UNIVERSITÀ DEGLI STUDI FIRENZE

UNIVERSITÀ DEGLI STUDI DI PERUGIA

INdAM Istituto Nazionale di Alta Matematica

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA

CURRICULUM IN INFORMATICA
CICLO XXXIV

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Matteo Focardi

Drones
Applications, Safety and Security Issues

Settore Scientifico Disciplinare INF/01

Dottorando:
Giulio Rigoni

Tutore:
Prof. Cristina M. Pinotti

Coordinatore:
Prof. Matteo Focardi

Anni 2018/2021

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA

CURRICULUM IN INFORMATICA
CICLO XXXIV

Sede amministrativa Università degli Studi di Firenze
Coordinatore Prof. Matteo Focardi

Drones
Applications, Safety and Security Issues

Settore Scientifico Disciplinare INF/01

Dottorando:
Giulio Rigoni

Tutore:
Prof. Cristina M. Pinotti

Coordinatore:
Prof. Matteo Focardi

Anni 2018/2021

Contents

List of Figures

List of Tables

Abstract

The rise in popularity of drones or Unmanned Aerial Vehicles (UAVs) has made them a very appealing subject for investigation and study in different topics like localization of sensors and people, agriculture, delivery of packages, and last but not least, security and safety. In all those scenarios, the introduction of UAVs simplified the task for everyday life, but at the same time UAVs are subjected to attacks carried on by malicious users that can turn them into weapons. The initial part of this thesis tackles the delivery scenario using UAVs, from different perspectives: (i)first, a UAV delivery system is compared with a standard truck system, showing pros and cons of UAVs also considering the wind. Results show that the introduction of UAVs greatly benefits the delivery system, due to the ability of UAVs to fly over obstacles, traffic, and in general ignoring the landscape. Therefore, UAVs outperform trucks when deployed both in urban and suburban environments. (ii) Then, we consider a UAV delivery system where, to serve a set of deliveries, the UAVs select the best starting point based on the distribution of customers, aiming at minimizing the total distance traveled. We modeled the delivery area as a mix of country-side (i.e., Euclidean) and urban (i.e., Manhattan) areas. We tested different algorithms and our results show that the starting point, given the distribution of customers, indeed matters. To further prove our findings, an open air simulator was introduced, and it consolidated the results. Lastly, (iii) we tried to improve the UAV energy efficiency, in a windy delivery scenario, by calculating the best takeoff and landing position for the UAV, depending on the blowing wind. Our solution is able to spare a considerable amount of energy even though the UAV has to take a longer path. The next Part proposes an overview of the main techniques considered state of the art in sensors localization, especially from a practical point of view. We compared different algorithms, and show that the goodness of the results heavily depends on the quality of the hardware in use, especially good antennas are required to achieve relevant results. The final Part of the thesis consists of a survey on the most modern attacks on UAVs, from multiple points of view, such as the number of UAVs deployed, the type of technology mounted, and the way they fly. Roughly speaking, the more UAV deployed, the safer due to the increased difficulty in compromising multiple targets

at once; moreover, applied technologies will keep the UAV safer. Unfortunately, avoiding and protecting from all kinds of attacks is impossible, therefore we propose an affordable drones detection framework, able to detect an unauthorized (i.e., a compromised or rogue) drone as soon as it enters a restricted area, like an airport. It is a very accurate machine learning-based system, able to detect even drones never encountered before (e.g., a new brand of drones) exploiting only the data stream sent by the drone camera.

Prelude

The primary aim of this thesis is to try answering important questions coming to light from the usage of Unmanned Aerial Vehicles (UAVs) or drones, especially on 2 macro aspects: (i) how are UAVs facilitating everyday life, and (ii) how to cope, from a security and safety point of view, with issues arising from their daily use.

Both inquiries come from the fact that in recent years UAVs have been widely used for different tasks, and they will gain even more attention given their exponential rise in the market. Not only the military but also the industry recognize the UAV's potential to transform our world. Positive impacts are already taking place in different scenarios such as emergency services, environmental inspection, transportation, commercial deliveries, even security and monitoring, and many, many more. Reports estimate that the global UAV market will rise from $\sim$ 27 billion dollars in 2021, to approximately 41 billion dollars before 2026 and 1.4 million units sold.[1] Ranging from the amateur model to more complex ones, the size and power vary from less than 20 cm to almost plane-size specimens able to launch satellites into low earth orbit [2]. Although multi-rotors (i.e., helicopter-like drone) get most of the attention from the public market, they are not the only option; fixed-wing drones use wings like a normal airplane. They only need to use energy to move forward, not hold themselves up in the air, so they are much more efficient compared to the multi-rotors. Some examples in Figure 1: from left to right a small, average, and a big drone, with rotors and wings.

Figure 1: Different drones dimensions: from coffee cup-size to plane-size drones.

Drones encompass a wide range of different vehicles that can be guided by a pilot or are pilot-less. Usually, the pilot is on the ground and maneuver the drone in line-of-sight (LOS) or beyond visual line of sight (BVLOS) using the combination of a camera mounted on the drone and an LCD screen on the controller (Fig. 2 left side). Very often now, drones can be guided using a smartphone inserted into the

[1] https://www.researchandmarkets.com/reports/5406415/global-drone-market-report-2021-2026

[2] https://www.aviationtoday.com/2020/12/14/aevums-ravn-x-drone-offer-rapid-launches-low-earth-orbit/

controller (Fig. 2, right side) or even, downloading and installing an app., directly from the smartphone.

Figure 2: Different drones controllers.

The problems with BVLOS flights are related to safety: it is important that they achieve the same safety level as manned aircraft, therefore, posing no risk for the aviation. Moreover, BVLOS flights require satellite and ground-based connectivity, airspace segregation, trusted and resilient information exchange/ communications, differently from the LOS flights. Especially the use of satellite technologies for guidance and tracking raises no trivial question from a privacy and cyber-security view.

Probably the less known, from the public perspective, UAVs application is in agriculture, where those devices are already been introduced for building smart solutions: it is called smart agriculture (Figure 1, central). Even though the impact is less evident compared to other scenarios (e.g., delivery of goods), tasks like monitoring fields, efficient irrigation, spraying fertilizers, and harvester controlling are only some examples of fields improved by the UAVs deployment. Recently a lot of effort is spent on pests control and mitigation, such as the European Union's Horizon 2020 project HALY.ID: Innovative ICT tools for targeted monitoring and sustainable management of the brown marmorated stink bug and other pests[3].

Connected to agriculture, surveillance of forest and wild animal is another emerging topic: the introduction of UAVs have greatly helped the protection of endangered animals from poachers or increased defense against unauthorized deforestation. Even in the case of fire, UAVs can act fastly for detection and initial mitigation to reduce the damages to the forest and the surroundings.

There are a huge number of humanitary gains from the UAV usage, first and foremost organ and medical deliveries in remote medical centers or food and supplies to isolated communities dealing with the aftermath of a natural disaster or a pandemic. Nowadays, in cities, most of the medical supplies are delivered by

[3] https://www.haly-id.eu/

special couriers or ambulances via roads. The same roads that are subjected to traffic and accidents, very often slowing down the deliveries; UAVs can easily solve this problem given the capability to fly over any obstacle and impediments. For instance, in Europe, UAVs have been deployed as part of a project [4] to prevents deaths through the fast delivery of first aid kits and such. Even the COVID-19 pandemic has created possibilities, like monitoring people or the medical distribution of supplies avoiding contacts and thus contagion.

Besides medical-related supplies, also consumer goods and food packages deliveries have been announced by the Big company of e-commerce. Initially, it did not seem safe or practical to have tiny buzzing robots crisscrossing the sky. In the last two years, however, there are several reasons to believe that those types of package delivery by drone may be coming soon, including the new regulations laws, just released in 2021 by the Federal Aviation Administration (FAA), that allow operators of small drones to fly over people and at night under certain conditions. Regulation laws and technical solutions are meant to help in overcoming the main worries about drones, that is autonomous package-delivery drones might collide with an aircraft carrying people or might crash in high populated areas. So it is highly expected that very soon transportation companies can further extend their business relying on drones that cross the last-mile to their customers.

The incredible benefits of UAVs won't stop there: UAVs are way more sustainable compared to standard means of transportation, representing a "green" alternative by reducing emissions in the ecosystem. In Europe, transport is responsible for about 25% of the greenhouse gas (GHG) emissions and a major contributor to climate change [5]. By replacing standard vehicles, and introducing an electric vehicle fleet in tasks like agriculture and delivery, reduction in both carbon dioxide (CO2) and other airborne pollutants is taking place. Thus, UAVs have the potential to significantly reduce the negative environmental impact.

