (exploitation). In addition, the costs or the payoff may be dynamic, that is, each option needs to be explored from time to time again.

Other studies like [38], present a specific approach to task allocation and task switching with soft deadlines in swarm robotics. A soft deadline is a deadline that should be met but if it is not met the quality of the result is only decreased in quality wihtout any catastrophic results. The approach is probabilistic and offline, where each robot is equipped with a so-called decision matrix that defines probabilities of switching between tasks at runtime. This matrix is not changed at runtime but pre-calculated and optimized offline. The system is still adaptive to changes in the swarm size, for example, in the case of robot failures or bad network connections.

There is no robot-to-robot communication and results are verified in simulation. [39] use evolutionary algorithms in a robot swarm to evolve behaviors that show specialization, that is, task partitioning. They investigate a scenario inspired by leaf-cutting ants that has potential for task partitioning.

1.2.1.12. Shepherding

Shepherding behaviors are interesting for swarm robotics because they often require more than one shepherd robot, the behaviors of the shepherds are mutually dependent, and the multi-robot approach is not a mere parallelization of a task that could be done also by a single robot. Initial work on Shepherding was made in [40] using genetic algorithms to learn behaviors for shepherding. Another similar study like the previous one is detailed in

[41], where a co-evolutionary approach with a special technique of merging and splitting populations is investigated in further detail. Notice that all the shepherding concepts have only been applied across multiple simulations but there is any kind of hardware implementation yet.

1.2.1.13. Heterogeneous Swarms

In recent works advantages of using heterogeneous robot swarms have been investigated in [42] where they discuss heterogeneity in robotics, and also point to a niche that seems not well covered yet. Just as in the animal world we see animals across all scales from its maximum velocity, we might want to think about a similar diversity in robotics.

Especially the combination of flying robots and ground-based robots seems an advantageous approach, for example, by using the visual overview of a quadrotor to instruct wheeled robots on the ground. This way still local information is used and the system can scale up but each robot has much more information available. Some researchers like [43], investigate the impact of software heterogeneity in an aggregation task inspired by a honeybee behavior by the use of evolutionary algorithms to combine appropriate numbers of predefined robot types. Investigation on the impact of diversity in a robot swarm on performance in a task allocation problem is carried out in [44], where each task has different requirements for the robots working on it.

1.3. UAV swarms

Since its beginning, engineers and researchers have considered the possibility of cooperation amongst multiple UAVs. The tasks that could be carried out with this new concept were, reconnaissance, surveying, search and rescue, monitoring/detection in dangerous environments, terrain mapping and hazardous material handling.

Team members of a UAV coalition can exchange sensor information, collaborate to track and identify targets, perform detection and monitoring activities, or even actuate cooperatively in tasks such as the transportation of loads. Overall, the main advantages of using multiple UAVs compared to its non-swarm counterpart can be summarized as follows [2]:

- Multiple tasking: A single UAV is limited to perform a task at any given time. On the other side, an UAV coalition can sense the environment and assign tasks to the members for its resolution. Not only individual multitasking is important, but the cooperation of many UAVs to act over a specific task that requires multiple members for its resolution.
- Heterogeneous members: A coalition with multiple heterogeneous UAVs offers additional advantages due to the possibility of exploiting their complementarities. UAVs equipped with different on-board sensors or with different performances can help each other to solve a specific task. One could think on big and complex UAV carrying expensive sensors and located on a safe position delegating the dangerous tasks to other expendable UAVs.

- Efficiency: The execution time of certain missions such as exploration, track and searching for targets can be decreased when multiple vehicles are present.
- Reliability: UAV swarms leads to redundant solutions offering greater fault tolerance and flexibility including real time reconfiguration in case of losing some of the UAVs.
- Technology: The development of small, relatively low-cost UAVs is caused by the advancements of embedded systems and technologies for integration and miniaturization. Apart from this, the progress on IT technologies experienced in the last decades plays an important role in multiple vehicle systems.
- Cost: A single UAV with the performance required to execute some tasks could be an expensive solution when comparing to several low-cost vehicles performing the same task. This is clear for UAVs and particularly in smallsize, light, and low-cost versions, where constraints such as power consumption, weight, and size play an important role.

Part II

SWARM ARCHITECTURE

CHAPTER 1. INTRODUCTION

1.1. Coordination and Cooperation

When a system involves multiple vehicles, coordination and cooperation play an important role. Coordination deals with the sharing of resources, including temporal and spatial coordination:

- Temporal coordination relies on synchronization among the different vehicles. This type of coordination is used for objects monitoring, where several synchronized perceptions of the objects could be required.
- Spatial coordination of UAVs deals with the sharing of the space among them to ensure that each UAV will operate safely in spite of the plans of the other UAVs and the potential dynamic and/or static obstacles. Some formulations within this field are extended from robotics path planning concepts.

In this context, the classical planning algorithms for a single robot with multiple bodies like the ones studied in [45] may be applied without adaptation for centralized planning. The main concern, however, is that the dimension of the state space grows linearly in the number of UAVs. Complete algorithms require time that is at least exponential in dimension, which makes them unlikely candidates for such problems. Sampling based algorithms are more likely to scale well in practice when there are many UAVs, but the resulting dimension might still be too high. For such cases, there are also decoupled path planning approaches such as the prioritized planning that considers one vehicle at a time according to a global priority.

Cooperation can be defined as a joint collaborative behavior that is directed toward some goal in which there is a common interest or reward. Moreover, given some task specified by a designer, a multiple-robot system displays cooperative behavior if, due to some underlying mechanism, there is an increase in the total utility of the system [46]. The cooperation of heterogeneous vehicles requires the integration of sensing, control, and planning in an appropriated decisional architecture.

These architectures can be either centralized or decentralized depending on the assumptions on the knowledge's scope and accessibility of the individual vehicles, their computational power, and the required scalability. A centralized approach will be relevant if the computational capabilities are compatible with the amount of information to process and the exchange of data meets both the requirements of speed (up-to-date data) and expressivity (quality of information enabling well-informed decision-taking). On the other hand, a distributed approach will be possible if the available knowledge within each distributed vehicle is sufficient to perform coherent decisions, and this required amount of knowledge does not endow the distributed components with the inconveniences of a centralized system (in terms of computation power and communication bandwidth requirements). One way to ensure that a minimal global coherence will be satisfied within the whole system is to enable communication between the vehicles of the system, up to a level that will warranty that the decision is globally coherent. One of the main advantages of the distributed

approach relies on its superior suitability to deal with the scalability of the system.

1.2. Communication and Networking

Recall that communication and networking also play an important role in the implementation of these techniques for multiple unmanned vehicles. Single vehicle communication systems usually have a single unique link between the UAV and the control station. The natural evolution of this communication technique toward multi-UAV configurations is the well-known star-shaped network configuration. While this simple approach to vehicles intercommunication may work well with small teams, it could not be practical or cost-effective as the number of vehicles grows. Thus, for example, in multi-UAV systems, there are some approaches of a wireless heterogeneous network with radio nodes mounted at fixed sites, on ground vehicles, and in UAVs. The routing techniques allow any two nodes to communicate either directly or through an arbitrary number of other nodes which act as relays [2].

When autonomous teams of UAVs should operate in isolated areas with any kind of infrastructure, using a mesh of ground stations to support communication between the mobile nodes is not possible. At this point is where ad hoc like networks take place, and the information exchanges occur only via the wireless networking equipment carried by the individual UAVs. Some autonomous configurations (such as close formation flying) result in relatively stable topologies. However, in others, rapid fluctuations in the network topology may occur when individual vehicles suddenly veer away from one another or when wireless transmissions are blocked by terrain features, atmospheric conditions, or signal jamming, or enemy attack. In spite of such dynamically changing conditions, vehicles in an autonomous team should maintain close communications with others in order to avoid collisions and facilitate collaborative team mission execution. In order to reach these goals, two different approaches have been adopted [2]:

- The first one is closer to the classical networks architecture, establishes a hierarchical structure and routes data in the classical down-up-down traversing as many levels of the hierarchy as needed to reach destination.
- The second one assist routing in such an environment is to use location information provided by positioning devices, such as global positioning systems such as GNSS (Global Navigation Satellite System), thus using what it is called location aware protocols.

These two techniques are compatible and can be mixed. For example, some of the levels in a hierarchical approach could be implemented using location aware methods.

CHAPTER 2. CLASSIFICATION

2.1. Multi-UAVs Architecures

2.1.1. Physical coupling Based

In this case, the UAVs are connected by physical links and then their motions are constrained by forces that depend on the motion of other UAVs. The lifting and transportation of loads by several UAVs lies in this category. The main problem is the motion-coordinated control, taking into account the forces constraints. From the point of view of motion planning and collision avoidance, all the members of the team and the load can be considered as a whole. As the number of vehicles is usually low, both centralized and decentralized control architectures can be applied.

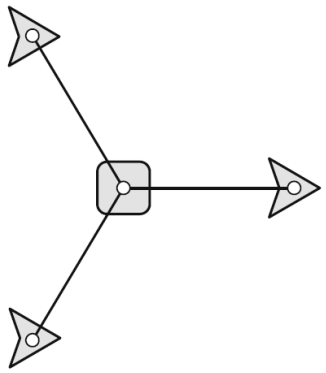

Figure 2.1: Representation of a physical coupling based architecture [2]

The transportation of a single object by multiple autonomous vehicles is a natural extension of the moving by several persons of a large and heavy object that cannot be handled by a single person. The coordinated control of the motion of each vehicle should consider the involved forces induced by the other vehicles and the load itself. For instance, in the case of several UAVs transporting a load using ropes, a force sensor in the rope can provide a measurement of the influence of the other UAVs and the load being transported. Each UAV could be controlled around a common compliance center attached to the transported object. Under the assumption that each UAV holds the object firmly with rigid links, the real trajectories of all of the UAVs are equal to the real trajectory of the object. However, in some transportation problems, this assumption cannot be applied, and the transported object moves with a dynamic behavior.

A suitable approach for the required coordinated control is the leader-follower scheme that will be more detailed in the next section. In this scheme, the desired trajectory is the trajectory of the leader. The followers estimate the motion of the leader by themselves through the motion of the transported object. Several examples of this approach can be found in the robotics community. The leader-follower scheme extended to multiple followers and to robots with non-holonomic constraints [47] has been implemented in an experimental system with three tracked mobile robots with a force sensor. In [48], the decentralized control

of cooperating mobile manipulators is studied with a designated lead robot being responsible for task planning. The control of each robot is decomposed (mechanically decoupled) into the control of the gross trajectory and the control of the grasp. The excessive forces due to robot positioning errors and odometry errors are accommodated by the compliant arms.

Lifting and transportation of loads by using multiple helicopters has been also a research topic for many years motivated by the payload constraints of these vehicles and the high cost of helicopters with significant payload. In addition, the use of multiple manned helicopters is also problematic and only simple operations, like load transportation with two helicopters, can be performed by extremely skillful and experienced pilots. The level of stress is usually very high, and practical applications are therefore rarely possible. Load transportation and deployment by one and several helicopters is very useful for many applications including the delivery of first-aid packages to isolated victims in disasters.

The autonomous lifting and transportation by two helicopters has been studied since the beginning of the nineties in [49] and [50]. Another study involving the simulation of a twin lift helicopter system and a real-time tool is detailed in [51]. More recent studies involve the use of autonomous quadrotors for transportation such as [52] and [53].

2.1.2. Formation Based

The vehicles are not physically coupled, but their relative motions are strongly constrained to keep the formation. Then, the motion planning problem can be also formulated considering the formation as a whole. Regarding the collision avoidance problem within the team, it is possible to embed it in the formation control strategy. Scalability properties to deal with formations of many individuals are relevant, and then, decentralized control architectures are usually preferred.

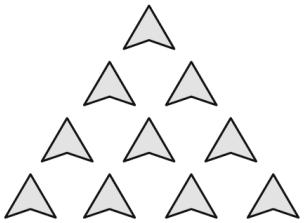

Figure 2.2: Representation of a formation based architecture [2]

In the formations, members of a group of vehicles must keep user-defined distances with the other group members. Those configurations can be either received via inter-vehicle communication or estimated using the sensors onboard. Formation control involves the design of distributed control laws with limited and disrupted communication, uncertainty, and imperfect or partial measurements.

Vehicle formation is a basic strategy to perform multi-vehicle missions including searching and surveying, exploration and mapping and active reconfigurable sensing systems. An added advantage of the formation architecture is that new members can be introduced to expand or upgrade the formation or to replace a lost member. The stability of the formation has been studied for a while, and researchers that proposed robust controllers to provide insensitivity to possibly large uncertainties in the motion of nearby agents and transmission delays in the feedback path.

The close formation flight control of homogeneous teams of fixed-wing UAVs airplanes is the most studied application for this kind of architecture. This large formation of small UAVs also offers benefits in terms of drag reduction and then increased payoffs in the ability to maintain persistent coverage of a large area. The most relevant studies in this topic are explained in [54] and [55], where both linear and nonlinear control laws are carefully tested in simulation environment. In order to bring those concepts into reality, the study presented in [56] tries to give a demonstration of two fixed-wing UAVs simultaneous flying with exactly the same flight plan.