But all the gains are vain without safety and security as mentioned before. In fact, a very important step is to ensure security for and from UAVs. When mentioning UAVs, people's minds quickly jump to the military context. Unfortunately, here the facets are way deeper than that. Improper use of UAVs can cause great damage to the environment and harm people. Malicious users (a.k.a. attackers) can exploit mounted technologies to carry on different kinds of attacks, the first in line is the simple act of spying on people using the UAV camera. Not only UAVs can be used as a vector for many attacks, but they are themselves victims of them. Hacking and hijacking are the two most known UAVs attacks, where they fall under the control of a third-party system. From this point onward, the owner of the UAV changes, and therefore it is forced to do as the attacker wants. For

[4] https://www.tudelft.nl/en/ide/research/research-labs/applied-labs/ambulance-drone

[5] https://www.eea.europa.eu/ims/greenhouse-gas-emission-intensity-of

instance, UAVs are used to smuggle drugs over the US from Mexico[6]. There are also instances of civilian drones tampered with and used to carry bombs during outrages or in warfare[7] The most direct and simple countermeasure is to shoot them down, but this straightforward method will, with a very high chance, trigger the bomb too.

In conclusion, as the first thing, it is imperative to be able to detect and locate UAVs, to adequately grant security and safety for and from drones in those civilian applications that benefit from them.

[6] https://www.bbc.com/news/world-latin-america-30931367

[7] https://www.washingtonpost.com/world/national-security/use-of-weaponized-drones-by-isis-spurs-terrorism-fears/2017/02/21/9d83d51e-f382-11e6-8d72-263470bf0401_story.html

Publications

This thesis is divided into three main parts. Part I and II show how UAVs are facilitating everyday life and some practical limitations. Specifically, Part I tackles the problem of the last mile-deliveries of food and supplies with drones comparing it with a standard-track based system (Chapter 1), and how the performances of those systems can be improved by selecting the initial location for the delivery (Chapter 2) or even by selecting the trajectories of the drone considering a windy scenario (Chapters 3). Part II instead, proposes an overview of the sensors networks localization topic, useful in dealing with situations such as the aftermaths of natural disasters or even for search and rescue missions. A practical point of view, especially considering the issues arising from a real-world implementation is proposed in Chapter 4. Lastly, in Part III some issues from a security and safety point of view are tackled (Chapters 5 and 6).

Following the list of publications from which this thesis is written, in the same order as the chapters are presented. Therefore, the first four publications refer to the main topic of this thesis, the delivery system. Fifth and sixth publications refer to the limitations in implementing sensor localization system while the last two publications represent what is a more personal topic, safety and security from and for drones:

1. Giulio Rigoni, Bhumika, Cristina M Pinotti, Debasis Das, and Sajal K Das. Uav-based dataset: a case study. *Submitted to: the 95th Vehicular Technology Conference (VTC2022-Spring).*

2. Francesco Betti Sorbelli, Cristina M Pinotti, and Giulio Rigoni. On the evaluation of a drone-based delivery system on a mixed euclidean-manhattan grid. *Submitted to: Transactions on Intelligent Transportation Systems, Special Issue on Intelligent Supply Chain in Modern Challenges.*

3. Lorenzo Palazzetti, Cristina M Pinotti, and Giulio Rigoni. A run in the wind: Favorable winds make the difference in drone delivery. In *Proceeding of the 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 109–116. IEEE, 2021

4. Francesco Betti Sorbelli, Federico Corò, Lorenzo Palazzetti, Cristina M Pinotti, and Giulio Rigoni. How the wind can be leveraged for saving energy in a truck-drone delivery system. *Submitted to: Transactions on Intelligent Transportation Systems, Special Issue on Intelligent Supply Chain in Modern Challenges*

5. Francesco Betti Sorbelli, Cristina M Pinotti, and Giulio Rigoni. Range-free localization algorithms with mobile anchors at different altitudes: A comparative study. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*, pages 1–10, 2020

6. Francesco Betti Sorbelli, Sajal K Das, Cristina M Pinotti, and Giulio Rigoni. A comprehensive investigation on range-free localization algorithms with mobile anchors at different altitudes. *Pervasive and Mobile Computing*, 73:101383, 2021

7. Francesco Betti Sorbelli, Mauro Conti, Cristina M Pinotti, and Giulio Rigoni. Uavs path deviation attacks: Survey and research challenges. In *Proceedings of the 2020 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–6. IEEE, 2020

8. Anas Alsoliman, Giulio Rigoni, Marco Levorato, Cristina Pinotti, Nils Ole Tippenhauer, and Mauro Conti. Cots drone detection using video streaming characteristics. In *Proceedings of the International Conference on Distributed Computing and Networking 2021*, pages 166–175, 2021

9. Anas Alsoliman, Giulio Rigoni, Marco Callegaro, Marco Levorato, Cristina Pinotti, and Mauro Conti. Intrusion detection framework for invasive fpv drones using video streaming characteristics. *Submitted to: ACM Transactions on Cyber-Physical Systems*

Part I

Last-mile Delivery System

Probably the most discussed, and most important, UAVs logistic application is the delivery scenario, where the Big companies of E-commerce (e.g., Amazon[8] and Google[9]) can efficiently and effectively use drones for many reasons, like extending their business by increasing the number of customers to be served or assisting customers located at hard-to-reach places which would be impossible for ground vehicles (trucks) to visit in a timely manner. Initially, it did not seem safe or practical to have tiny buzzing robots crisscrossing the sky. In the last two years, however, there were several reasons to start believing that package delivery by drone may be coming soon, including the new regulations laws, just released in 2021 by the Federal Aviation Administration (FAA), that allow operators of small drones to fly over people and at night under certain conditions [10, 11, 12].

In the current *truck-based* delivery system, a single ground vehicle can accomplish, on average, 110-120 deliveries in a single day in an urban area, and 40-50 deliveries in a mixed-urban area [13]. Trucks can be stuck in a traffic jam and can delay the deliveries, and in the worst case, they could even cancel many of them. *Drones*, instead, have the ability to fly avoiding traffic, and in suburban areas they can also minimize the distance to travel by short-cutting the routes. Also, a recent research [14] has proven that drones are a means of transportation socially well accepted and considered useful. However, drones have peculiarities [15]. First of all, their payload is bounded in weight and size, and so at the moment, they cannot perform more than a single delivery at a time. In fact, they must go back to the distribution point, often the depot (warehouse), for loading on-board the package to be delivered to the next customer. Another constraint is that drones cannot fly beyond certain altitudes by law. In most of the countries, this limit is fixed to 120 meters [16, 17]. Lastly, they are strongly subjected to weather conditions, especially the wind can raise concerns during drone deliveries. Fortunately, a recent work [18] presented an energy model that permits to estimate the expected energy consumed given the UAV type and speed, the payload and the blowing wind, therefore posing the basis for a myriad of future researches.

Considering the aforementioned limitations, such as the limited payload, delivery of food and especially fast-food delivery with drones is very appealing. Quick Lunch Boxes, like pizzas, are well transportable by weight and bulk even by a medium commercial drone. Food delivery should be accomplished in a short time and usually is required in a burst at peak times, e.g., at dinner time, when the traffic on the roads is intense. So flying instead of driving avoids contention on the roads. The customers usually select the food provider nearby to avoid long waiting times and hence the drone flight range should be not too long for delivery food, as required by the drone batteries. At the same time, if drones will result fast

[8] https://www.aboutamazon.it/innovazioni/prime-air
[9] https://x.company/projects/wing/

enough, they could help providers to enlarge their business, thus making drones profitable for providers.

The rest of this Part is organized as follow:

Chapter 1 tackles the *last-mile deliveries* of food and supplies with drones comparing it with a standard *track-based* system using a real dataset collected in South America, Bogotà. Due to the lack in the literature of *UAV-based* delivery dataset, we initially created our own dataset using an open air simulator. Starting from the *truck-based* dataset, we considered only the feasible deliveries (i.e., compatible range with UAVs battery size) and ignored the payload due to the nature of it, that is lightweight fast-food. What we aim at, is to gauge the positive impacts of the UAV deployment in terms of times to deliver and total space traveled compared to the truck, and the impact of the wind on the flights. We tested four different wind scenarios (i.e., wind blowing from North, South, East, and West) and different UAV speeds and flight altitude and we found that the time to deliver and the total space traveled by UAVs far outperform the trucks. Given the UAVs capability to fly over structures, like tall buildings, and in general to pass over the natural landscape, we speculated that in urban scenarios UAVs have the upper hand because of traffic and building while in suburbs they can fly over parks and rivers. Even in presence of headwinds, the performances slightly degrade but are still better than the truck.

To increase the *UAV-based* delivery system performances, in Chapter 2 we study which is the better starting point for the UAVs, in other words, in this scenario we allow the UAVs initial position (a temporary depot, like a pod or a chart) to be moved in different places, based on the deliveries destination. Notice that the UAVs must come back to the temporary depot after every single delivery, before starting a new one. For this reason, we consider multiple UAVs flying simultaneously to different delivery locations, but this won't affect the results because we aim at minimizing the total distance traveled after all the deliveries. We modeled the delivery area as a mix of suburban area and urban area, wherein the former the UAVs can fly following a euclidean path while in the latter they must follow a manhattan path. We define multiple strategies for the selection of the depot and we evaluate their performances by: (i) considering different distributions of the deliveries inside the delivery area, and (ii) by changing the proportion of the type of areas (i.e., suburbs and urban). Results show that some approaches work better when the delivery distribution favors the suburban area, while others work better in the urban area, and this also depends on which type of area dominates the whole delivery area. As a final step, we also simulate the system using an open air simulator and compare the results between our approach and the optimal solutions obtained with a brute force technique, showing that our algorithm produces mostly

the same values but quickly.

Finally, in Chapters 3 we study how to select the specific trajectories for each delivery, especially considering the wind conditions. In this case, we assume a *hybrid truck-UAV system*; the truck moves following a predefined path, and at a certain position (i.e., *take-off* position) the UAV starts the flight from the truck, reaches the customer, and comes back to the truck that in the meantime has moved in a new position (i.e., *landing* position for the UAV) in his original path. Thus, we propose an algorithm to correctly and efficiently compute the best *take-off* and *landing* position for the drone in order to minimize the total energy spent to make the delivery in a windy scenario. The core idea is based on the fact that the wind blowing in the delivery area (i.e., the *global wind*) is not equal to the wind the drone faces while flying (i.e., *relative wind*) because the direction of the truck influences how the wind impacts the drone in his trajectory. Therefore, the proposed algorithm first calculates the wind that the drone faces and based on that, computes the best trajectory. Evaluating the system with both a synthetic wind dataset and a real one, shows that the proposed system spares lots of UAV energy, compared to the shortest path possible (i.e., euclidean path), even though the UAV has to travel more space, validating the assumption that choosing different trajectory in case of wind improves the UAV performances.

Chapter 1

A UAV-Based Dataset: a Case Study[1]

Food delivery is very important and it will be even more in the near future. Nowadays, it is done using small truck, motorcycle or in some specific and restrict area, even by bicycle. Our aim is to prepare and make available a delivery dataset of UAV deliveries, because at the moment, we are not aware of any kind of such dataset. In this Chapter, with the new dataset, we focus at identifying and starting to quantify the advantages of a UAV-based delivery food service compared to a standard truck-based delivery system. This is made possible thanks to a drone network simulator and a real test-bed of truck deliveries collected in Bogotá. Furthermore, we want to show how the wind relates to the UAV flight; in fact, it is the primarily "force" that impacts the UAV performance as the traffic does for the truck, but in some circumstances it can help the flight instead of disadvantage it.

Contribution: Our contributions are:

- the creation of the first UAV-based delivery dataset (UAV-DB) made available on GitHub;
- the identification of the most suitable UAV deliveries from a preliminary analysis on UAV-DB;
- the evaluation of the gain in time and distance using a drone with respect to a truck in Bogotá deliveries;
- the impact of tailwinds on UAV performances.

[1]This work has been submitted to VTC2022-Spring - The 2022 IEEE 95th Vehicular Technology Conference ⟶ Publication n. [1]

Organization: The rest of the Chapter is organized as follows. Section 1.1 presents some relevant works in the same field. Section 1.2 describes the dataset creation and simulation process. Lastly, in Section 1.3, we evaluate our solution comparing it with a standard truck-based system, considering different wind conditions and flight altitudes.

1.1 The related work

Literature offers some work regarding several aspects of deliveries in cities (e.g., path planning, salesman problem) and a fair number of papers are also related to the last mile delivery problem using standard means of trasportations (i.e., trucks). At the same time, UAVs are gaining a foothold in the delivery scenario, especially recently, after the Big company of e-commerce tested UAVs in different cities and the new regulations laws from FAA just released early this year. Therefore, also UAV-based delivery is gaining interest [19, 20].

Some attention has received the impact of the wind on the drone flight. Authors in [21] solve what they call the Mission-Feasibility Problem (MFP), where the impact of the wind on the UAV energy consumption is considered to state the feasibility of the delivery.

In [3], the Authors consider a truck-UAV system. The authors consider a drone carried by a truck and describe how to select the take-off and the landing point of the drone under wind to minimize the energy consumption. It is shown that under a favorable wind, even though the UAV travels more distance, the energy consumption is lower.

At the best of our knowledge, there are no papers that compare food deliveries made with trucks (or other wheeled vehicle) and drones.

1.2 Our system

In this Section, we first present our dataset, especially its composition and its origins. Then, we explain how we built it, focusing on its core assumptions and the tool we utilize for the purpose.

1.2.1 The Dataset

The first step is the creation of a UAV-based delivery dataset (i.e., UAV-DB) considering that in the literature, there are none available. We built our own UAV-DB simulating starting from an already known truck-based delivery dataset,

Table 1.1: Bogotá dataset main headers.

Headers	Description
Moment	Time of delivery. Morning, Afternoon or Noon
Name of Provider	Name of Provider
Expected Delivery Time	Expected time t reach the customer
Cost Delivery	Delivery cost for the customer
Latitude and Longitude	Starting point for the delivery
DailyTraffic	Daily traffic in colors (green, orange, red)
ClientLatitude and ClientLongitude	Customer location
Distance_mts	Distance from starting point to the customer
Time_sec	Actual time to reach the customer

referred from now as T-DB [22]. The most important columns of T-DB are described in Table 1.1. Using BlueSky open Air Traffic Simulator (ATS) [23], with a UAV plugin built-in, we run (by simulation) the original deliveries in Bogotá with drones.

To speed up the simulation process, T-DB underwent a pre-processing phase: we selected the deliveries whose Distance_mts is under 5km. After the pre-processing, the T-DB consists of a total number of ≈ 7000 deliveries that are simulated, in the next step, to create the UAV-DB dataset to be analyzed. The simulator creates .log files as simulations output, where at a fix time step (i.e., 1 second) different simulation values are saved for each UAV simulated. For instance, in the logs are reported the UAV altitude, ground speed, true airspeed, calibrated airspeed, latitude and longitude, and flown distance. Moreover, the heading field is added to store the direction of the straight line that connects the starting point with the customer. The heading field is reported in the meteorological coordinate system, whose axes are labeled with the 4 cardinal directions (in clock-wise order: North, East, South, and West). The log files created by ATS with different drone and wind speeds, and different altitude, are made available to the community under GitHub. Finally, a script parses the .log files and extracts information for all the simulated deliveries, and stores them in an organized manner in a .csv file, called UAV-DB. The UAV-DB maintains almost all the columns of T-DB, plus all the details about the UAV flight.

1.2.2 BlueSky Open Air Traffic simulator

To create the UAV-DB, first of all, we select the UAV to be used for the delivery task. In our experiments, the virtually deployed UAV is a DJI Matrix 600, a real UAV available on the market, which could be suited for light deliveries (it carries a payload up to 6 Kg and size 525 mm $\times$ 480 mm $\times$ 640 mm). Thus, DJI Matrix 600

could fit fast-food deliveries like the ones monitored in T-DB. Namely, the sources of the deliveries in T-DB are the food chain restaurants such as McDonald's, Domino's, Viva la Pizza, Burger King, and Kfc in Bogotá. According to DJI product information, DJI Matrix 600 can reach the maximum speed of 65km/h, which is approximately 20 m/s, and resists wind up to 10 m/s.

When a UAV is created, the default altitude is set to approximately 10 meters, after that the UAV has to climb to the desired altitude while moving towards the destination. Once the destination is reached, the UAV starts the descent to the ground. Considering two altitudes is beneficial for the system evaluation, giving us some insights related to the impact of altitude on the delivery time. Moreover, considering the two selected altitudes and the average building height in Bogota, we can ignore the constraint of crushing over obstacles, and a simple mapping of every tall skyscraper as "no-fly zone" is enough for secure delivery.

The wind is as important for UAVs as traffic is for trucks. For example, DJI informs that the drone safely withstands a maximum wind of 10m/s. We set and test two different wind speeds: 5m/s and 10 m/s, and the drone speed to 10 m/s and 20m/s. In our experiments, we noticed that the drone can assure the correct delivery process when the speed of the wind is below half of the UAV's speed.

The simulator uses for the wind and the UAV directions the meteorological coordinate system, which sets the 0 direction at the 90 degree of the Cartesian and numbers the degrees clockwise. In this Chapter, however, we refer to wind using the Cartesian coordinate system, which is more usual for us. Hence to facilitate the reader we report here our terminology both in Cartesian and meteorological coordinate systems. We consider the 4 main wind directions in the Cartesian coordinate system:

- 0 degree wind - the wind is blowing from West to East and has a meteorological direction of 90 degrees;
- 90 degree wind - the wind is blowing from South to North and has a meteorological direction of 0 degrees;
- 180 degree wind - the wind is blowing from East to West and has a meteorological direction of 270 degrees;
- 270 degree wind - the wind is blowing from North to South and has a meteorological direction of 180 degrees.

We also group the headings in 4 sectors S1, S2, S3, and S4 as shown in Table 1.2.

Table 1.2: UAV Headings.