Other cocepts in the formation based architecture are related to the leader-follower approach. In this approach, some vehicles in the formation are going to be designated as leaders that track predefined trajectories, while the followers track transformed versions of these trajectories according to given schemes. This also means that path planning only needs to be performed in the leader workspace. The leader-follower pattern is studied on unmanned helicopters [57], where the leader is commanded to fly on some predefined trajectories, and each follower is controlled to maintain its position in formation using the measurement of its inertial position and the information of the leader position and velocity, obtained through a wireless modem. Another relevant study applies to aircraft formation flights [58]. This research simulated the performance of a formation controller designed to have an inner and outerloop structure, where the outerloop guidance control laws minimized the forward, lateral, and vertical distance error by controlling the engine propulsion and generating the desired pitch and roll angles to be tracked by the innerloop controller. In the formation flight configuration, a radio control pilot maintained ground control of the leader aircraft, while the autonomous follower aircraft maintained a predefined position and orientation with respect to the leader aircraft. Other studies like [59] tried to simulate those controllers but also accounting collision avoidance between the members. Another study involving formation flights of aircraft considered that the leader UAV was given a predetermined flight plan, and the trajectory of the UAV was updated once per second in real time through the ground station to keep the follower at a fixed distance offset from the leader [60].

Other methods are based on a virtual leader, a moving reference point whose purpose is to manipulate the vehicle group behavior. The lack of a physical leader among the vehicles implies that any vehicle is interchangeable with any other in the formation. A solution based on a virtual leader approach combined with an extended local potential field is presented in [61] for formation flight and formation reconfiguration of small-scale autonomous helicopters. And for fixed-wing models, experimental results with YF-22 research aircraft can be found in [62].

Practical applications of formation control should include a strategy for obstacle avoidance

and reconfiguration of the formation. The avoidance of big obstacles could be performed by changing the trajectory of the whole formation to go around the obstacle [63]. If the obstacles are smaller than the size of the formation, the vehicles should be able to compromise the formation until the obstacle is passed. In order to do so, the obstacle avoidance behavior should be integrated in the control strategy of the individual members of the formation to avoid obstacles. Hybrid control techniques have been applied to avoid obstacles and solve the formation reconfiguration [64].

2.1.3. Swarm Based

They are homogeneous teams of many vehicles which interactions generate emerging collective behaviors. The resulting motion of the vehicles does not lead necessarily to formations. Scalability is the main issue due to the large number of vehicles involved, and then pure decentralized control architectures are mandatory.

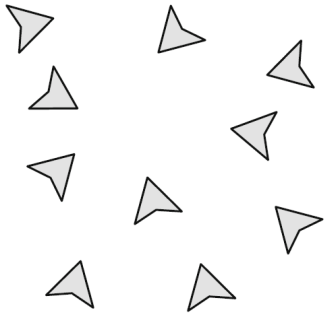

Figure 2.3: Representation of a swarm based architecture [2]

The key concept in the swarms is that complex collective global behaviors can arise from simple interactions between large numbers of unintelligent agents. This swarm cooperation is based on concepts from biology [65] and typically involves a large number of homogeneous individuals, with relatively simple sensing and actuation and local communication and control that collectively achieve a goal. This can be considered as a bottom-up cooperation approach. The agents execute the same program and interact only with other nearby agents by measuring distances and exchanging messages.

The concept of operations for a micro-UAV system is adopted from nature from the appearance of flocking birds and swarming bees. This bio-inspired motivation of swarms can be found, in [66], which describes an adaptive task assignment method for a team of fully distributed vehicles with initially identical functionalities in unknown task environments. This study employs a simple self-reinforcement learning model inspired by the behavior of social insects to differentiate the initially identical vehicles into specialists of different task types. At the end, this results in stable and flexible division of labor, and being capable of dealing with the cooperation problem of the vehicles engaged in the same type of task.

In [67] different mechanisms that allow populations of behavior-based robots to perform collectively tasks without centralized control or use of explicit communication are pre-

sented. Research made in [68] provides the results of implementing group behaviors such as dispersion, aggregation, and flocking on a team of robots. The study presented in [69] applies a rule-based, decentralized control algorithm that relies on constrained randomized behavior. Other considerations to take into account are the UAV restrictions on sensors, computation, and flight envelope which are evaluated in a simulation of an air vehicle swarm searching for and mapping a chemical cloud within a patrolled region. Another behavior based decentralized control strategy for UAV swarming by using artificial potential functions and sliding-mode control technique is presented in [70]. Individual interactions for swarming behavior are modeled using the artificial potential functions. For tracking the reference trajectory of the swarming of UAVs, a swarming center is considered as the object of control. Research in [71] tries to describe a multiagent-based prototype system that uses swarming techniques inspired from insect colonies to perform automatic target recognition using UAVs in a distributed manner within simulated scenarios. In another study a swarm of UAVs is used for searching one or more evading targets, which are moving in a predefined area while trying to avoid a detection by the swarm [72]. By arranging themselves into efficient geometric flight configurations, the UAVs optimize their integrated sensing capabilities, enabling the search of a maximal territory.

2.1.4. Intentional Cooperation Based

When describing the swarm based architecture, it only deals with homogeneous teams without explicit consideration of tasks decomposition and allocation, performance measures, and individual efficiency constraints of the members of the team. In order to address those drawbacks, intentional cooperation based architecture was defined.

According to [73], the UAVs belonging to team move according to trajectories defined by individual tasks that should be allocated to perform a global mission. These UAV trajectories typically are not geometrically related as in the case of the formations. In this case, problems such as multi-UAV task allocation, high-level planning, plan decomposition, and conflict resolution should be solved, taking into account the global mission to be executed and the different UAVs involved. In this case, both centralized and decentralized decisional architectures can be applied.

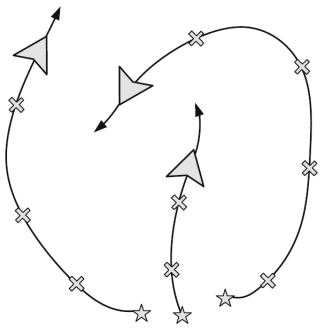

Figure 2.4: Representation of a intentional cooperation based architecture [2]

In the intentional cooperation approaches, each individual executes a set of tasks (subgoals that are necessary for achieving the overall goal of the system and that can be

achieved independently of other subgoals) explicitly allocated to perform a given mission in an optimal manner according to planning strategies. The UAVs cooperate explicitly and with purpose but also has the limitation of independent subgoals, where if the order of task completion is mandatory, additional explicit knowledge has to be provided to state ordering dependencies in the preconditions. It is also possible to follow a design based on collective interaction, in which entities are not aware of other entities in the team [74].

Most important factors in these systems include determining which UAV should perform each task so as to maximize the efficiency of the team and ensuring the proper coordination among team members to allow them to successfully complete their mission. In [75] a domain-independent taxonomy for the multiagent task allocation problem is presented. In the last years, a popular approach to solve this problem in a distributed way is the application of market based negotiation rules.

Once the tasks have been allocated, the next step consists on the complete coordination of vehicle motions, and is usually done by means of suitable multi-vehicle path/velocity planning strategies. The main purpose is to avoid potential conflicts among the different trajectories when sharing the same working space. Formal approaches to the collision avoidance problem and different approaches that can be applied to solve it can be found in [76].

In the case of teams composed by heterogeneous members, there are lots of challenging aspects, to consider. In [77] the current state of the technology, existing problems, and potentialities of platforms with multiple heterogeneous UAVs are studied. This heterogeneity is divided in 2 parts:

- UAV platforms looking to exploit the complementarities of the aerial vehicles, such as helicopters and airship.
- Information-processing capabilities onboard, ranging from pure remotely operated vehicles to fully autonomous aerial robots.

In [78] a distributed architecture for the autonomous coordination and cooperation of multiple UAVs for civil applications is presented. The architecture is endowed with different modules that solve the usual problems that arise during the execution of multipurpose missions, such as task allocation, conflict resolution, and complex task decomposition. One of the main objectives in the design of the architecture was to impose few requirements to the execution capabilities of the autonomous vehicles to be integrated in the platform. Basically, those vehicles should be able to move to a given location and activate their payload when required. Thus, heterogeneous autonomous vehicles from different manufacturers and research groups can be integrated in the architecture developed, making it easily usable in many multi-UAV applications. The software implementation of the architecture was tested in simulation and finally validated in field experiments with four autonomous helicopters. The validation process included several multi-UAV missions for civil applications in a simulated urban setting:

- Surveillance strategies.

- Cooperative searching.
- Fire detection, monitoring and extinguishing.
- Load transportation and deployment.
- People tracking.

Cooperative perception can be considered as an important tool in many applications based on intentional cooperation schemes. It can be defined as the task of creating and maintaining a consistent view of a world containing dynamic objects by a group of agents each equipped with one or more sensors. In particular, cooperative perception based on artificial vision has become a relevant topic in the multi-robot domain, mainly in structured environments [79]. In [80] cooperation perception methods for multi-UAV system are studied, where each UAV extracts knowledge from its sensors by applying individual perception techniques, and the overall cooperative perception is performed by merging the individual results. This approach requires knowing the relative position and orientation of the UAVs. In many outdoor applications, it is assumed that the position of all the UAVs can be obtained by means of GPS and broadcasted through the communication system. However, if this is not the case, the UAVs should be capable of identifying and of localizing each other [81] which could be difficult with the onboard sensors. Another approach consists of identifying common objects in the scene. Then, under certain assumptions, the relative pose displacement between the vehicles can be computed from these correspondences. In the ANSER project studied in [82], a decentralized sensor data fusion using multiple aerial vehicles is also researched and experimented with fixed-wing UAVs with navigation and terrain sensors.

Part III

SWARM COMMUNICATIONS

CHAPTER 1. INTRODUCTION

1.1. Communication and Networking

As we have seen in UAV Architecture part, communication and networking are one of the most crucial aspects when implementing of swarm of unmanned aerial vehicles. Single vehicle communication systems usually have a direct link between the vehicle and the control station. When dealing with multiple UAVs, the traditional communication between single UAV and ground station turns into star-shaped network configuration [2].

The star-shaped network is an excellent configuration for working with small teams, however, cost-effectiveness concerns arises as the number of UAVs grow in numbers. Some alternatives to overcome those problems are wireless heterogeneous network with radio nodes mounted at fixed sites, both on ground vehicles and UAVs. The routing techniques allow any two nodes to communicate either directly or through an arbitrary number of other nodes which act as relays.

In many occasions, it is not possible to maintain a fixed infrastructure on ground, specially when operating on remote regions. In those cases, networks could be formed in an ad hoc fashion, where information is exchanged only via the wireless networking equipment carried by each UAV on the network. Some autonomous configurations (such as close formation flying) result in relatively stable topologies.

It should be noticed that rapid fluctuations in the network topology can be a serious concern to deal with. This can be caused when individual vehicles suddenly veer away from one another or when wireless transmissions are blocked by terrain features, atmospheric conditions, signal jamming, or enemy attack. Vehicles in an autonomous team should maintain close communications with others in order to avoid collisions and facilitate collaborative team mission execution. In order to reach these goals, two different approaches have been adopted. The first one establishes a hierarchical structure and routes data in the classical down-up-down traversing as many levels of the hierarchy as needed to reach destination. The second one consist on taking advantage of location information provided by positioning devices such as GNSS. These two techniques are compatible and can be mixed. For example, some of the levels in a hierarchical approach could be implemented using location aware methods.

CHAPTER 2. PROTOCOLS

2.1. Ad-hoc networks

2.1.1. Fundamentals

The idea of ad hoc networking goes back to the US Defense Advanced Research Projects Agency (DARPA) packet radio network, which was used in the 1970s [83]. A mobile ad hoc network is a collection of mobile devices establishing a short-lived or temporary network in the absence of a supporting structure. Mobile ad hoc networks can be used in establishing efficient, dynamic communication for rescue, emergency, and military operations. A commercial application, such as Bluetooth, is one of the recent developments utilizing the concept of ad hoc networking.

An ad hoc network can be conceived as a collection of wireless mobile nodes which are constantly changing a network without the use of any existing network infrastructure. Those routers have the ability to move randomly and organize themselves in an arbitrary manner. Multihop, mobility, and large network size combined with device heterogeneity, bandwidth, and contrained energy supply make the design of adequate routing protocols a major challenge. Some form of routing protocol is in general necessary in such an environment, since two hosts that may wish to exchange packets might not be able to communicate directly [83].

Mobile users will want to communicate in situations in which no fixed wired infrastructure is available. In those situations, a collection of mobile hosts with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. Such networks have received considerable attention in recent years in both commercial and military applications, due to the attractive properties of building a network on the fly and not requiring any hard-wired infrastructure such as base station or central controller.

Some of the challenges that ad hoc networks are facing are limited wireless transmission range, hidden-terminal problems, packet losses due to transmission errors, mobility-induced route changes and battery constraints. Mobile ad hoc networks could enhance the service area of access networks and provide wireless connectivity into areas with previously poor or no coverage. Connectivity to wired infrastructure will be provided through multiple gateways with possibly different capabilities and utilization. To improve performance, the mobile host should have the ability to adapt to variation in performance and coverage and to switch gateways when this would be beneficial. To enhance the prediction of the best overall performance, a network-layer metric has better overview of the network.