Sector	Cartesian System	Meteorologic System
S1	$[0, 45] \cup [315, 360]$	$[45, 135]$
S2	$[45, 135]$	$[315, 360] \cup [0, 45]$
S3	$[135, 225]$	$[225, 315]$
S4	$[225, 315]$	$[135, 225]$

1.3 Evaluation

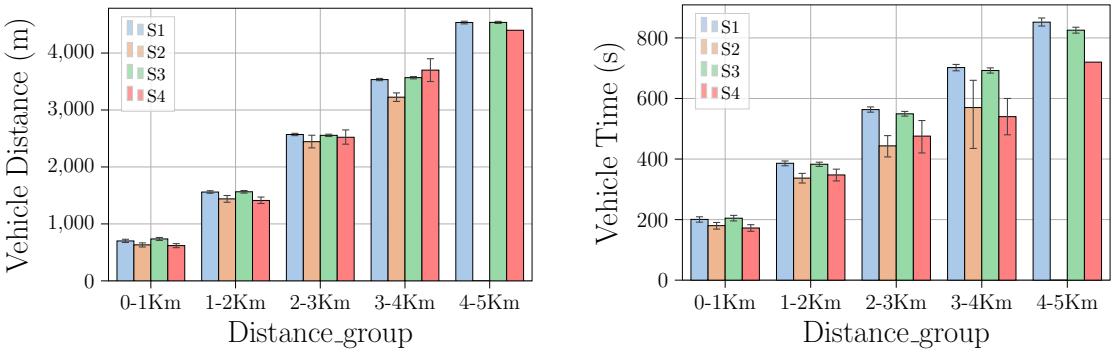
In this Section, we evaluate the characteristics of the UAV trajectories in absence and in presence of wind. Section 1.3.1 analyzes in terms of time to deliver and distance traveled the drone trajectory vs the truck road, considering different drone speed and different flight altitudes. In Section 1.3.2, we evaluate the UAV trajectory time and distance in presence of the wind with respect to the trajectories in absence of wind.

1.3.1 UAV-based delivery in absence of wind

In this Section, we compare a simulated UAV-based delivery system to the standard truck-based one, in terms of delivery time and traveled distance, while assuming the absence of wind during the flight. For a better understanding of the results, we group the deliveries into 4 different groups based on the vehicle delivery distance: (i) under 1km, (ii) from 1km to 2km, (iii) between 3km and 4km, and (iv) from 4km to 5km. Then, we reorder the deliveries in each distance group based on their heading direction as anticipated in Section 1.2.2. We will see that the grouping by headings is much more insightful than that by distances. Notice that some sectors are void because the dataset does not make any assumption on the customer distribution in the Bogotá area.

We extract from T-DB the average distance traveled by the truck in Figure 1.1(a) and the corresponding travel time in Figure 1.1(b). Those data are used as the baseline to which compare the UAV performances. For each distance group, the ground-trajectories in sector S1 and S3 are slightly longer than those in S2 and S4, with only one exception (group 3-4 km). The longer ground-trajectories take more time. The difference between S1 and S3 and S2 and S4 is much greater in time than in distance. Therefore, S1 and S3 are slower than S2 and S4. Examining the vehicle plots, in S1 and S3, on average, the vehicle speed is 5 m/s. In S2 and S4, on average, the vehicle speed is 6 m/s.

Figure 1.2(a) reports the distance covered by the UAV preserving the same grouping of the deliveries described above. It can be easily seen that the UAV



(a) (b)

Figure 1.1: Vehicle deliveries grouped by distance and sectors.

average distance is always smaller than that of the ground vehicle. Specifically, the deliveries in S2 and S4 cover a distance that is always slightly less than 500 m independently of the distance covered by the ground vehicle. Hence, we observe that to serve the deliveries in S2 and S4, the UAV trajectories are much shorter than the ground-vehicle trajectories. Instead, the UAV covers the deliveries in S1 and S3 by traveling about half of the distance of the ground vehicle.

In order to explain this behavior, first, we observe that the drone follows a straight line between the source and the destination. Experimental confirmation is shown in Fig. 1.3(a) by plotting on the map the latitude and longitude values recorded in the flight log produced by the ATS. So while the UAV trajectory simulated by ATS follows the Euclidean distance between source and destination, the distances traveled by the ground vehicle depend on the road map. The increase in the ground distance may depend on the road-map planning (i.e., the ground vehicle applies detours to avoid orographic obstacles) or longer routes have been selected (instead of the shortest road) to avoid traffic jam. The fact that on average the vehicle speed is higher in S2 and S4 trips seems to confirm that the longer trips are in less congested areas. The map of Bogotá seems to confirm that long roads are selected to avoid orographic obstacles: the city map has an elongated shape with parks to the north (S2) and south (S4). The vehicle makes a long loop to avoid entering the green areas, and it can go slightly faster than in S1 and S3 because S2 and S4 are areas less crowded, and thus less congested.

To quantify how much shorter is the UAV's journey, for each delivery, we compute the ratio between the distance traveled by the ground vehicle and the distance covered by the UAV. We then plot in Figure 1.2(b) the average of the delivery ratios along with the confidence interval at 95% level. The ratio is constant, equal to 1,5-2, and independent from the distance in S1 and S3. We deduce that the ratio 1,5 in S1 and S3 is the scaling factor to convert the UAV distance (Euclidean

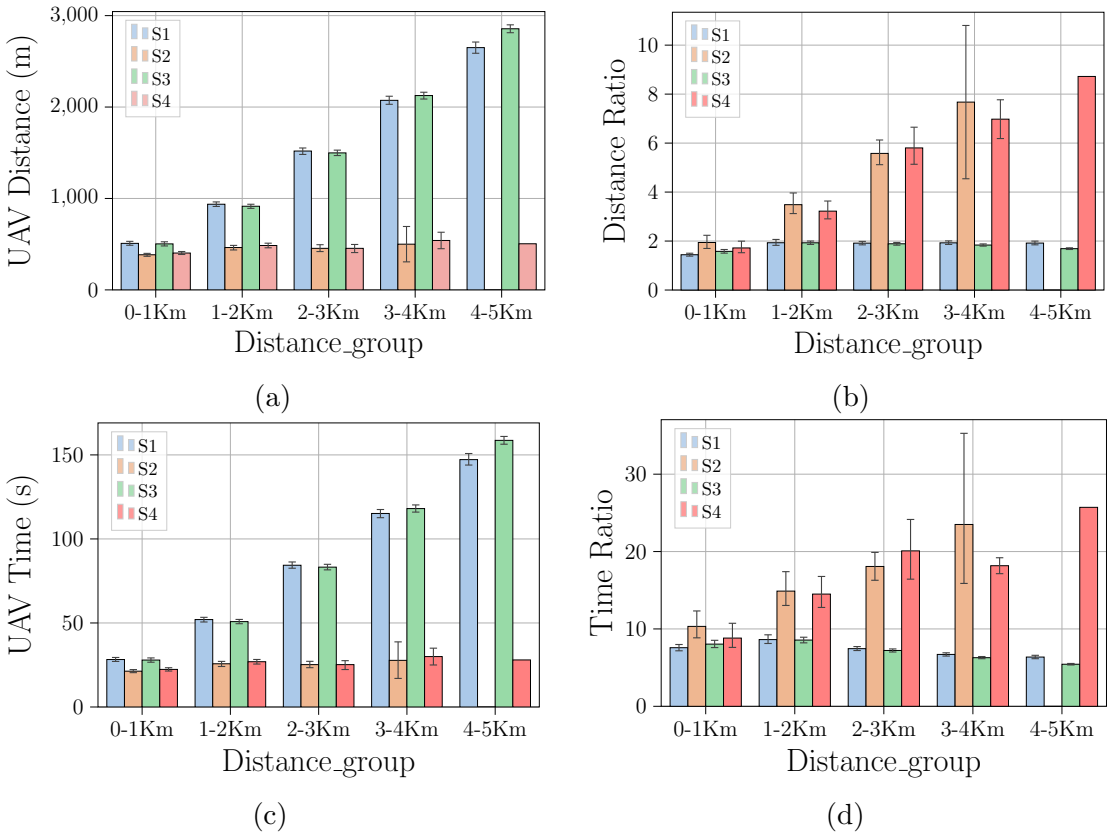
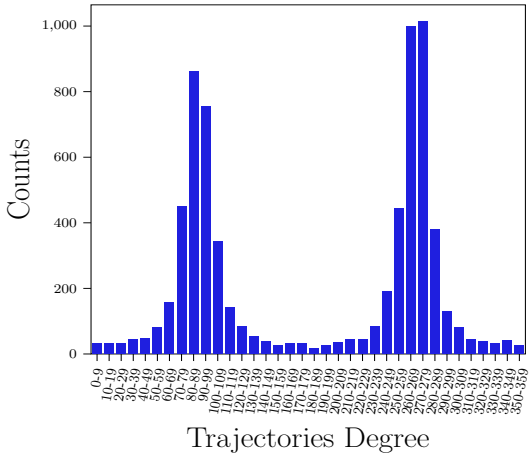


Figure 1.2: Deliveries grouped by Distance-group and trajectory heading with UAV moving at 20 m/s and altitude 100 m.

distance) into the vehicle distance in downtown Bogotá. Instead, in S2 and S4, the ratio between UAV trajectories and vehicle trajectories reaches 2 or more for the deliveries of the group 4-5 Km. Regard all the deliveries together in absence of wind, Figure 1.6(b) shows that the scaling factor between UAV and truck per km in S2 and S4 is between 2,3 and 2,8 when the drone moves at 20m/s. We also observed that the same ratio increases between 3 and 3,5 when the drone moves at 10m/s. Not surprisingly, the road-map is definitely longer than the Euclidean distance in suburban areas. From this analysis, we derive that the use of UAVs in suburban areas can be very advantageous, especially if the UAV proceeds slowly.