Ad hoc networking brings features like easy connection to access networks, dynamic multihop network structures, and direct peer-to-peer communication. The multihop property of an ad hoc network needs to be bridged by a gateway to the wired backbone. The gateway must have a network interface on both types of networks and be a part of both the global routing and the local ad hoc routing. Users could benefit from ubiquitous networks in several ways. User mobility enables users to switch between devices, migrate sessions, and

still get the same personalized services. Host mobility enables the users' devices to move around the networks and maintain connectivity and reachability.

2.1.2. Applications

The following are the applications of ad hoc wireless networks [83]:

- Community network
- Enterprise network
- Home network
- Emergency response network
- Vehicle network
- Sensor network
- Education
- Entertainment
- Coverage extension
- Commercial and civilian environments

As we have previously said in section 2.1.1., wireless ad hoc or on-the-fly networks are characterized by the lack of infrastructure. Nodes in a mobile ad hoc network are free to move and organize themselves in an arbitrary fashion. A connection between 2 users may have multiple links between them, making any network completely heterogeneous. This allows an association of various links to be a part of the same network.

Ad hoc networks are suited for use in situations where an infrastructure is unavailable or to deploy one is not cost effective. One of many possible uses of mobile ad hoc networks is when the need for collaborative computing is present. Work has been going on to introduce the fundamental concepts of game theory and its applications in telecommunications. The cooperation of the users is necessary to the operation of ad hoc networks; therefore, game theory provides a good basis to analyze the networks.

A mobile ad hoc network can also be used to provide applications for crisis management services, where the entire communication infrastructure is destroyed. By using a mobile ad hoc network, an infrastructure could be set up in hours instead of weeks, as is required in the case of wired-line communication.

2.2. Flying Ad-hoc neworks

Flying Ad-hoc Networks (FANETs) are a particular case inside Ad-hoc networks, where its main field of application is on UAV network systems. FANETs architecture guarantees that all UAVs can communicate with each other and to the base station in real-time without having any pre-defined fixed infrastructure. One of the most important features is the reliability of the network, where a UAV that has lost its connection recently, can still reconnect to the network through other UAVs. All the previous features have been explained in ad hoc networks 2.1.1., however there are some challenges present when dealing with high speed vehicles like aircraft. Those include fast movements in a free space that must hold enough separation distance to avoid collisions.

A communication architecture identifies how information exchange between the base station and a UAV or between the UAVs. In FANETs architecture, UAVs render real-time communication in ad hoc manner that can abolish the need for the infrastructure and it rectifies the communication range constraint. FANETs architecture performs significant role in scenarios where real-time communication and range constraints are main issues, and where it is difficult to provide an infrastructure. In FANETs, as the UAVs connect and detach to the network frequently and, hence, ad hoc network between the UAVs have been found to be the best solution in most cases. Additionally, for quick and robust communication between the UAVs, decentralized communication architecture is more suitable. There are several different communication architectures proposed for multi-UAV systems, in which we introduce three communication architectures for FANETs [84].

2.3. Routing protocols

The purpose of this section is to review all the available routing protocols that suits to FANETs requirements.

2.3.1. Static Routing Protocols

In static routing protocols, each UAV has a routing table that remains the same all time. Static routing protocols are applicable in cases when the topology of the network does not change and also when the choices in route selection are limited. Here, each UAV communicate with other the UAVs or the ground station and stores their information only. This leads in reducing the number of communication links. However, in case of a failure for updating the routing table, it is compulsory to wait until the mission is completed. As a result, there protocols are not fault tolerant.

2.3.1.1. Load Carry and Deliver Routing

In the Load Carry and Deliver Routing (LCAD) [85] model, a UAV store data from a source ground node, convey these valuable data by flying to a destination ground node. This routing mechanism is feasible for bulk data transfer and delay tolerant applications with minimum hops [86]. The main objectives of LCAD routing is to maximize throughput and

increase the security. However, the main downside of this protocol is related to increasing distance between the communicating UAVs, where delay becomes exceptionally large and intolerable. To decrease the transmission delay, multiple UAVs can be used on the same path, where distance among the UAVs must be minimum and speed of UAVs can be increased. Another option is to divide a LCAD network into smaller sub-networks.

2.3.1.2. Multilevel Hierarchical Routing

Multilevel Hierarchical Routing (MLH) routing protocol is designed to deal with the network scalability issue [87]. Here, the network can be grouped into a number of clusters designated in different operation areas. Each cluster has its own cluster head (CH), which describe the entire cluster and also has connections outside cluster with other clusters. Those clusters have a set of properties:

- Each cluster in the MLH network can have multiple tasks assigned.
- All the UAVs in a cluster are within the direct communication range of the CH.
- The CH is directly or indirectly connected with the upper layer UAVs or satellites.

MLH can have better performance repercussions if the UAVs are arranged in different swarms with large operation area, and multiple UAVs are deployed in the network. However, the most crucial design issue for MLH routing is the cluster information. The high mobility of the UAVs may require frequent cluster information, so mobility prediction clustering address to solve this issue with the help of the dictionary Trie structure prediction algorithm [88] and link expiration time mobility model. In this model, the highest weighted UAV among its neighbors is chosen as the cluster head. The CH selection criteria can enhance the stability of the clusters and the Cluster Heads (CHs). Clustering algorithm for UAV networks is presented in [89]. It designate the clusters on the ground first, and then keep updating it during the operation. Ground clustering determines the clustering plan, and then select the CHs based on the geographical information. Moreover, just after the deployment of UAVs, the cluster structure is calibrated according to the mission updates. This routing model can significantly enhance the stability and ensure the artistry of dynamic networking.

2.3.1.3. Data Centric Routing

Data Centric Routing (DCR) can also be implemented on FANETs, where data is requested and collected with reference to the data characteristics instead of sender or receiver IDs. This routing algorithms can be choose when the data request is generated by a number of UAVs, and data distribution is done by on-demand algorithms. DCR can be used in FANETs to provide numerous types of applications on homogeneous multi-UAV systems in order to accumulate explicit data from a specific mission area. Publish-subscribe model is usually valid for this sort of architecture [90]. It connects automatically data producers (publishers) with data consumers (subscribers). The producer node adopts which information need to be published and then starts data dissemination. After reaching the published data to a UAV in the network, it try to find the subscription messages on it and then forwards that data towards the intended UAV. Data-centric routing algorithms are decoupled in three dimensions:

- Space decoupling: Communicating UAVs can be anywhere and knowing each other's ID or location is not mandatory.
- Time decoupling: Communicating UAVs are not required to be online simultaneously and data can be forwarded to the subscribers instantly or later.
- Flow decoupling: Data delivery can be accomplished reliably by asynchronous communication structure. This routing model can be preferred when the number of UAVs are less in the network, and where the UAVs have a predetermined flight-plan, which involves minimum assistance between the clusters

2.3.2. Proactive Routing Protocols

In the proactive routing protocol (PRP) every node periodically maintains one or more tables indicating the complete topology of the network. This routing protocol has advantage of having routes immediately accessible when needed. However, it suffers additional overhead cost because of preserving up-to-date information and as a consequence throughput of the network may be influenced since control messages are sent out unnecessarily even when there is no data traffic. This makes the proactive routing protocol a bad choice for highly dynamic mobile and large UAV networks. Second, also for cases when the network topology change or connection failure occurs, these routing protocols shows a slow reaction to failure.

2.3.2.1. Destination- Sequenced Distance Vector

It is based on Bellman-Ford algorithm with little modification required by making it more suitable for UAV adhoc networks. In DSDV each UAV must distinguish everything about all of the other UAVs connected in the network [91]. The routing table here is periodically updated about the entire network with sequence number to avoid routing loops.The recently used route with highest sequence number is given preference over on a route with lowest sequence number. The main benefits of DSDV are both the simplicity and the usage of the sequence numbers, which promises the loop free data transmission. However, the main drawback of this routing algorithm is the periodic updating of up-to-date routing table, which creates an overhead to the network. This protocol is not suitable for highly dynamic networks where topology changes more frequently. Also it support single path routing and does not support multipath routing.

2.3.2.2. Optimized Link State Routing

In Optimized Link State Routing (OLSR) routes are continuously stored and updated in tables [92]. When a route is required, the protocol tries to determine all the possible destinations without any initial delay. With the aim of establishing a communication process between the UAVs in the network running a protocol instance, OLSR uses a unique packet, which comprises of more than one message. OLSR packets can carry three different type of messages, each one for a specific purpose:

- HELLO message, which is transmit periodically to find connectivity with neighbor, link sensing, and MPR signaling
- Topology Control (TC) message, which advertise to maintain link states information and Multiple Interface Declaration (MID) message, which accomplish the multiple interface declaration on a node.

This periodic flooding behavior results in the form of large overhead. With the use of MPRs mechanism in OLSR, the message overheads can be decreased and thus improving the latency. Because the MPR UAV can only forward the messages during the flooding process. The sender UAV specifies a set of MPR UAVs so that the MPR UAVs can cover two hop neighbors. A UAV which selects another UAV as a MPR UAV member is called MPR selector of that node. However, the most significant design parameters for OLSR is the number of MPRs, which influences the delay intensely. Apparently, as the number of MPR shrinks, the overhead will reduce consequently. On this way, a new method is suggested for reducing the number of members of MPR. For each sending data packets, the sender UAV computes the distance to the receiver UAV, then if the distance is larger than the maximum distance that can be attained by using the directional antenna, or also if the Omni-directional antenna cannot achieve the destination, the UAV will apply the DOLSR algorithm Otherwise, OLSR will be used usually [93].

2.3.3. Reactive Routing Protocols

The reactive routing protocol (RRP) discovers or maintains a route on demand. The routing table here is periodically updated, when there is some data to send, if there is no connection between two nodes, there is no need to calculate a route between them. Thus these routing protocols retain the routes only that are presently in use [94], therefore as a consequence it overcome the overhead problem of RRP. In this routing model, there are two types of messages generated:

- Route request: It is transmitted from the source UAV to all adjacent UAVs using flooding mechanism to discover the path, and each UAV uses the same approach until it reaches the destination UAV.
- Route reply: It is initiated by the destination UAV and goes to the source UAV using the unicast communication mode.

In this routing approach, there is no need to refresh all tables in the network. Reactive routing protocols are bandwidth efficient, because there is no periodic updates. The main shortcoming of RRP is that it takes long time to find the route and as a result, high latency may occur in the network during the optimal route finding process.

2.3.3.1. Dynamic Source Routing

Dynamic Source Routing (DSR) protocol [95] allows a network to be self-configuring, self-organizing and without the need for having any available infrastructure. The main reason to consider DSR in front of other protocols is its reactive nature, allowing it to be used for

multi-hop wireless mesh networks. In DSR the source only tries to find a path to a destination in a scenario, whenever it has data to send. DSR is more suitable than proactive methods for FANETs, where the mobility of the UAVs is high, and the topology is not stable. Updating a routing table by a proactive methods is not that much optimal due to the high mobility of the UAVs. However, repetitive path finding by reactive method before each packet delivery can also be exhaustive.

2.3.3.2. Ad Hoc On-Demand Distance Vector

Ad Hoc On-Demand Distance Vector (AODV) is the enhanced version of both DSDV and DSR routing protocols, studied in [96] and [97]. It inherits periodic updates from DSDV and hop-to- hop routing from DSR. Due to the reactive behavior, AODV discover a route only when it is desired and does not retain routes to destination that are not active in the communication process. AODV routing protocol comprises of three phases:

- Route discovery
- Packet transmitting
- Route maintaining

Whenever a source UAV wishes to send a packet, it first initiates a route discovery operation to detect the location of the intended UAV and then forward packet over a determined route without having a loop during packet transmitting phase. Route maintenance phase take place to restore link failure. This routing protocol uses a sequence number to find an up-to-date optimal route towards the destination. An expiration time is used in order to maximize route's freshness. In this method, intermediate UAVs also update their routing tables. However, network congestion is an issue with AODV due to the high dynamic nature of FANETs system.

2.3.3.3. Time-slotted on-demand Routing

Time-slotted on-demand routing protocol [98] is basically time-slotted version of Ad-hoc On-demand Distance Vector Routing (AODV). AODV sends its control packets on random-access mode, whereas time-slotted on-demand protocol practices dedicated time slots in which only one UAV can send data packet. This routing method not only increase the usable bandwidth efficiency, but also avoid the packet collisions and increases the packet delivery ratio.

2.3.4. Hybrid Routing Protocols

The Hybrid Routing Protocol (HRP) is a combination of both proactive and reactive routing protocols, taking the best features and to overcome the limitations from both worlds. Reactive routing protocols generally needs extra time to discover route and proactive routing protocols has huge overhead of control messages. These shortcomings can be mitigated by using HRP. Hybrid protocols are especially appropriate for large networks, and is based on the concepts of zones where intra-zone routing is executed with the help of proactive

routing and inner-zone routing is achieved using the reactive routing approach.

2.3.4.1. Zone Routing Protocol

Zone Routing Protocol (ZRP) [99] is basically based on the abstract concept of zones which made it suitable for large network spans and diverse mobility patterns. In this routing approach each UAV has been assigned to an specific zone. The size of the zone is determined by a radius of a determined length which is based on the number of UAVs to the perimeter of the zone and can be controlled by adjusting the transmission power of each UAV. The routing within the zone is called as intra-zone routing. The intra-zone routing uses proactive routing approach to maintain the routes. If the source and destination UAVs are available in the same zone, the source UAV can start data communication immediately. The inter-zone routing is responsible for sending data packets to outside of the zone, and it utilizes reactive routing approach to maintain and finding the optimal routes. The delay caused by the route discovery is minimized by using bordercasting. Reply messages are only generated by border UAVs of a zone. The border UAVs then repeat either by selecting inter- or intra-zone routing as required.

2.3.4.2. Temporarily Ordered Routing Algorithm

Temporarily Ordered Routing Algorithm (TORA) is a highly adaptive on-demand routing protocol, suitable for multihop networks [100]. When using this routing protocol, each UAV on the network only update routing information about adjacent UAVs. The key features of using this routing algorithm is to limit the propagation of control message in a highly mobile environment in order to minimize quick reactions to topological changes. It erases invalid routes and searches for new routes in a single-pass of the distributed algorithm. Particularly, TORA uses reactive routing protocols, but it also use proactive approaches in some cases. It construct and maintains a Directed Acyclic Graph (DAG) from the source to destination UAV. There are several routes among these UAVs in DAG. It is preferred for quick computing of new routes in case of disconnected links and for enhancing adaptability. TORA is not based on the shortest path algorithm, longer routes are normally used to minimize network overhead. Each UAV has a parameter value known as height in DAG, and no two UAVs have the same height value. Data flow from the higher UAVs to lower UAVs like top-down approach. It offers loop-free routing, because of no data flow towards the higher height UAVs. In the route discovery process, this height parameter is returned to the requesting UAV, and in this approach all the intermediate UAVs maintain their routing tables according to the incoming routes and heights information.

2.3.5. Geographic Based Routing Protocols

Position-based routing protocols have been proposed to assume knowledge of the geographical position information of UAVs to supports efficient routing [101]. In this type of protocols, they assume that the source UAV knows about the physical position of the communicating UAVs and sends message to the destination UAVs without route discovery. Generally, each UAV determines its own position with the help of GPS system or any

other type of positioning facility. This routing algorithm is primarily inspired by two subjects:

- A position facility is normally used by the sender of a packet to locate the physical position of the receiver.
- A forwarding approach is used to forward data packets to the intended UAV.

2.3.5.1. Greedy Perimeter Stateless Routing

Greedy Perimeter Stateless Routing (GPSR) [102], which is also a position-based protocol, having better performance comparatively to proactive and reactive routing algorithms. Some simulations have shown that greedy geographic based forwarding routing protocols are applicable for densely deployed UAVs Network. However, the reliability of the network can be a severe issue in situation of sparse deployments. A combination of other mechanisms, such as face routing, should be used for the applications that target 100% reliability. However, initially for FANETs deployments the existing MANET routing algorithms have tested, in which most are not well suited for FANETs, due to the UAV distinct obstruction such as rapid variation in the link quality and very high mobility of the flying nodes.

2.3.5.2. Geographic Position Mobility Oriented Routing

Geographic Position Mobility Oriented Routing (GPMOR) [103]. The conventional position based solutions only depend on the location information of the UAVs. However, GPMOR also figure out the movement of UAVs with Gaussian-Markov mobility model, and it utilizes this data to locate the next hop. It is investigated that this routing mechanism can provide data forwarding effectively with reference to packet delivery ratio and latency.

2.3.6. Hierarchical Routing Protocols

In hierarchical routing protocols the ability of choosing proactive and of reactive routing rely on the hierarchical level of the network in which a UAV resides. This specific routing is primarily determined with some proactive planned routes and then helps the request from by triggered nodes through reactive protocol at the lower levels. The main downside of this protocol include complexity and scheme of addressing which response to traffic request and as a repercussion it drape the interconnecting factors.

2.3.6.1. Mobility Prediction Clustering Algorithm

Mobility Prediction Clustering Algorithm (MPCA) proposed for UAV networking [104] based on the attributes of UAVs. It governs on the dictionary of trie-structure prediction algorithm and link expiration time to solve the issues regarding high mobility of the UAVs.The main advantage of this algorithm is to reduce the instability of clustering and improves the network performance.

2.3.6.2. Clustering Algorithm of UAV networking

Clustering algorithm is based to overcome the difficulty of managing out of sight UAVs. In case of multi UAV system, it construct clusters on the ground, and then adjust it in the space to improve the stability and flexibility of near-space clusters [89].

Part IV

SWARM ALGORITHMS

CHAPTER 1. NAVIGATION

1.1. Inertial Navigation

1.1.1. Fundamentals review

An inertial navigation system (INS) is the most commonly used dead-reckoning navigation system. The current position of any craft can be determined by integrating velocity, which is also determined by the integration acceleration measurements obtained using an Inertial Measurement Unit (IMU). An attitude solution is also maintained by integrating the IMU's angular rate measurements. Following initialization, navigation can proceed without further information from the environment. One of the most important strengths of inertial navigation systems is that they are self-contained.

Inertial navigation has been used since the past decades for applications such as aviation, submarines, ships, and guided weapons. These systems can typically operate either stand-alone or as part of an integrated navigation system. For newer applications, such as light aircraft, helicopters, unmanned air vehicles (UAVs), land vehicles, mobile mapping, and pedestrians, low-cost sensors are typically used and inertial navigation forms part of an INS/GNSS or multisensor integrated navigation system.

An INS comprises an inertial measurement unit and a navigation processor, as shown in figure 1.1. The IMU is in charge of measuring specific force and angular rate using a set of accelerometers and gyroscopes. The navigation processor may be packaged with the IMU and the system sold as a complete INS. Alternatively, the navigation equations may be implemented on an integrated navigation processor or on the application's central processor. Marine, aviation, and intermediate grade inertial sensors tend to be sold as part of an INS, while tactical grade inertial sensors are usually sold as an IMU. In either case, the function is the same, so the term inertial navigation system is applied here to all architectures in which a three-dimensional navigation solution is obtained from inertial sensor measurements.

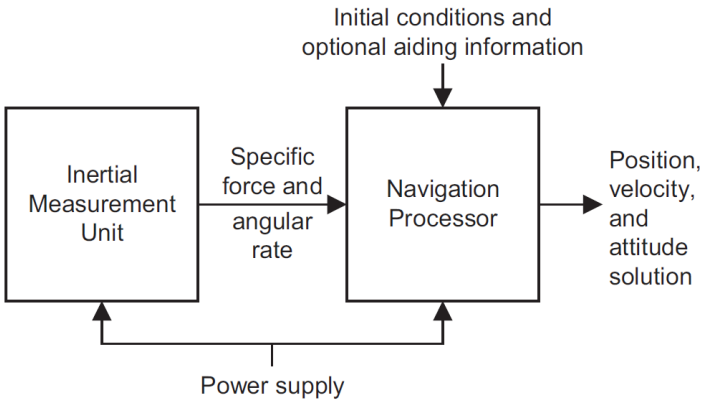

Figure 1.1: Inertial Navigation System schematic [3]

1.1.2. General equations derivation

This section only tries to explain the most basic equations related to inertial navigation integration. For more in-deep analysis refer to [3] where more complex equations about INS are derived for multiple cases.

A body *b* is constrained to move with respect to an Earth-fixed reference frame *p* in a straight line perpendicular to the direction of gravity. The body's axes are fixed with respect to frame *p*, so its motion has only one degree of freedom. Its Earth-referenced acceleration may be measured by a single accelerometer with its sensitive axis aligned along the direction of motion (neglecting the Coriolis force). If the speed, v_{pb}, is known at an earlier time, t_0, it may be determined at a later time *t*, simply by integrating the acceleration, a_{pb}:

$$v_{pb}(t) = v_{pb}(t_0) + \int_{t_0}^{t} a_{pb}(t)\,dt \tag{1.1}$$

Similarly, if the position r_{pb} at time t_0 is known, its value at time *t* may be obtained by integrating the velocity:

$$r_{pb}(t) = r_{pb}(t_0) + \int_{t_0}^{t} v_{pb}(t)\,dt \tag{1.2}$$

Two accelerometers are required to measure the acceleration along two orthogonal axes. However, their sensitive axes will be aligned with those of the body, *b*. To determine the acceleration along the axes of frame *p*, the heading of frame *b* with respect to frame *p*, Ψ_{pb}, is required. The rotation of the body with respect to the reference frame may be measured with a single gyro sensitive to rotation in the horizontal plane (neglecting Earth rotation). Thus, three inertial sensors are required to measure the three degrees of freedom of motion in two dimensions. If the heading Ψ_{pb} is known at the earlier time t_0, it may be determined at the later time *t* by integrating the angular rate, $\omega^b_{pb,z}$:

$$\Psi_{pb}(t) = \Psi_{pb}(t_0) + \int_{t_0}^{t} \omega^b_{pb,z}(t)\,dt \tag{1.3}$$

The accelerometer measurements may be transformed to the *p*-frame resolving axes using a 2x2 coordinate transformation matrix:

1.2. Satellite Based Navigation

1.2.1. Fundamentals review

Global Navigation Satellite Systems (GNSS) are a kind of self-positioning navigation systems that provide a 3 dimensional positioning solution by passive ranging using radio signals transmitted by a constellation of orbiting satellites. The position solution is calculated by the user equipment, which does not transmit any signals for positioning purposes.

A number of systems aim to provide global coverage. The most well-known is the Navigation by Satellite Ranging and Timing (NAVSTAR) Global Positioning System, owned and operated by the United States government and usually known simply as GPS. The Russian GLONASS is also fully operational. At the time of this writing, the European Galileo system is finishing its deployment, while a global version of the Chinese Beidou system

is under final development. In addition, a number of regional satellite navigation systems enhance and complement the global systems.

Some authors use the term GPS to describe satellite navigation in general, while the term GNSS is sometimes reserved for positioning using signals from more than one satellite navigation system. Here, the term GPS is reserved explicitly for the NAVSTAR system, while the term GNSS is used to describe features common to all of the systems. Similarly, the terms GLONASS, Galileo, Beidou, and so forth are used to describe features specific to those systems.

1.2.2. Architecture

The architecture of a satellite navigation system consists of three components 1.2: the space segment, the control or ground segment, and the user segment, which, in turn, comprises multiple pieces of user equipment. Each GNSS has its own independent space and control segments, whereas user equipment may use signals from one, two, or multiple GNSS.

The space segment comprises the satellites, collectively known as a constellation, which broadcast signals to both the control segment and the users. Some authors use the term space vehicle instead of satellite. A typical GNSS satellite has a mass of around 1000 kg and is about 5m across, including solar panels. Each fully operational constellation comprises at least 24 satellites. By 2020, there were more than 100 GNSS satellites in orbit.

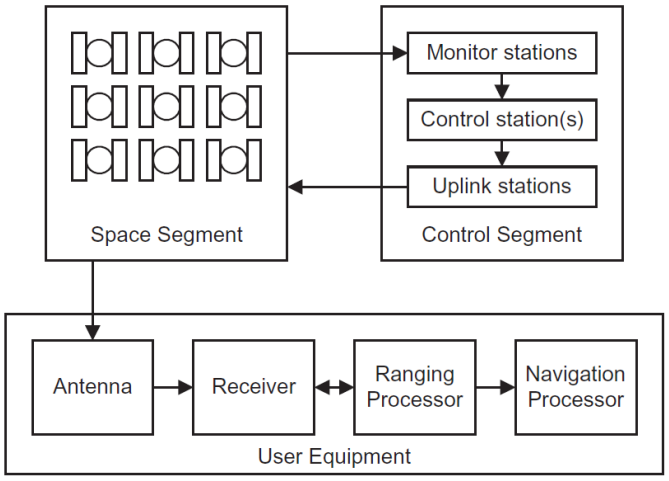

Figure 1.2: GNSS system architecture [3]

GPS, GLONASS, Galileo, and Beidou satellites are distributed among a number of medium Earth orbits (MEOs). GPS satellites orbit at a radius of 26,580 km, perform two orbits per sidereal day. To provide suitable signal geometry for 3-D positioning, the satellites in each constellation must be distributed across several nonparallel orbital planes. Therefore, in contrast to the equatorial orbits of geostationary satellites, GNSS orbital planes are inclined with respect to the equator (at 55° for GPS). This also provides better coverage in

polar regions. With a clear line of sight, between 5 and 14 transmitting GPS satellites are visible at most times. More satellites are visible in equatorial and polar regions than at mid-latitudes.

GNSS satellites broadcast multiple signals on several frequencies. These can incorporate both ranging codes and navigation data messages. The ranging codes enable the user equipment to determine the time at which the received signals were transmitted, while a data message includes timing parameters and information about the satellite orbits, enabling the satellite positions to be determined. A number of atomic clocks aboard each satellite maintain a stable time reference.

The control segment, or ground segment, consists of a network of monitor stations, one or more control stations, and a number of uplink stations. The GPS control segment comprises 16 monitor stations, 12 uplink stations, and two control stations. The monitor stations obtain ranging measurements from the satellites and send these to the control stations. The monitor stations are at precisely surveyed locations and have synchronized clocks, enabling their ranging measurements to be used to determine the satellite orbits and calibrate the satellite clocks. Radar and laser tracking measurements may also be used.

The control stations calculate the navigation data message for each satellite and determine whether any maneuvers must be performed. This information is then transmitted to the space segment by the uplink stations. Most satellite maneuvers are small infrequent corrections, known as station keeping, which are used to maintain the satellites in their correct orbits. However, major relocations are performed in the event of satellite failure, with the failed satellite moved to a different orbit and a new satellite moved to take its place. Satellites are not moved from one orbital plane to another.