From this analysis, we can also learn some properties of the Bogotá road-map. The traveled vehicle distance is at least as long as the Manhattan distance in the center of Bogotá since the Cauchy-Schwarz inequality the ratio between the the Manhattan distance $|x_d - x_s| + |y_d - y_s|$ and the Euclidean distance $\sqrt{(x_d - x_s)^2 + (y_d - y_s)^2}$ between any source (x_s, y_s) and any destination (x_d, y_d) is always smaller than $\sqrt{2}$.

(a) An example of UAV trajectory fixed the source and the customer and changing the wind.



(b) Number of trajectory occurrence every 10 degree of the Meteorological Coordinate system.

It is also worthy to note that the time- and distance-ratio of the 3-4 km long deliveries in S2 have a very wide confidence interval. This can be explained with urban or suburban deliveries falling in that sector.

Regarding the flight duration in Figure 1.2(c), the duration of the UAV journeys is entirely consistent with the fact that in the simulation the drone speed is fixed to 20 m/s. As said, comparing the duration of the vehicle's trajectories with their average length, we derive that the vehicle moves at the speed of 5 m/s (or 18 Km/h) in S1 and S3 and 6.5 m/s in S2 and S4. These truck speeds, although so moderate, are realistic when the truck travels in a big city, with multiple intersections, often controlled by traffic lights, and in suburban districts.

In order to evaluate the gain in time using the drone, again, for each delivery, we calculate the ratio between the vehicle-trip duration and the UAV-trip. Then we plot in Figure 1.2(d) the average value of the ratios along with the confidence interval at 95% level. From the results, the UAV travels at least 5 times faster than the vehicle in urban (S1 and S3) areas and up to 8 times faster in suburban (S2 and S4) areas. These results are easily justified when one considers that the speed of the drone is at least 3 times that of the vehicle and the distance traveled by the drone is 2 to 10 times shorter than that of the vehicle. Since the ratio in Figure 1.2(d) is given by the ratio between the speed of the drone and that of the vehicle multiplied by the ratio of the UAV and truck distances in Figure 1.2(b), the results are justified. By the previous reasoning, the UAV gain in time is expected to decrease when the UAV speed decreases. When the UAV's speed halves, as in Figure 1.4(d), the ratio decreases with respect the one in Figure 1.2(b). However, the ratio does not halve as it is for the UAV speed. To

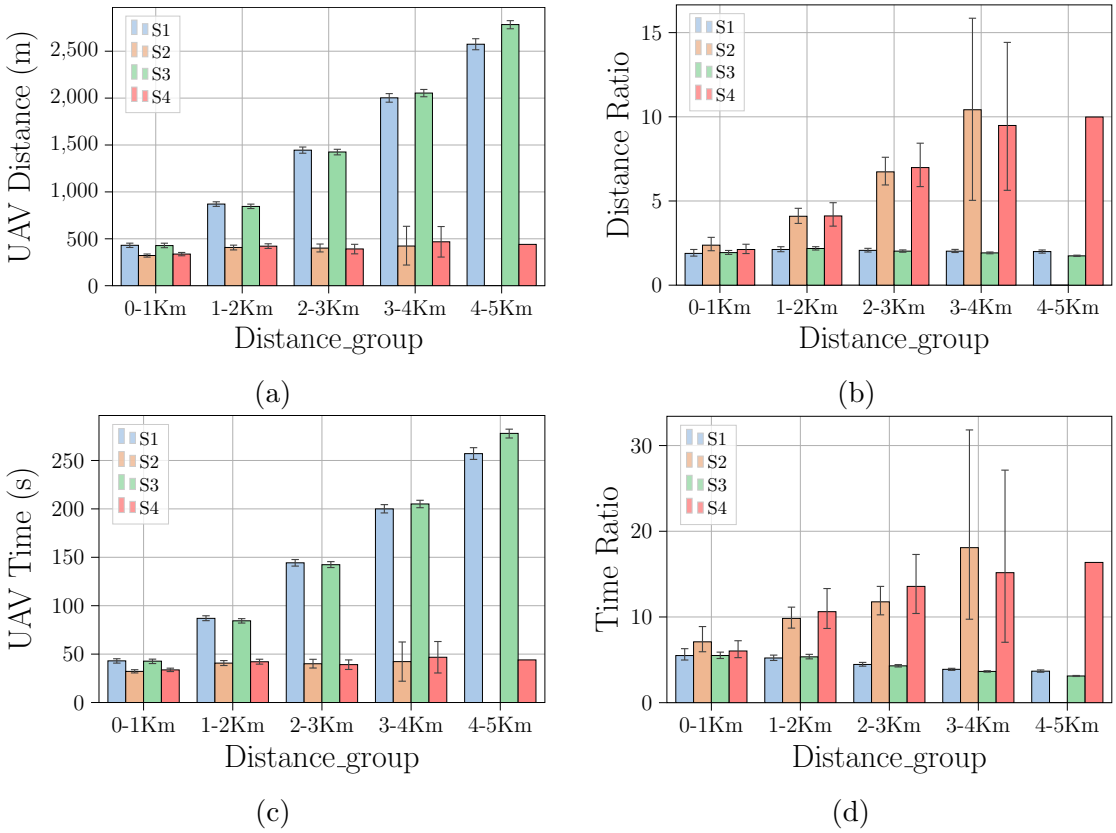


Figure 1.4: Deliveries grouped by Distance-group and trajectory heading with UAV moving at 10 m/s and altitude 100 m.

understand this behavior, we reconsidered the distance traversed and the duration of the drone path. As one can see, the distance decreases (see Fig.1.4(a)), when the drone speed is just 10 m/s, thus implying the drone trajectory is a function not only of the source and destination but also of the speed of the drone. As a consequence, the duration of the flight increases when the speed halves, but the duration does not double because the distance to be covered decreases. Therefore, the results in Figure 1.4(d) can be explained. We conclude that the role of the speed of the drone is very important and impacts all the flight parameters, including the trajectory. The role of the speed deserves more investigation.

Finally, Figure 1.5 analyzes the impact of the flight altitude on the covered distance and on the flight time when the UAV moves at 20m/s. We report the results for the minimum and maximum distance group only for the S1 and S2 sectors. Even though the values on the y-axes vary, the increase in time and distance is the same for the two distance groups and the two sectors. From these results, it seems that the increase just reflects the increase in the altitude (i.e., the

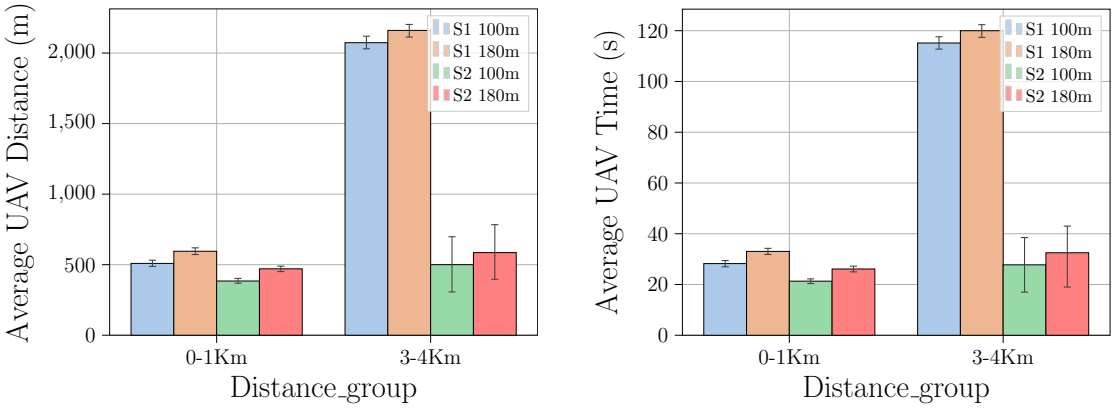


Figure 1.5: Impact of the altitude in the average UAV flight distance and time.

drone must traverse 160m more), and has no effects on other flight parameters.

1.3.2 The wind impact

In this Section we analyze the 4 wind scenarios described in Section 1.2.2, and to better understand how the wind influences the flight of a UAV, we keep the delivery grouped based on their heading. For each delivery heading (i.e., trajectory direction), we expect to see the effect of a favorable wind (i.e., wind with the same direction of the UAV heading), not favorable wind (i.e., a wind with a direction opposite to the heading), and lateral wind that partially affects the UAV. As an example, consider a UAV trip having heading 90. In this case, the 0 wind blows in favor and the UAV speed is incremented by the wind speed. Instead, a 180 degree wind pushes the UAV back and the wind speed decreases the UAV speed. The effect of the wind on the UAV trajectory can be in a simplistic way explained with the vectorial sum of the wind vector and UAV trajectory vector. Although this explanation leaves out important forces that oppose the wind, such as the thrust of the motor and the mass of the drone, from what we see in our experiments, the effect of the wind on the duration of the trip is relevant and obeys the vectorial sum rule at least for the tailwind and the headwind.

In the following, we abandon distance grouping and analyze the effect of wind on deliveries grouped only by headings. Since the effect of the wind depends on the trajectory heading, we first count the occurrences of the headings (see Figure 1.3(b)) in each sector. This will help us to interpret the results on the sectors.