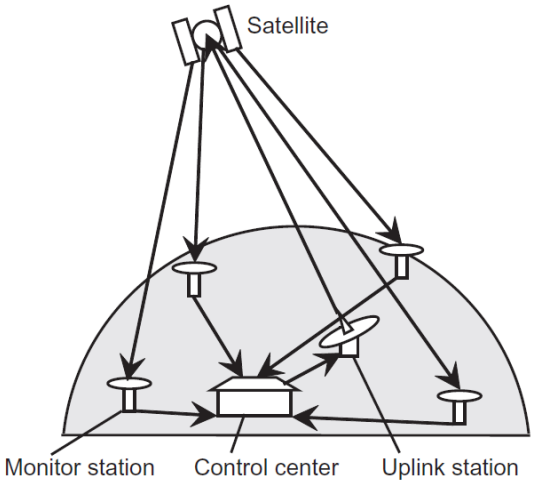

Figure 1.3: Control segment operation [3]

Nearly all GNSS user equipment receives either GPS signals alone or GPS together with one or more of the other systems. GNSS user equipment is commonly described as GPS, GLONASS, Galileo, Beidou, and GNSS receivers, as appropriate. However, the receiver

forms only part of each set of user equipment. The antenna converts the incoming GNSS radio signals to electrical signals. These are input to the receiver, which demodulates the signals using a clock to provide a time reference. The ranging processor uses acquisition and tracking algorithms to determine the range from the antenna to each of the satellites used from the receiver outputs. It also controls the receiver and decodes the navigation messages. Lastly, the navigation processor uses the ranging measurements to compute a position, velocity, and time solution.

1.2.3. Positioning

A GNSS position solution is determined by passive ranging in three spatial dimensions. Using a range measurement from a single satellite signal, the user position can be anywhere on the surface of a sphere of radius r centered on that satellite, which is also known as surface of position (SOP). When ranges to two satellites are used, the locus of the user position is the circle of intersection of the surfaces of two spheres of radii r_{a1} and r_{a2}. Adding a third range measurement limits the user position to two points on that circle as depicted in figure 1.4.

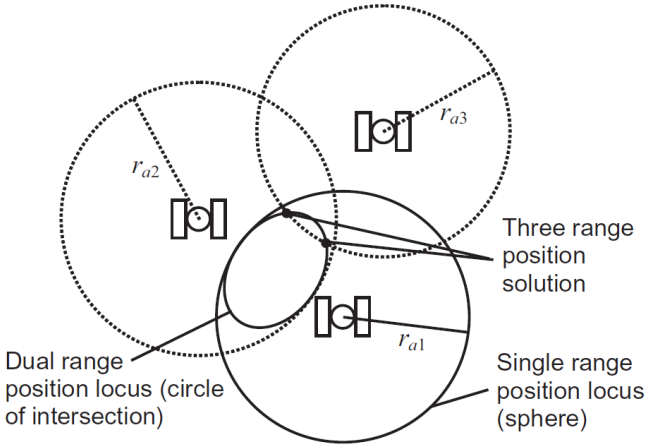

Figure 1.4: Positioning from multiple satellites [3]

The previous figure is just an schematic concept of how positioning works, but in real life only one position solution will be viable in practice. Apart from this, 4 satellites are required (and not 3 as shown in figure 1.4) to calculate the complete navigation solution accounting receiver's clock error. When both solutions are viable, an additional ranging measurement can be used to resolve the ambiguity. In GNSS, the receiver and satellite clocks are not synchronized. Also, the measurements made are pseudo-range, not range.

1.3. INS/GNSS Integration

1.3.1. Fundamentals review

As we have explained in 1.1., inertial navigation had a considerable number of advantages in front of other navigation systems. It operates continuously, has low hardware faults, provides a relatively fast output (compared to GNSS receivers) of at least 50 Hz and exhibits

low short-term noise. It provides effective attitude, angular rate, and acceleration measurements as well as position and velocity. It is also invulnerable to jamming and interference, and is non-radiating (which is important for military stealth). On the other side, an inertial navigation solution requires initialization and its accuracy then degrades with time as the inertial instrument errors are integrated through the navigation equations. Furthermore, INS capable of providing effective sole-means navigation for more than a few minutes after initial alignment are expensive at around $100,000.

GNSS provides a high long-term position accuracy with errors limited to a few meters (stand-alone), while user equipment is available for less than $100. However, compared to INS, the output rate is low, typically around 10 Hz, the short-term noise of a code-based position solution is high, and standard GNSS user equipment does not measure attitude. GNSS signals are also subject to obstruction and interference, so GNSS cannot be relied upon to provide a continuous navigation solution.

The benefits and drawbacks of INS and GNSS are complementary, so by integrating them, the advantages of both technologies are combined to give a continuous, high-bandwidth, complete navigation solution with high long- and short-term accuracy. In an integrated INS/GNSS, or GNSS/INS, navigation system, GNSS measurements prevent the inertial solution drifting, while the INS smooths the GNSS solution and bridges signal outages.

INS/GNSS integration, sometimes known as hybridization, is suited to established inertial navigation applications such as ships, airliners, military aircraft, and long-range missiles. Integration with GNSS also makes inertial navigation practical with lower cost tactical-grade inertial sensors, making INS/GNSS a suitable navigation solution for light aircraft, helicopters, UAVs, short and medium range guided weapons, smaller boats, and potentially trains. INS/GNSS is sometimes used for road vehicles and personal navigation.

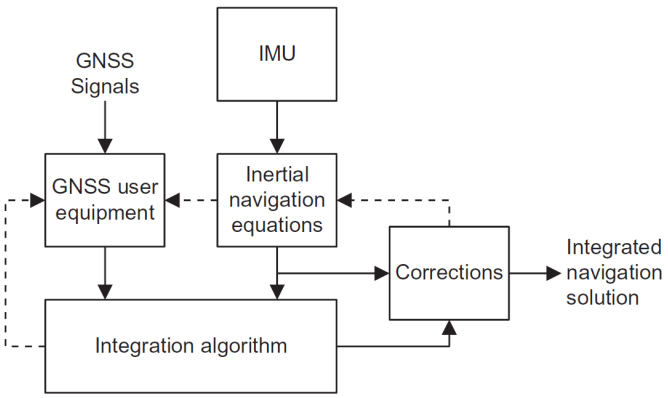

Figure 1.5: Standard INS/GNSS Architecture [3]

The basic configuration of a typical INS/GNSS navigation system is depicted in figure 1.5. The integration algorithm compares the inertial navigation solution with the outputs of GNSS user equipment and estimates corrections to the inertial position, velocity, and attitude solution, usually alongside other parameters by the implementation of a Kalman filter [105], which can have different architectures (the most well known are loosely coupled, tightly coupled and ultra-tightly coupled). Also, the difference between those architectures

depends on the type of data shared between INS and GNSS. Those the advantages and drawback of all of them [106]:

- Loosely coupled architecture is relatively easy to calculate, but its accuracy falls far behind with the other ones, specially when dealing with anti-interference abilities.
- Tightly coupled architecture is just a bit more complex, however has a much better anti-interference abilities. It also provides good navigation solution even if there are less than 4 satellites.
- Ultra-tightly coupled is one of the most complex architectures with a high calculation load.

The corrected inertial navigation solution then forms the integrated navigation solution. This architecture ensures that an integrated navigation solution is always produced, regardless of GNSS signal availability. The dotted lines in figure 1.5 show data flows present in some systems but not others.

The hardware configuration of INS/GNSS systems varies. The integration algorithm may be hosted in the INS, the GNSS user equipment, or separately. Alternatively, everything may be hosted in one unit, sometimes known as an embedded GNSS in INS (EGI) or integrated GNSS/INS (IGI). Where the inertial navigation equations and integration algorithm share the same processor, but the IMU is separate, the system is sometimes known as an integrated IMU/GNSS or GNSS/IMU. However, an IMU/GNSS is no different to an INS/GNSS.

1.3.2. INS/GNSS integration simulation

Figure 1.6 shows a simulation consisting on the movement of an aircraft which performs a turning manoeuvre in a total simulation time of 300 seconds. The title of each subplot is self-explanatory.

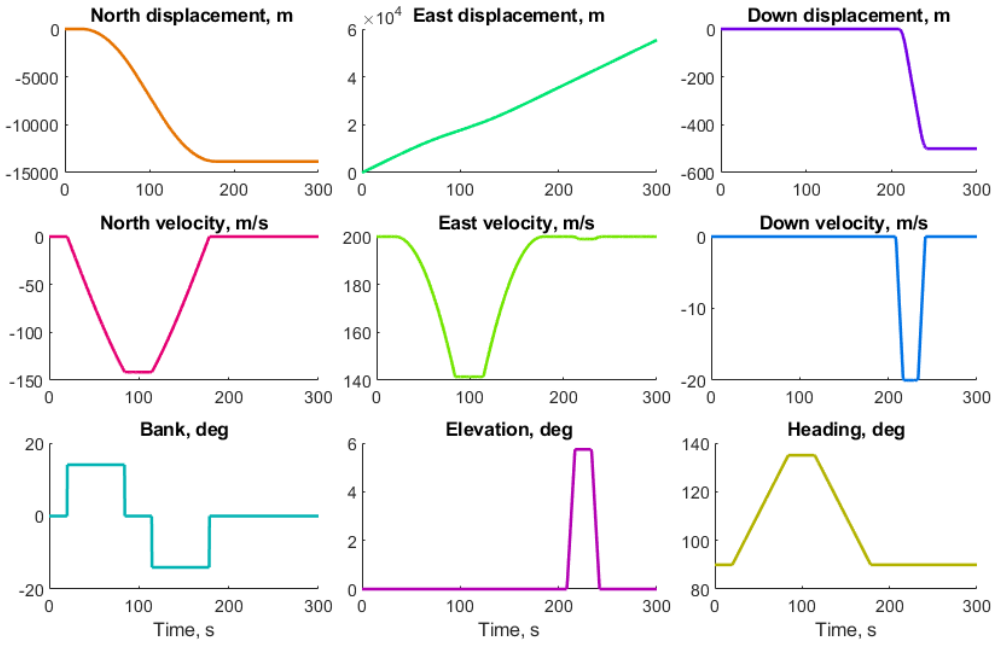

Figure 1.6: Plot representing movement of an aircraft

Figure 1.7 tries to give the performance of the GNSS/INS integration with a decrease of error over time coupled with some randomness coming from the estimations of the Kalman Filter.

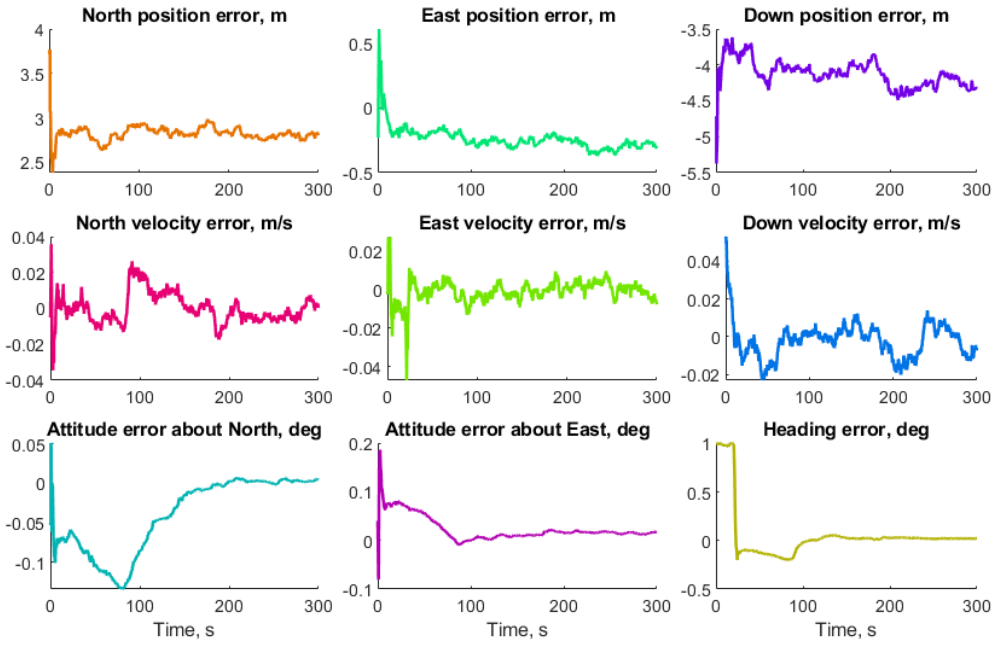

Figure 1.7: Errors of navigation for position, velocity and attitude corrected over time

The data configured for this GNSS/INS integration is extracted from [105]. According to the reference, this data is the most suitable for an aviation grade IMU and GPS receiver, and also for the Kalman Filter settings. Notice that we are using a tightly coupled integration, which for a UAV swarm with relatively cheap hardware its the most optimal architecture for integration. What this architecture does is to correct the inertial navigation system output with the pseudo-range and pseudo-range rate from GNSS satellite observations [105]).

- Integration type: Tightly coupled.
- Attitude initialization errors (deg): -[0.05, 0.04, 1]
- IMU accelerometers biases (μg): [900, -1300, 800]
- IMU gyroscope biases (deg/hour): [-9, -13, -8]
- IMU accelerometer scale factor and cross-coupling errors (ppm): [500, -300, 200; -150, 600, 250; -250, 100, 450] * $10e^{-6}$
- IMU gyroscope scale factor and cross-coupling errors (ppm): [400, -300, 250; 0, -300, -150; 0, 0, -350] * $10e^{-6}$
- IMU accelerometer noise root power spectral density (μg Hz): 100
- IMU gyroscope noise root power spectral density (deg/hour): 0.01
- IMU accelerometer quantization level (m/s^2): 0.01
- IMU gyroscope quantization level (rad/s): 0.0002
- GPS signal in space error (m): 1

- GPS zenith ionosphere error (m): 2
- GPS zenith troposphere error (m): 0.2
- GPS code tracking error (m): 1
- GPS range rate tracking error (m/s): 0.02
- GPS initial receiver clock offset (m): 10000
- GPS initial receiver clock drift (m/s): 100
- Kalman Filter initial attitude uncertainty per axis (deg): 1
- Kalman Filter initial velocity uncertainty per axis (m/s): 0.1
- Kalman Filter initial position uncertainty per axis (m): 10
- Kalman Filter initial accelerometer bias uncertainty per instrument (m/s^2): 9.80665
- Kalman Filter initial gyroscope bias uncertainty per instrument (deg/hour): 10
- Kalman Filter initial clock instrument uncertainty per axis (m): 10
- Kalman Filter initial clock drift uncertainty per axis (m/s): 0.1
- Kalman Filter IMU gyroscope noise power spectral density (deg^2/hour): 0.002
- Kalman Filter IMU accelerometer noise power spectral density (μg^2 Hz): 200
- Kalman Filter receiver receiver clock frequency drift power spectral density (m^2/s^3): 1
- Kalman Filter receiver receiver clock phase drift power spectral density (m^2/s): 1
- Kalman Filter pseudo-range measurement noise (m): 2.5
- Kalman Filter pseudo-range rate measurement noise (m/s): 0.1

CHAPTER 2. TRAJECTORY PLANNING

2.1. Fundamentals review

The essential objective of trajectory planning is to provide structured mobility, or in other words, to facilitate moving or flying multiple vehicles (in our case UAVs) from one location to another. There may be several locations to visit before reaching the final destination, and hence several consecutive paths may be required. Generally, there will be several predefined points of interest on a known or partially known map/area. The UAV will have a specific attitude, which is combined with its location to give the UAV pose $P(x,y,z,\theta,\phi)$, where (x,y,z) is the UAV location or waypoint and (θ,ϕ) are the horizontal and vertical angles, respectively. Consider a UAV moving from one pose, P_s, to another, P_f where P_s and P_f are labelled the start and finish poses, respectively. Path planning involves producing one or more flight paths $r(q)$ connecting P_s and P_f. Mathematically, this can be represented as [4]:

$$P_s \xrightarrow{r(q)} P_f \tag{2.1}$$

where $r(q)$ is the resulting path, and q is defined as a path parameter. This parameter can be a length variable $(0 \leq q \leq s)$ for a straight-line path or an angle variable $(0 \leq q \leq \theta)$ for a curved path. The choice of path variable depends on the path formulation. However, equation 2.1 only shows the most essential mathematical concept. For a single UAV flying from a location with start pose $P_s(x_s,y_s,z_s,\theta_s,\phi_s)$ to a location with finish pose $P_f(x_f,y_f,z_f,\theta_f,\phi_f)$, we can write the following equation:

$$P_s(x_s,y_s,z_s,\theta_s,\phi_s) \xrightarrow{r(q)} P_f(x_f,y_f,z_f,\theta_f,\phi_f) \tag{2.2}$$

And finally, in the case of swarms where multiple UAVs are present, we write the following equation, where i represents the i-th UAV:

$$P_{s,i}(x_{s,i},y_{s,i},z_{s,i},\theta_{s,i},\phi_{s,i}) \xrightarrow{r_i(q)} P_{f,i}(x_{f,i},y_{f,i},z_{f,i},\theta_{f,i},\phi_{f,i}) \tag{2.3}$$

At this point we have only considered the mathematical representation of multiple paths for multiple UAVs which can move freely in a space without any imposed restrictions on them. We can distinguish the following constraints:

- Flyable flight trajectories that meet the kinematic and motion constraints of a single UAV.
- Safe flight trajectory where the UAV can avoid fixed and moving obstacles.
- System requirements like maintaining communications range and trajectory length and time minimization.

$$P_{s,i}(x_{s,i},y_{s,i},z_{s,i},\theta_{s,i},\phi_{s,i}) \xrightarrow{\sqcup r_i(q)} P_{f,i}(x_{f,i},y_{f,i},z_{f,i},\theta_{f,i},\phi_{f,i}) \tag{2.4}$$

Where the new parameter $\sqcup$ includes all the constraints defined previously.

2.2. Methods for trajectory generation

2.2.1. Dubins Path

The most simple path we can imagine consists of a combination of linear segments and constant-curvature segments. Combining both of these rectilinear and circular arcs will produce the shortest-length manoeuvre for a vehicle between two coordinates in space. Appart from the previously defined coordinates, the orientation of the vehicle is also important (heading and elevation in three dimensional case).

The shortest trajectory connecting two positions is the well known Dubins path developed in 1957 [107]. A Dubins path can be defined as the shortest possible path that meets the maximum-curvature bound between two points with specific orientations in a plane is either a CLC or a CCC path, or a subset of them, where C represents circular arc and L represents straight-line tangent to C. The CLC path is formed by connecting two circular arcs by a line that is tangential to them, and the CCC path is formed by three consecutive circular arcs that are tangential to each other. The subsets of these paths are CL, LC and CC. Figure 2.1 shows the CLC and CCC paths.

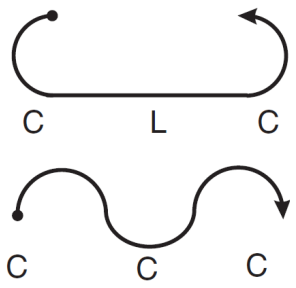

Figure 2.1: CLC and CCC types of Dubins path [4]

Figure 2.1 only tries to give an schematic representation on the available Dubins paths. When performing the mathematical computations, the resulting curves resemble more to the ones depicted in figure 2.2.

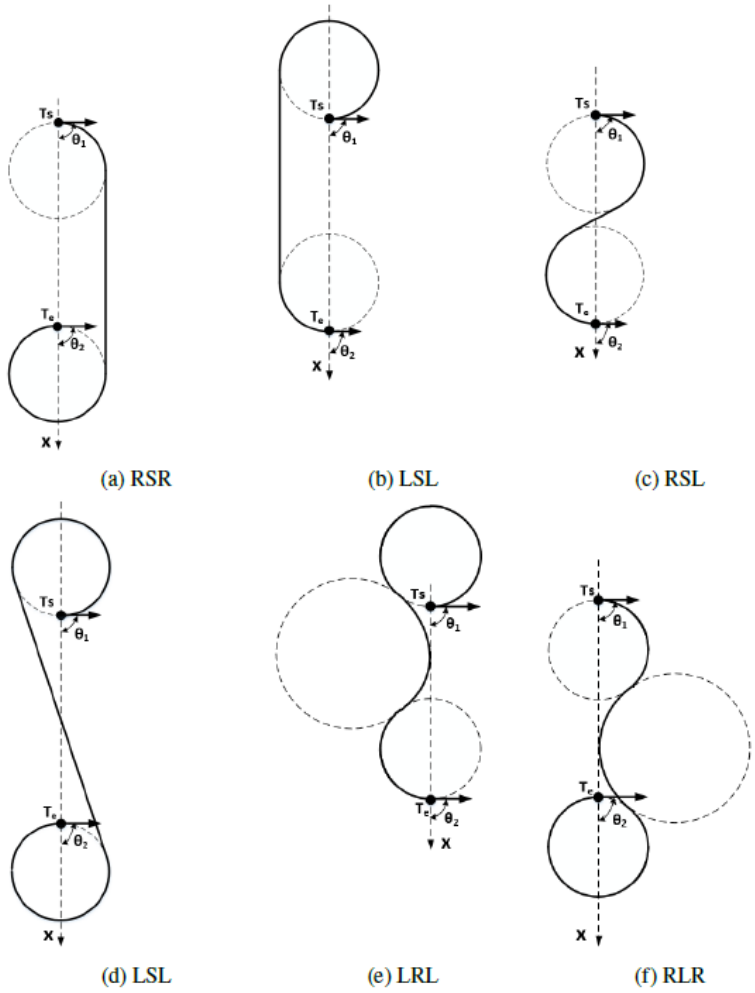

Figure 2.2: Detailed Dubins paths [5]

2.2.2. 3D trajectory extension

Two dimensional path planning is a well studied topic and can be found on multiple research papers and books, since it is applied in lots of fields concerning robotics. However, when dealing with aerospace vehicles, three dimensions are the most common thing and several methods have been proposed. Some of those methods are listed below:

- Dubins path extension using differential geometry [4]
- Pythagorean Hodographs Paths in 3D [4]
- Dubins path with Bezier interpolation [6]

The Dubins path with Bezier interpolation for generating a 3D trajectory seems to be the most promising method in terms of computational efficiency and ease of use since it is calculated as a linear algebra problem [6].

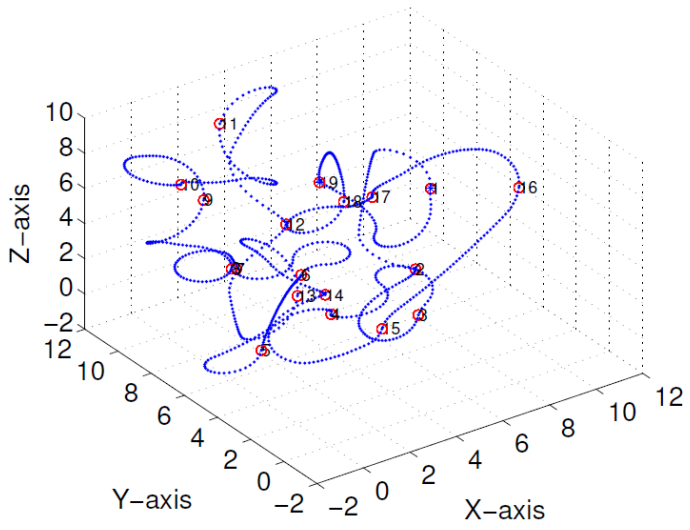

Figure 2.3: Full 3D trajectory made with the Bezier interpolation [6]

CHAPTER 3. TASK ASSIGNMENT

3.1. Introduction

This chapter tries to be an extension of a typical search and prosecute mission, in which multiple UAVs cooperate with each other as a team to detect and prosecute targets. The advantage of deploying multiple UAVs is translated in mission robustness and reduced completion time. UAVs may have to use various types and quantities of resources to completely prosecute a target. A single UAV may not have sufficient resources, hence a sub-team of UAVs whose total resources are sufficient to completely prosecute the target needs to be assigned. This sub-team of UAVs is called a coalition and each agent in the coalition is called a coalition member. The agent that detected the target is called the coalition leader. Other objectives for the mission are:

1. Prosecute the target in minimum time so the UAVs can accomplish the mission as fast as possible.
2. Minimize the coalition size due to the fact that UAVs have limited sensor range, therefore they need to distribute the search effort to detect the target.
3. Simultaneously prosecute the target to induce maximum damage to the target (in case of a military target).

3.2. Research on task assignment

There has been lots of studies regarding to multi-robot task allocation, starting with [75]. In the case of task allocation with UAVs (which are flying vehicles) we must consider the following:

- UAVs travel with greater velocities than ground robots.
- Resources of each robot do not deplete with use.

Lots of researchers have developed task allocation algorithms for efficiently allocating UAVs to different tasks assuming the following pre-requisites [108]:

- UAVs are homogeneous
- UAVs can carry resources not depleting after use
- UAVs can prosecute the target with or without resources

As we have said previously, task allocation must be done amongst the members of a coalition. A coalition can be defined as a group of team members that have agreed to cooperate with each other in a temporary manner to execute a single task. Once the task is accomplished, the coalition members can perform other tasks. Determining the optimal coalition from a group of agents is a computationally intensive task, which is considered to be an NP-hard problem [109]. To overcome this, some algorithms providing a near-optimal solutions have been developed during the years.

Forming a coalition to achieve tasks is an active field of research on both the multi-agent and multi-robot communities [110] [111]. The coalition formation algorithms developed in the multi-agent community cannot be directly applied to multiple robot systems since the resources cannot be transferred from one robot to another.

The algorithms developed on the previous studies require significant amount of computation time and communication which may not be possible in UAV networks with current technology. This is due to the fact that UAVs move at considerable high speeds without the possibility to stop while flying. Hence, these algorithms cannot be directly applied to the UAV scenario. Other studies present a coalition formation scheme where a coalition leader robot broadcasts the existence of a task and other robots reply by providing their availability [112]. The leader robot evaluates all possible coalitions and sends an accept decision to the robots that it considers suitable. At the end, the final goal is to minimize the length of task tours of the UAVs with known target positions. The resources of the tracking agents have the same capability and the problem of depleting resources is not addressed. Studies on the multiple agent rendezvous problem are explained in [113]. Those studies address the issue of multiple agents arriving simultaneously to a common location by cooperating with each other.