In UAV-DB, the deliveries are not evenly distributed among the sectors: there

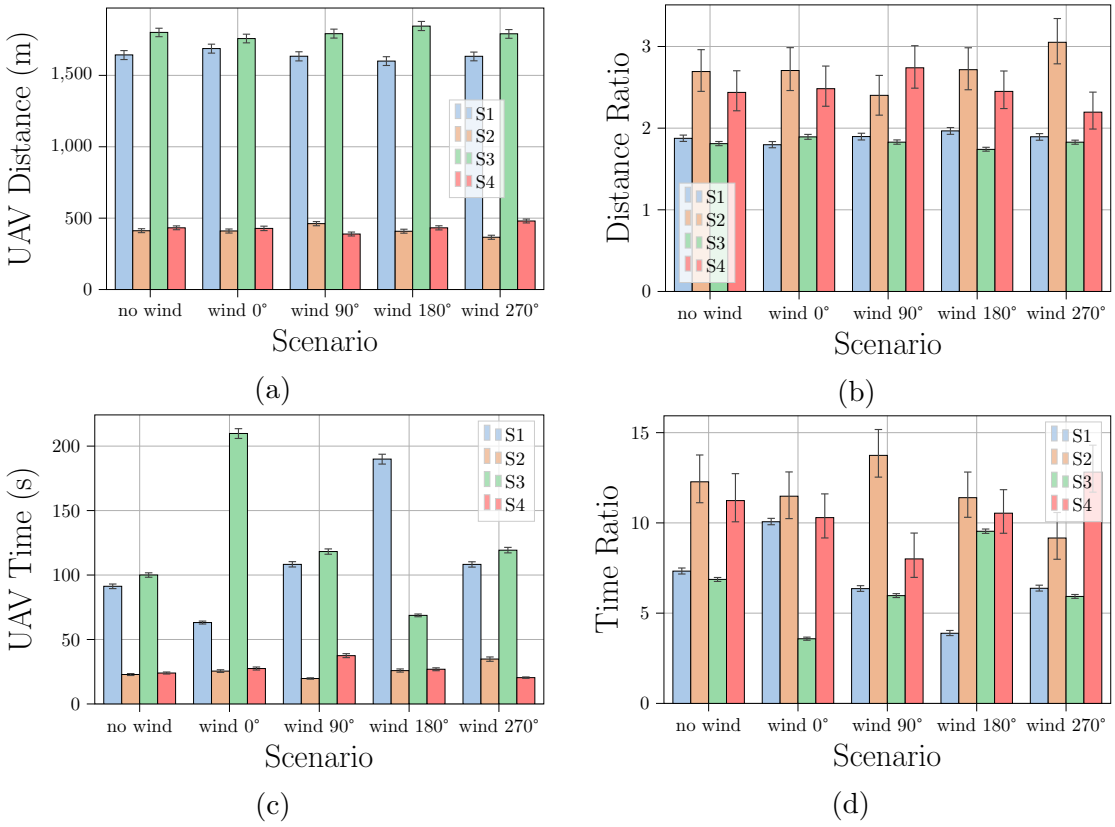


Figure 1.6: Barplots grouped by wind scenario and trajectory heading with wind speed 10 m/s and UAV moving at 20 m/s.

are many more deliveries in S1 and S3 than in S2 and S4. Moreover, in S1 and S3, the heading 90 and 270 dominate. We then expect that the wind 0 and wind 180 have more impact that the other wind scenarios.

Figure 1.6 describes how the 4 wind scenarios impact the time and the distance traveled by the UAV when the UAV moves at 20m/s and the wind speed is 10m/s. We report as a baseline the results in absence of wind.

The 4 wind scenarios have no tangible impact on the distance (see Figures 1.6(a) and 1.6(b)). The change in the distance is minor, and it is probably due to the wind pushing initially the UAV away from the original route, making it slightly turn and re-set the original path to the destination as depicted in Figure 1.3(a). It is probably the increased thrust of the motor that brings the drone back on track. This could probably be evaluated if we could measure motor thrust and/or power consumption.

The effect of the tailwind and headwind on the time of the flight is strong. As regards the flight time (see Figures 1.6(c) and 1.6(d)), when the wind blows

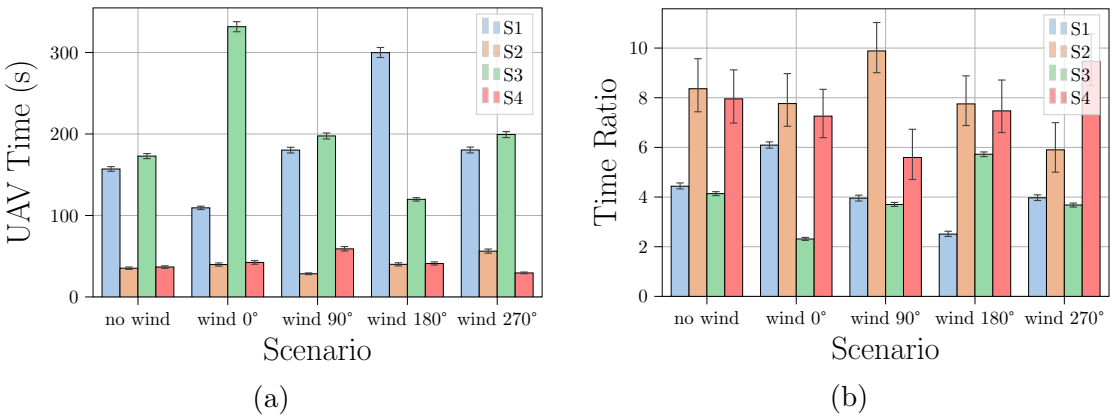


Figure 1.7: Barplots grouped by wind scenario and trajectory heading with wind speed 5 m/s and UAV moving at 10 m/s.

at 0 degrees, the flight time for the deliveries whose heading fall in Sector S1 significantly decreases, while the flight time for the deliveries whose heading fall in Sector S3 increases. The deliveries with heading 90 in S1 have the wind in favor, while the wind is against for those with heading 270 in S3. With respect to the baseline, S3 loses more than S1 wins. The deliveries in S3 move at an average speed of about 8 m/s (the UAV speed decreases by 12 m/s), while those in S1 at an average speed of 27 m/s (the UAV speed increases 7m/s). It is likely that the increase and decrease are different because the 270 heading in S3 is more represented than the 90 heading in S1. So, the deliveries in favor of wind are less than those with headwinds and the decrease is higher than the increase. With the wind at 180 degrees, the results are opposed as expected. The deliveries in S3 take less time than in the scenario with no wind, while those in S1 take a longer time. The wind is in favor of the heading dominant in S3 and against the heading dominant in S1. On average the speed in S3 is 28 m/s, while in S1 is 8.5 m/s. The UAV speed increase of 8m/s is higher than in the scenario with wind 0 because the deliveries with the wind in favor (i.e., heading 270) are more frequent (with wind 0, 90 was the heading with the wind in favor). Under the wind 0 and wind 180 scenario, S2 and S4 don't change much from the baseline. The directions in S2 and S4 are subject to a crosswind and the impact is minimal.

Under the wind 90 and 270 scenarios, the speedup is moderate with respect to the speedup achieved with wind 0 and 180 because there are fewer deliveries in S2 and S4 than in S1 and S3. The time flight for the deliveries in S1 and S3 which experience crosswind does not differ much from the baseline although many deliveries belong to S1 and S3. So we deduce that the impact of the crosswind is minimal, not comparable in strength with the impact of the tailwind or headwind.

Figure 1.7 reports how the wind affects the time of the flight if the UAV moves

at 10m/s and the wind speed is 5m/s. Figure 1.7 presents the same trends as Figure 1.6 both qualitatively and quantitatively. The time ratio in Figure 1.7(b) degrades with respect that in Figure 1.6(d), but it remains well above the half of Figure 1.6(d). As we already noticed in Figure 1.4, halving the speed of the drone, the gain is not halved because the covered distance is somehow shorter when the drone is slower. Again the drone seems to perform better when it does not run at the maximum speed, therefore confirming that the role of the speed of the drone is important also in presence of the wind.

In conclusion, we created the first UAV-based delivery dataset, obtained by simulating a real truck dataset in Bogotá. Our preliminary dataset shows that the drone is definitely faster than the truck both in the city and in the suburbs. So we believe it is worthy to explore the use of this means of transportation.

We found that with a favorable wind the drone and wind speed almost add up, while headwinds can slow down the drone. The impact of wind on the flight duration is definitely crucial for deriving the expected delivery time to the customer. Therefore, the wind cannot be ignored devising a delivery system based on drones.

Finally, the UAV speed impacts all the aspects of the flight: time, duration, and effect of the wind. It is then crucial to better understand the UAV speed role for implementing a UAV food delivery system.

Although these early results are very interesting, more work is required on the following aspects that are currently not covered by the simulator: (i) the energy consumed of the drone, (ii) the impact of the payload on all the aspects of the flight.

Beyond the delivery food system, this preliminary study offers also interesting research sparks, like the possibility of obtaining information on the overflown territory from the superimposition of information deduced from different means of transport.