Typical missions of the military field like search and destroy mission, have also been studied in [114], where the target location, coalition leader and the number of UAVs required to form a coalition are predefined before executing the task. The coalition leader searches for other UAVs to form a coalition and simultaneously attack the target by generating commands given by a time-optimal control problem.

In more contemporary studies like [115], the coalition leader determines the agent with the latest arrival time at the target and establishes this time as the arrival time of the coalition. Then, coalition members replan their paths to meet this constraint and also agents do not communicate with each other and have only one communication with the leader, thus reducing communication costs. Moreover, it is assumed that agents have non-uniform payload resources and limited sensor range for target search. Once an agent detects a target, it broadcasts the information about the kind of resources required to prosecute the target. Each of the UAVs that have at least one of the required resources send their information to the detecting agent. The detecting agent determines a coalition of agents to prosecute the target. To determine the coalition, a polynomial time sub-optimal and optimal coalition formation algorithms are developed. Once the coalition is formed, the agents coordinate with each other by modifying their paths such that they attack the target simultaneously. In order to determine the deviation of the mission from the global optimal mission where the target locations with their resource requirements, and the UAV initial positions with their resources are known a priori, a global optimization problem is solved. Finding the globally optimal solution for this kind of problem is computationally intensive and difficult. Linear Programming based techniques may not be the most effective technique taking all the constraints of the problem, so Particle Swarm Optimization technique (PSO) turns out to be the best option producing a global optimal solution which can meet the constraints.

Part V

SIMULATION TOOLS

CHAPTER 1. OPEN SOURCE TOOLS

1.1. Introduction

This chapter will review some of the tools required to design and analyze the swarm functionality, according to all the things we have explained in the previous parts. Recall that in this chapter only a brief description of the tools is given, and for more information the reader should study those tools by themselves.

Apart from the software libraries that are mentioned in this chapter, some algorithms like the ones explained in task allocation or trajectory should be implemented from scratch using C++ as preferable language, for better compatibility with other libraries.

1.2. Eigen

Eigen is a high-level C++ library of template headers for linear algebra, matrix and vector operations, geometrical transformations, numerical solvers and related algorithms. This will be used when dealing with the heavy load computations of matrices applied in the calculation of 3D trajectory paths and Kalman filters [116].

1.3. GPSTK (GPS Toolkit)

Developing a fully fledged GPS (or GNSS) simulator from scratch can be an enormous task and a bit useless if we have available open source tools that allow us to carry out this kind of task. According to its creators, GPSTk core library is fully developed in C++ language, and it provides the most robust, broadly useful, and platform independent code in the GPSTk. It provides a number of models and algorithms found in GPS textbook and classic papers, such as solving for the user position or estimating atmospheric refraction. Common formats are supported as well, such as RINEX or SP3. There are several categories of function that provide the base functionality for the GPSTk applications and for a number of other independent projects [117].

This library is the most suitable election to simulate GPS data and couple it to a simulated INS to implement a Kalman filter and implement a navigation solution.

1.4. OMNet++

OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. The word network involves wired and wireless communication networks, on-chip networks, queueing networks, and so on. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, photonic networks, etc., is provided by model

frameworks, developed as independent projects. OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment, and a host of other tools [118].

This software library will be more difficult to use since it comes with its own user interface and allows to generate any type of network available in any kind of environment. The C++ code written to implement the details of the protocol of swarm communications should be merged within this application to properly simulate the network with the custom protocol and also account adverse effects like losses of connections, etc.

1.5. ParadisEO

ParadisEO is a C++ framework dedicated to the reusable design of metaheuristics, hybrid metaheuristics, and parallel and distributed metaheuristics. ParadisEO provides a broad range of features including evolutionary algorithms, local searches, Particle swarm optimization, the most common parallel and distributed models and hybridization mechanisms, amongst others. This framework must be used when dealing with the task allocation problem, where those kind of metaheuristic algorithms are applied [119].

Part VI

CONCLUSIONS & FUTURE WORK

CONCLUSIONS

In this thesis we have reviewed the essentials on all the topics and concepts involved in the design of an UAV swarm. At the end of this thesis, the reader would have acquired a set of concepts to be used as a baseline to develop his own UAV swarm simulation tool, and also to perform more in-deep study on the algorithms available on this topic.

Recall that swarming technologies not ony apply to UAV but also on ground robots. The last has lots of studies that have been used as a reference for the UAV swarms and they must also be considering when studying in detail this kind of technology.

Appart from this, when talking about UAV swarms we must not forget the UAV itself, which is the remaining half part of the complete system. However, the author did not made any special mention of the aircraft due to the fact that swarming technology is more related to algorithmic topics and not aircraft design.

Regarding to the algorithms and protocols covered in this thesis, the author encourages the reader to get better documentation on those by looking at the references. Those algorithms covers lots of equations with detailed explanation filling several pages. The mathematical formulae covered in this thesis are used only for explanatory purposes.

FUTURE WORK

This master thesis has only shown a few insights on UAV swarms. The author is planning to develop his own simulation tool in the near future, using all the open source technologies provided in this thesis.

The author of this simulation tool plans to make it publicly available, so the open source community can contribute to it. This can potentially increase the interest of the open source community on other fields related to UAV technologies. In the long term, private companies interested in UAV technologies can decide to provide additional funding to any related projects, having a huge impact on the global economy and research institutions.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Heiko Hamann. Swarm robotics: A formal approach. 2018. ix, 5, 6, 7

[2] Kimon P Valavanis and George J Vachtsevanos. *Handbook of Unmanned Aerial Vehicles-5 Volume Set*. New York, NY, USA: Springer, 2014. ix, 13, 18, 19, 20, 22, 23, 29

[3] Paul D Groves. Principles of gnss, inertial, and multisensor integrated navigation systems, [book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2), 2015. ix, 43, 44, 45, 46, 47, 48

[4] Antonios Tsourdos, Brian White, and Madhavan Shanmugavel. *Cooperative path planning of unmanned aerial vehicles*. John Wiley & Sons, 2010. ix, 53, 54, 55

[5] Yiheng Wang. *3D Dubins curves for multi-vehicle path planning*. Missouri University of Science and Technology, 2018. ix, 55

[6] Wenyu Cai and Meiyan Zhang. Smooth 3d dubins curves based mobile data gathering in sparse underwater sensor networks. *Sensors*, 18(7):2105, 2018. ix, 55, 56

[7] Thomas Schmickl and Heiko Hamann. Beeclust: A swarm algorithm derived from honeybees. *Bio-inspired Computing and Communication Networks. CRC Press (March 2011)*, 2011. 6

[8] David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee. Pheromone robotics. *Autonomous Robots*, 11(3):319–324, 2001. 7

[9] Heiko Hamann, Marc Szymanski, and Heinz Wörn. Orientation in a trail network by exploiting its geometry for swarm robotics. In *SIS*, pages 310–315, 2007. 7

[10] Matt Wilson, Chris Melhuish, Ana B Sendova-Franks, and Samuel Scholes. Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies. *Autonomous Robots*, 17(2):115–136, 2004. 8

[11] Chris Melhuish, Matt Wilson, and Ana Sendova-Franks. Patch sorting: Multi-object clustering using minimalist robots. In *European Conference on Artificial Life*, pages 543–552. Springer, 2001. 8

[12] Hongli Ding and Heiko Hamann. Sorting in swarm robots using communication-based cluster size estimation. In *International Conference on Swarm Intelligence*, pages 262–269. Springer, 2014. 8

[13] Marco Dorigo, Elio Tuci, Vito Trianni, Roderich Groß, Shervin Nouyan, Christos Ampatzis, Thomas Halva Labella, Rehan O'Grady, Michael Bonani, Francesco Mondada, et al. Swarm-bot: Design and implementation of colonies of self-assembling robots. *Computational intelligence: principles and practice*, pages 103–135, 2006. 8

[14] Rehan O'Grady, Roderich Groß, Francesco Mondada, Michael Bonani, and Marco Dorigo. Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain. In *European Conference on Artificial Life*, pages 272–281. Springer, 2005. 8

[15] Michael Rubenstein and Wei-Min Shen. Scalable self-assembly and self-repair in a collective of robots. In *2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 1484–1489. IEEE, 2009. 8

[16] Guy Theraulaz and Eric Bonabeau. Modelling the collective building of complex architectures in social insects with lattice swarms. *Journal of theoretical biology*, 177(4):381–400, 1995. 9

[17] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014. 9

[18] Frederico Augugliaro, Sergei Lupashin, Michael Hamer, Cason Male, Markus Hehn, Mark W Mueller, Jan Sebastian Willmann, Fabio Gramazio, Matthias Kohler, and Raffaello D'Andrea. The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine*, 34(4):46–64, 2014. 9

[19] Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay Kumar. Cooperative grasping and transport using multiple quadrotors. In *Distributed autonomous robotic systems*, pages 545–558. Springer, 2013. 9

[20] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987. 10

[21] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6):1226, 1995. 10

[22] Tamás Vicsek and Anna Zafeiris. Collective motion. *Physics reports*, 517(3-4):71–140, 2012. 10

[23] Christian A Yates, Radek Erban, Carlos Escudero, Iain D Couzin, Jerome Buhl, Ioannis G Kevrekidis, Philip K Maini, and David JT Sumpter. Inherent noise can facilitate coherence in collective swarm motion. *Proceedings of the National Academy of Sciences*, 106(14):5464–5469, 2009. 10

[24] Ali E Turgut, Hande Çelikkanat, Fatih Gökçe, and Erol Şahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2):97–120, 2008. 10

[25] Christoph Moeslinger, Thomas Schmickl, and Karl Crailsheim. A minimalist flocking algorithm for swarm robots. In *European Conference on Artificial Life*, pages 375–382. Springer, 2009. 10

[26] Thomas Schmickl and Karl Crailsheim. Costs of environmental fluctuations and benefits of dynamic decentralized foraging decisions in honey bees. *Adaptive Behavior*, 12(3-4):263–277, 2004. 11

[27] Bernd Meyer, Madeleine Beekman, and Audrey Dussutour. Noise-induced adaptive decision-making in ant-foraging. In *International Conference on Simulation of Adaptive Behavior*, pages 415–425. Springer, 2008. 11

[28] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous robots*, 13(2):127–141, 2002. 11

[29] Ralf Mayet, Jonathan Roberz, Thomas Schmickl, and Karl Crailsheim. Antbots: A feasible visual emulation of pheromone trails for swarm robots. In *International conference on swarm intelligence*, pages 84–94. Springer, 2010. 11

[30] Shervin Nouyan, Roderich Groß, Michael Bonani, Francesco Mondada, and Marco Dorigo. Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, 2009. 11

[31] Deborah M Gordon. The organization of work in social insect colonies. *Nature*, 380(6570):121–124, 1996. 11

[32] Istvan Karsai and Thomas Schmickl. Regulation of task partitioning by a "common stomach": a model of nest construction in social wasps. *Behavioral Ecology*, 22(4):819–830, 2011. 11

[33] Payam Zahadat, Sibylle Hahshold, Ronald Thenius, Karl Crailsheim, and Thomas Schmickl. From honeybees to robots and back: division of labour based on partitioning social inhibition. *Bioinspiration & biomimetics*, 10(6):066005, 2015. 11

[34] Payam Zahadat and Thomas Schmickl. Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior*, 24(2):87–101, 2016. 12

[35] Thomas Lemaire, Rachid Alami, and Simon Lacroix. A distributed tasks allocation scheme in multi-uav context. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3622–3627. IEEE, 2004. 12

[36] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous agents and multi-agent systems*, 28(1):101–125, 2014. 12

[37] Giovanni Pini, Arne Brutschy, Gianpiero Francesca, Marco Dorigo, and Mauro Birattari. Multi-armed bandit formulation of the task partitioning problem in swarm robotics. In *International Conference on Swarm Intelligence*, pages 109–120. Springer, 2012. 12

[38] Yara Khaluf, Mauro Birattari, and Heiko Hamann. A swarm robotics approach to task allocation under soft deadlines and negligible switching costs. In *International Conference on Simulation of Adaptive Behavior*, pages 270–279. Springer, 2014. 12

[39] Eliseo Ferrante, Ali Emre Turgut, Edgar Duéñez-Guzmán, Marco Dorigo, and Tom Wenseleers. Evolution of self-organized task specialization in robot swarms. *PLoS computational biology*, 11(8):e1004273, 2015. 12

[40] Alan C Schultz, John J Grefenstette, and William Adams. Robo-shepherd: Learning complex robotic behaviorsy. 12

[41] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Cooperative coevolution of partially heterogeneous multiagent systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 297–305, 2015. 13

[42] Ronald C Arkin and Magnus Egerstedt. Temporal heterogeneity and the value of slowness in robotic systems. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1000–1005. IEEE, 2015. 13

[43] Daniela Kengyel, Heiko Hamann, Payam Zahadat, Gerald Radspieler, Franz Wotawa, and Thomas Schmickl. Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms. In *International conference on principles and practice of multi-agent systems*, pages 201–217. Springer, 2015. 13