Chapter 2

Drone-Based Delivery System on a Mixed Euclidean-Manhattan Grid[1]

In this Chapter, we focus on a drone-based delivery scenario. Given that the drone has to deliver one package at a time, it is crucial to define a suitable starting point for any subset of deliveries. Usually, in a rural area formed by low buildings and houses, a drone can easily fly on straight lines between any two locations. On the other hand, in a downtown area of a big city formed by tall buildings and possibly skyscrapers, the drone's flight is more constrained due to the presence of obstacles. Considering that a drone cannot fly beyond the maximum altitude, inside a city a drone can only fly over the roads/streets. We summarize these two different contexts as follows: in the rural scenario, the drone moves freely according to the Euclidean metric, while in the urban scenario, the drone moves according to the Manhattan metric [24]. We model this delivery area as a Euclidean-Manhattan-Grid (EM-grid), where customers reside on both the areas. Now, given a subset of customers inside an EM-grid to be served by a single drone, the objective is to find the temporary depot, called *distribution point*, that minimizes the sum of distances between all the customers and the distribution point. Recall that the drone has to perform multiple round trips to/from the distribution point.

[1]Part of this work has been published to (DCOSS), Wi-DroIT workshop - the 15th International Conference on Distributed Computing in Sensor Systems, and the full work has been submitted to IEEE T-ITS - Transactions on Intelligent Transportation Systems journal ⟶ Publications n. [2]

Contribution: The contributions of this Chapter are summarized as follows.

- We present the EM-grid model which characterizes the delivery area for the drone.
- We define the Single Distribution Point Problem (SDPP), and devise time-efficient algorithms for computing the distribution point. Our solution is optimized with respect to the set of customers and computationally light because it needs to be recomputed every time the set varies.
- We extensively and comprehensively compare the performance of our presented algorithms.
- Using an open air simulator called BlueSky, we compare the performance of our best solution with that of fixed distribution points, emulating the depot's location of a delivery company.

Organization: The rest of the Chapter is organized as follows. Section 2.1 reviews the relevant related work. Section 2.2 formally defines the SDPP and shows the optimal solution. Section 2.3 presents time-efficient algorithms for solving SDPP. Finally, Section 2.4 evaluates the effectiveness of our algorithms, and compares our solution with the fixed solution through a simulator.

2.1 Related Work

In the literature, there are many works solving the last-mile delivery problem with drones. In the seminal work [25], the authors study the cooperation between one truck and one drone to deliver packages to customers. The authors introduced the flying sidekicks traveling salesman problem (FSTSP), a variant of the TSP, in which a drone first takes off from the truck, then proceeds to deliver goods to a customer, and finally rejoins the truck in a third location. As the drone flies, the truck makes deliveries to other customers, but has to stop waiting for the drone at the rendezvous location. An optimal mixed-integer linear programming (MILP) and two heuristic solutions are proposed to solve problems of practical sizes.

The authors in [26], instead, tackle the problem of solving the traveling salesman problem with a drone (TSP-D), using a drone that rides on the truck. A branch and bound technique is applied for solving the TSP-D. The Euclidean and Manhattan metrics have been considered in [27] to evaluate distribution systems. The delivery area is modeled as a circular region with a central warehouse and customers randomly distributed throughout the region. In the area, 13 optimization algorithms are simulated. The comparison in terms of time and traveled

distance, measured with both Euclidean and Manhattan metrics, shows that different algorithms perform better in different situations, therefore authors recommend employing different algorithms depending on the neighborhood.

The importance of a strategic planning on urban delivery services is also studied in [28]. Specifically, the preferred method and local impacts of vehicle trip may vary by neighborhood characteristics (e.g., traffic or customer demands). Instead of searching for an optimal route, the authors focus on the estimation of the vehicles miles traveled (VMT) per order, considering different types of neighborhoods, delivery scenarios, and strategies. The system is evaluated in Chicago, showing that alternative delivery strategies can largely reduce the VMT per order based on the type of neighborhood.

Although these two papers do not consider drones, they impact our work because they underline the importance of planning different delivery strategies depending on the delivery scenario. They also suggest that distance is one of the important parameters for evaluating a delivery system.

In this Chapter, we focus on a delivery area called EM-grid and firstly introduced in [24] that is formed by two contiguous areas, i.e., rural and urban. In the former area, the drone connects two locations by following the unique line that passes through them, while in the latter area, due to the tall buildings and the altitude limits, the drone flies over the roads. Simplifying, in the rural area, the drones' movements are modeled by the Euclidean metric, while in the urban one by the Manhattan metric. In [24], the drone's mission consists of performing one delivery for each destination of the grid. The authors envisioned the drone like the mailman in days gone. That is, the authors expect that the drone will drop off something every day in each house. In that scenario, considering that the drone has to go back to the depot after each delivery and has to serve all the points of the grid, the problem is to find the location for the depot that minimizes the sum of the traveled distances. The exact solution of the problem with a delivery for each grid point is computed in logarithmic time in the length of the side of the EM-grid. A similar approach is investigated in [29], assuming that the drone's mission visits only a subset of the grid's vertices. The scenario is slightly different because the drone moves inside a warehouse, and the vertices are items (of customer orders) to be collected on shelves.

In [21], authors consider the problem of delivering goods using a drone when environmental factors are present, especially the wind. They propose a framework based on time-dependent cost graphs to model the problem considering that a drone can serve a single customer before returning to the warehouse. The delivery area map, is therefore modeled with a weighted graph whose costs are time-dependent (i.e., the wind can change affecting the energy consumption of the drone), and fixed topology. To solve the problem, three approaches are proposed,

i.e., offline, online, and greedy online. The performance is evaluated in terms of flight missions accomplished with success with a given battery energy budget.

Focusing on computation time and exploiting GPU parallel computation, the work in [30] proposes a centralized system for computing energy-efficient time-varying routes for drones in a multi-depot multi-drone delivery system. The authors devise a centralized parallel algorithm called Parallel Shortest Route Update (PSRU) that constantly updates the drones' delivery routes avoiding the whole recomputation from scratch, becoming 4.5× faster than the state-of-the-art algorithms.

Authors in [31] investigate the collaboration among a truck and multiple drones in a last-mile package delivery scenario, introducing the multiple drone-delivery scheduling problem (MDSP). Each delivery has an energy cost associated, a reward based on the priority of the delivery, and both launch and rendezvous times with the truck. The work aims at finding the optimal scheduling for the drones that maximizes the overall reward, subject to the drone's battery capacity while ensuring that the same drone performs deliveries that do not overlap. Results show that the presented problem is NP-hard, and therefore, different heuristic for solving the problem in a time-efficient way are proposed.

Last, but not least, delivery with drones may impact environmental sustainability (i.e., energy consumption and greenhouse gas emissions). The authors in [32] propose a mixed-integer (0-1 linear) green routing model with traffic restrictions, and a genetic algorithm to efficiently solve the complex routing problem, showing that drones can accomplish more deliveries and at a lower cost (in terms of CO_2 emissions and energy consumption) compared to standard methods, and that traffic types impact on the results.

2.2 Problem Definition

In this section, we first introduce the system model which characterizes the delivery area and how the drone moves inside it, and then we formally describe the problem to solve. Our aim is to find the distribution point that minimizes the sum of distances between the distribution point and all the customers to be served by the drone. The distribution point is a location inside the delivery area that depends on the set of customers. So, changing the set of customers, the distribution point varies. To implement our solution, the drone has to work in symbiosis with a mobile pod that brings all the packages to be delivered for that particular set of customers at the selected distribution point. We neglect the transportation cost (additional distance) for moving the mobile pod from the main depot to the distribution point because such a cost is paid just once for any set of customers and does not affect the drone's resources.

2.2.1 The System Model

The positions of the customers to be served lie inside a *Euclidean-Manhattan-Grid* (EM-grid). This area is defined as a 2-dimensional grid $G = (R, C, K)$ which consists of R rows, C columns, and a column parameter $K \in [1, C]$ (which defines the *Border* B) that separates the *Euclidean* sub-grid E (i.e., the rural area) from the *Manhattan* sub-grid M (i.e., the urban area). Specifically, $E = \{1, \ldots, R\} \times \{1, \ldots, K\}$, $B = \{1, \ldots, R\} \times \{K\} \subseteq E$, and $M = \{1, \ldots, R\} \times \{K + 1, \ldots, C\}$. The border B consists of the single column K.

Conventionally, the delivery area is only Euclidean if $K = C$, and only Manhattan when $K = 1$. In an EM-grid, any internal vertex $u = (r_u, c_u)$ of G, i.e., with $1 < r_u < R$ and $1 < c_u < C$, is connected to the four adjacent vertices $(r_u, c_u \pm 1)$ and $(r_u \pm 1, c_u)$; whereas, in general, any vertex of the grid, i.e., with $1 \leq r_u \leq R$ and $1 \leq c_u \leq C$, is connected only to the existing adjacent vertices (i.e., an external vertex has only three or two neighbors). For simplicity, we assume that the distance between any two pairs of consecutive vertices on the same row or column is constant and unitary. Figure 2.1 shows an example of EM-grid.