[44] Amanda Prorok, M Ani Hsieh, and Vijay Kumar. Formalizing the impact of diversity on performance in a heterogeneous swarm of robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5364–5371. IEEE, 2016. 13

[45] Jean-Claude Latombe. Robot motion planning, 1991. *Cité page*, 18, 1990. 17

[46] Y Uny Cao, Andrew B Kahng, and Alex S Fukunaga. Cooperative mobile robotics: Antecedents and directions. In *Robot colonies*, pages 7–27. Springer, 1997. 17

[47] Kazuhiro Kosuge and Manabu Sato. Transportation of a single object by multiple decentralized-controlled nonholonomic mobile robots. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 3, pages 1681–1686. IEEE, 1999. 19

[48] Thomas G Sugar and Vijay Kumar. Control of cooperating mobile manipulators. *IEEE Transactions on robotics and automation*, 18(1):94–103, 2002. 19

[49] Manoj Mittal, JVR Prasad, and Daniel P Schrage. Nonlinear adaptive control of a twin lift helicopter system. In *1990 American Control Conference*, pages 743–748. IEEE, 1990. 20

[50] H Keath Reynolds and Armando A Rodriguez. H/sup infinity/control of a twin lift helicopter system. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pages 2442–2447. IEEE, 1992. 20

[51] C-I Lim, Richard P Metzger, and Armando A Rodriguez. An interactive modeling, simulation, animation and real-time control (mosart) twin lift helicopter system environment. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 4, pages 2747–2751. IEEE, 1999. 20

[52] Nathan Michael, Jonathan Fink, and Vijay Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, 2011. 20

[53] Koushil Sreenath, Nathan Michael, and Vijay Kumar. Trajectory generation and control of a quadrotor with a cable-suspended load-a differentially-flat hybrid system. In *2013 IEEE International Conference on Robotics and Automation*, pages 4888–4895. IEEE, 2013. 20

[54] F Giulietti, L Pollini, and M Innocenti. Autonomous formation flight ieee control systems magazine, 2000. 21

[55] Corey Schumacher and Sahjendra Singh. Nonlinear control of multiple uavs in close-coupled formation flight. In *AIAA Guidance, navigation, and control conference and exhibit*, page 4373, 2000. 21

[56] Jonathan How, Yoshiaki Kuwata, and Ellis King. Flight demonstrations of cooperative control for uav teams. In *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, page 6490, 2004. 21

[57] Ben Yun, Ben M Chen, Kai Yew Lum, and Tong H Lee. Design and implementation of a leader-follower cooperative control system for unmanned helicopters. *Journal of Control Theory and Applications*, 8(1):61–68, 2010. 21

[58] Yu Gu, Brad Seanor, Giampiero Campa, Marcello R Napolitano, Larry Rowe, Srikanth Gururajan, and Sheng Wan. Design and flight testing evaluation of formation control laws. *IEEE Transactions on Control Systems Technology*, 14(6):1105–1112, 2006. 21

[59] Damien Galzi and Yuri Shtessel. Uav formations control using high order sliding modes. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006. 21

[60] Selcuk Bayraktar, Georgios E Fainekos, and George J Pappas. Experimental cooperative control of fixed-wing unmanned aerial vehicles. In *2004 43rd IEEE conference on decision and control (CDC)(IEEE Cat. No. 04CH37601)*, volume 4, pages 4292–4298. IEEE, 2004. 21

[61] Tobias Paul, Thomas R Krogstad, and Jan Tommy Gravdahl. Modelling of uav formation flight using 3d potential field. *Simulation Modelling Practice and Theory*, 16(9):1453–1462, 2008. 21

[62] Giampiero Campa, Yu Gu, Brad Seanor, Marcello R Napolitano, Lorenzo Pollini, and Mario L Fravolini. Design and flight-testing of non-linear formation control laws. *Control Engineering Practice*, 15(9):1077–1092, 2007. 21

[63] Jaydev P Desai, James P Ostrowski, and Vijay Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE transactions on Robotics and Automation*, 17(6):905–908, 2001. 22

[64] Shannon Zelinski, T John Koo, and Shankar Sastry. Hybrid system design for formations of autonomous vehicles. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 1, pages 1–6. IEEE, 2003. 22

[65] Amanda JC Sharkey. Robots, insects and swarm intelligence. *Artificial Intelligence Review*, 26(4):255–268, 2006. 22

[66] Dandan Zhang, Guangming Xie, Junzhi Yu, and Long Wang. Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, 55(7):572–588, 2007. 22

[67] C Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218, 1993. 22

[68] Maja J Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441, 1993. 23

[69] Michael A Kovacina, Daniel W Palmer, Guang Yang, Ravi Vaidyanathan, et al. Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles. In *IROS*, pages 2782–2788, 2002. 23

[70] K Han, J Lee, and Youdan Kim. Unmanned aerial vehicle swarm control using potential functions and sliding mode control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 222(6):721–730, 2008. 23

[71] Prithviraj Dasgupta. A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(3):549–563, 2008. 23

[72] Yaniv Altshuler, Vladimir Yanovsky, Israel A Wagner, and Alfred M Bruckstein. Efficient cooperative search of smart targets using uav swarms1. *Robotica*, 26(4):551–557, 2008. 23

[73] Lynne E Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE transactions on robotics and automation*, 14(2):220–240, 1998. 23

[74] Lynne E Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. In *AAAI fall symposium: regarding the intelligence in distributed intelligent systems*, pages 1–6, 2007. 24

[75] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research*, 23(9):939–954, 2004. 24, 57

[76] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012. 24

[77] Aníbal Ollero and Iván Maza. *Multiple heterogeneous unmanned aerial vehicles*, volume 37. Springer, 2007. 24

[78] Ivan Maza, Fernando Caballero, Jesus Capitan, José Ramiro Martinez-de Dios, and Anibal Ollero. A distributed architecture for a robotic platform with aerial sensor transportation and self-deployment capabilities. *Journal of Field Robotics*, 28(3):303–328, 2011. 24

[79] Sebastian Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *The International Journal of Robotics Research*, 20(5):335–363, 2001. 25

[80] Luis Merino, Fernando Caballero, J Ramiro Martínez-de Dios, Joaquin Ferruz, and Aníbal Ollero. A cooperative perception system for multiple uavs: Application to automatic detection of forest fires. *Journal of Field Robotics*, 23(3-4):165–184, 2006. 25

[81] Kurt Konolige, Dieter Fox, Benson Limketkai, Jonathan Ko, and Benjamin Stewart. Map merging for distributed robot navigation. In *Proceedings 2003 IEEE/RSJ international conference on intelligent robots and systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 212–217. IEEE, 2003. 25

[82] Salah Sukkarieh, Eric Nettleton, Jong-Hyuk Kim, Matthew Ridley, Ali Goktogan, and Hugh Durrant-Whyte. The anser project: Data fusion across multiple uninhabited air vehicles. *The International Journal of Robotics Research*, 22(7-8):505–539, 2003. 25

[83] Subir Kumar Sarkar, Tiptur Gangaraju Basavaraju, and C Puttamadappa. *Ad hoc mobile wireless networks: principles, protocols and applications*. Auerbach publications, 2007. 31, 32

[84] Muhammad Asghar Khan, Alamgir Safi, Ijaz Mansoor Qureshi, and Inam Ullah Khan. Flying ad-hoc networks (fanets): A review of communication architectures, and routing protocols. In *2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*, pages 1–9. IEEE, 2017. 33

[85] Chen-Mou Cheng, Pai-Hsiang Hsiao, HT Kung, and Dario Vlah. Maximizing throughput of uav-relaying networks with the load-carry-and-deliver paradigm. In *2007 IEEE Wireless Communications and Networking Conference*, pages 4417–4424. IEEE, 2007. 33

[86] Thomas Jonson, Jonah Pezeshki, Victor Chao, Kristofer Smith, and James Fazio. Application of delay tolerant networking (dtn) in airborne networks. In *MILCOM 2008-2008 IEEE Military Communications Conference*, pages 1–7. IEEE, 2008. 33

[87] Gary B Lamont, James N Slear, and Kenneth Melendez. Uav swarm mission planning and routing using multi-objective evolutionary algorithms. In *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pages 10–20. IEEE, 2007. 34

[88] Charalampos Konstantopoulos, Damianos Gavalas, and Grammati Pantziou. A mobility aware technique for clustering on mobile ad-hoc networks. In *International Conference on Distributed Computing and Networking*, pages 397–408. Springer, 2006. 34

[89] Kesheng Liu, Jun Zhang, and Tao Zhang. The clustering algorithm of uav networking in near-space. In *2008 8th International symposium on antennas, propagation and EM theory*, pages 1550–1553. IEEE, 2008. 34, 40

[90] Jeff Ko, Amit Mahajan, and Raja Sengupta. A network-centric uav organization for search and pursuit operations. In *Proceedings, IEEE Aerospace Conference*, volume 6, pages 6–6. IEEE, 2002. 34

[91] Charles E Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM computer communication review*, 24(4):234–244, 1994. 35

[92] T Clausen PJacquet and L Viennot. Optimized link state routing [olsr] protocol for adhoc network. In *proceedings of IEEE Multi Topic Conference, INMIC*, 2001. 35

[93] Abdel Ilah Alshabtat, Liang Dong, J Li, and F Yang. Low latency routing algorithm for unmanned aerial vehicles ad-hoc networks. *International Journal of Electrical and Computer Engineering*, 6(1):48–54, 2010. 36

[94] Shabana Habib, Somaila Saleem, and Khawaja Muhammad Saqib. Review on manet routing protocols and challenges. In *2013 IEEE Student Conference on Research and Developement*, pages 529–533. IEEE, 2013. 36

[95] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996. 36

[96] M Shobana and S Karthik. A performance analysis and comparison of various routing protocols in manet. In *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pages 391–393. IEEE, 2013. 37

[97] Charles E Perkins and Elizabeth M Royer. Ad-hoc on-demand distance vector routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100. IEEE, 1999. 37

[98] J Hope Forsmann, Robert E Hiromoto, and John Svoboda. A time-slotted on-demand routing protocol for mobile ad hoc unmanned vehicle systems. In *Unmanned Systems Technology IX*, volume 6561, page 65611P. International Society for Optics and Photonics, 2007. 37

[99] ZJ Haas and MR Pearlman. Zone routing protocol (zrp) a hybrid framework for routing in ad hoc networks, 2nd edn. ad hoc networking, vol. 1, chapter 7, 2001. 38

[100] D Helen and D Arivazhagan. Reducing overhead in ad-hoc network via temporarily ordered routing algorithm. *Wireless Communication*, 6(4):140–143, 2014. 38

[101] Ram Shringar Raw, Daya Krishan Lobiyal, and Sanjoy Das. An analytical approach to position-based routing protocol for vehicular ad hoc networks. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 147–156. Springer, 2012. 38

[102] Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000. 39

[103] Lin Lin, Qibo Sun, Jinglin Li, and Fangchun Yang. A novel geographic position mobility oriented routing strategy for uavs. *Journal of Computational Information Systems*, 8(2):709–716, 2012. 39

[104] Chunhua Zang and Shouhong Zang. Mobility prediction clustering algorithm for uav networking. In *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pages 1158–1161. IEEE, 2011. 39

[105] Bruce P Gibbs. *Advanced Kalman filtering, least-squares and modeling: a practical handbook*. John Wiley & Sons, 2011. 48, 51

[106] Kai Chen, Sensen Pei, Fuqiang Shen, and Shangbo Liu. Tightly coupled integrated navigation algorithm for hypersonic boost-glide vehicles in the Icef frame. *Aerospace*, 8(5):124, 2021. 49

[107] Lester 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*, 79(3):497–516, 1957. 54

[108] PB Sujit, A Sinha, and D Ghose. Multi-uav task allocation using team theory. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 1497–1502. IEEE, 2005. 57

[109] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial intelligence*, 111(1-2):209–238, 1999. 57

[110] Onn M Shehory, Katia Sycara, and Somesh Jha. Multi-agent coordination through coalition formation. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 143–154. Springer, 1997. 58

[111] Lovekesh Vig and Julie A Adams. Market-based multi-robot coalition formation. In *Distributed Autonomous Robotic Systems 7*, pages 227–236. Springer, 2006. 58

[112] Lynne E Parker and Fang Tang. Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, 94(7):1289–1305, 2006. 58

[113] Jie Lin, A Stephen Morse, and Brian DO Anderson. The multi-agent rendezvous problem. In *42nd ieee international conference on decision and control (ieee cat. no. 03ch37475)*, volume 2, pages 1508–1513. IEEE, 2003. 58

[114] Tomonari Furukawa, Frédéric Bourgault, Hugh F Durrant-Whyte, and Gamini Dissanayake. Dynamic allocation and control of coordinated uavs to engage multiple targets in a time-optimal manner. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2353–2358. IEEE, 2004. 58

[115] Joel G Manathara, PB Sujit, and Randal W Beard. Multiple uav coalitions for a search and prosecute mission. *Journal of Intelligent & Robotic Systems*, 62(1):125–158, 2011. 58

[116] Eigen official website. https://eigen.tuxfamily.org/index.php?title=Main_Page. 61

[117] Gpstk official repository. https://gitlab.com/sgl-ut/GPSTk. 61

[118] Omnet++ offcial website. https://omnetpp.org/. 62

[119] Paradiseo offcial repository. https://gitlab.inria.fr/paradiseo/paradiseo. 62