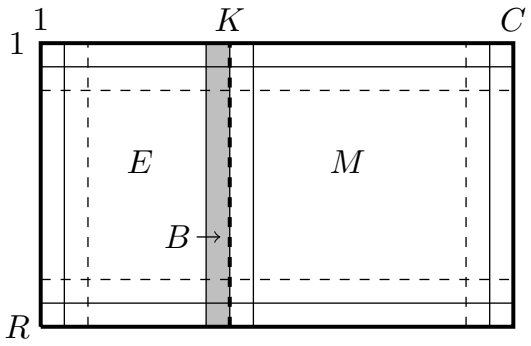


Figure 2.1: The EM-grid with R rows, C columns. Each cell identified by the pair row and column, represents the "block" of a customer.

The drone follows either the Euclidean metric when it moves inside the rural area or the Manhattan metric when it moves inside the urban area. In fact, for any two vertices u, v in G, the distance $d(u, v)$ is the length of the shortest path traversed by the drone in the EM-grid to go from $u = (r_u, c_u)$ to $v = (r_v, c_v)$. Recalling that the Euclidean and Manhattan distances are defined, respectively, as:

$$d_E(u, v) = \sqrt{(r_u - r_v)^2 + (c_u - c_v)^2}$$
$$d_M(u, v) = |r_u - r_v| + |c_u - c_v|,$$

then, the distance $d(u, v)$ for $u, v \in G$ is given by:

$$d(u,v) = \begin{cases} d_E(u,v) & \text{if } u,v \in E \\ d_M(u,v) & \text{if } u,v \in (M \cup B) \\ \min\limits_{w \in B}\{d_E(u,w) + d_M(w,v)\} & \text{if } u \in E, v \in M \\ d(v,u) & \text{if } u \in M, v \in E \end{cases}$$

where $d(u, v) = d(v, u)$ because both Euclidean and Manhattan distances are symmetric.

By Lemma 1, the shortest path $d(u, v)$ is unique and passes through the vertex in row v on the border B.

Lemma 1 ([24])**.** *Consider an EM-Grid $G = (R, C, K)$. Given $u = (r_u, c_u) \in E$ and $v = (r_v, c_v) \in M$, then $d(u, v) = d_E(u, h) + d_M(h, v)$ with $h = (r_v, K)$.*

Thus, from now on, $d(u, v)$ is finally given by:

$$d(u,v) = \begin{cases} d_E(u,v) & \text{if } u,v \in E \\ d_M(u,v) & \text{if } u,v \in (M \cup B) \\ d_E(u,h) + d_M(h,v) & \text{if } u \in E, v \in M \\ & h = (r_v, K) \in B \\ d_M(u,h) + d_E(h,v) & \text{if } u \in M, v \in E \\ & h = (r_u, K) \in B \end{cases} \tag{2.1}$$

2.2.2 The Problem Formulation

The basic task in a delivery area is to serve a subset of customers with the help of a drone. Due to payload constraints, the drone cannot serve all the customers on the same flight, and it has necessarily to go back and forth from a specific position inside the delivery area, called *distribution point*, to all the customers. In an EM-grid, a drone has to fly from the distribution point to all the customers. A *mission* $\mathcal{M} = \cup_{i=1}^{m}\{u_i^{(t_{u_i})}\}$ for a drone consists of m distinct customers (vertices of G), denoted as u_i, with $i = 1, \ldots, m$, each with *multiplicity* $t_{u_i} \geq 1$. That is, the customer u_i has ordered t_{u_i} different packages that the drone has to separately deliver. The mission $\mathcal{M}$ consists of overall $n = \sum_{i=1}^{m} t_{u_i}$ packages to be delivered. Let $\mathcal{M}_E$ be the subset of customers which reside in E, and $\mathcal{M}_M$ the subset of customers which reside in M. Clearly, $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_M$. Let $n_E = \sum_{u \in \mathcal{M}_E} t_u$ and $n_M = \sum_{u \in \mathcal{M}_M} t_u$. Moreover, let the *cost function* for the drone applied to $\mathcal{M}$

having set the distribution point at $u \in G$ be:

$$\mathcal{C}(\mathcal{M}, u) = 2 \sum_{v \in \mathcal{M}} t_v d(v, u) \tag{2.2}$$

that is, the cost function is the sum of the distances between any item $v \in \mathcal{M}$ and the point u. Notice that the multiplicative constant 2 considers the round trip for each delivery. To ease the notation, we can write $\mathcal{C}(u)$ instead of $\mathcal{C}(\mathcal{M}, u)$ because $\mathcal{M}$ is assumed to be invariant. Our aim is to find the optimal distribution point $u^* = (r^*, c^*)$ in G that minimizes the cost function:

$$u^* = \arg\min_{u \in G} \mathcal{C}(u). \tag{2.3}$$

We denote this problem as the Single Distribution Point Problem (SDPP) because the goal is to find the *single* distribution point in EM-grids.

2.2.3 The Optimal OPT Algorithm

The brute-force algorithm OPT optimally solves SDPP. It sequentially considers each vertex $u \in G$ as candidate to be the distribution point, and returns the point u^* that minimizes Eq. (2.3). Hence, since the grid G consists of R rows and C columns, for a given mission $\mathcal{M}$ with n packages to deliver, the time complexity for finding the distribution point u^* by performing an exhaustive search is $\mathcal{O}(nRC)$.

Since both R and C can be exponentially large with respect to n, OPT is a pseudo-polynomial algorithm. So, in the next section we devise better time-efficient approximation and heuristic algorithms for solving SDPP.

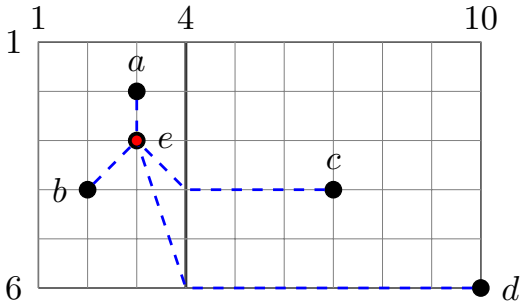


Figure 2.2: An EM-grid $G = (6, 10, 4)$ with $a = (2, 3)$, $b = (4, 2)$, $c = (4, 7)$, $d = (6, 10)$, and $e = (3, 3)$. The u^* is in red in position $(3, 3)$. The dashed lines that connect all the points with u^* represent the paths of the drone.

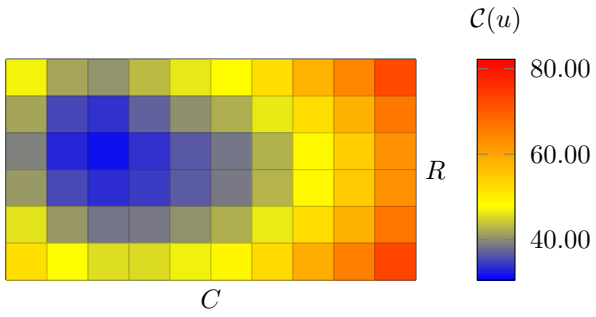


Figure 2.3: Heatmap of the example in Figure 2.2. The colors illustrate the value $\mathcal{C}(u)$ for each $u \in G$. The best points are the ones with *cold* colors (blue), while the worst ones are in *hot* colors (red).

Figure 2.2 highlights an example of EM-grid which will be used for describing all the presented algorithms in Section 2.3. By running the optimal algorithm OPT that exhaustively tries each vertex as solution, the distribution point that minimizes Eq. (2.3) is $(3, 3)$. When using the point u^*, the total cost by computing Eq. (2.2) for the drone is, approximately, 31.98.

Finally, Figure 2.3 shows the importance of choosing a good distribution point in the delivery area. Each *square* represents a vertex $u \in G$ whose *color* quantify the value $\mathcal{C}(u)$. From Figure 2.2, we know that the optimal point is $u^* = (3, 3)$, and hence, according to the adopted coloring, its square is the *coldest* one (i.e., blue). It is worth nothing that, although the example is very small, if we randomly select a distribution point (e.g., the top and bottom right corners) the total cost is more than twice the optimal cost. At the same time, for reasons of symmetry, there is a subset of candidate points whose cost is very close to the optimal cost. Please note that the costs in Figure 2.3 are relative to the example in Figure 2.2. Changing the customers' positions, the heatmap changes and thus the optimal and the other (possible) distribution points.

We now focus on two special cases of SDPP when $m = 2$, i.e., when there are only two customers. In general, finding the optimal solution with an arbitrary number of customers is not trivial. However, in the following scenario with only two customers, we can reduce the number of possible candidate distribution points and hence determine the optimal solution.

Lemma 2. *Given $\mathcal{M} = \{u^{(t_u)}, v^{(t_v)}\}$, then the distribution point u^* belongs to the shortest path between u and v.*

Proof. We select a vertex $q \in G$ that does not belong to the shortest path between u and v. Without loss of generality, let $t_u \geq t_v$. Let p be the vertex of the shortest path such that $d(u, p) = d(u, q)$. Then, the cost $\mathcal{C}(q)$ of setting the distribution

point at q yields:

$$\begin{aligned}
\mathcal{C}(q) &= t_u d(u,q) + t_v d(q,v) \\
&= t_u d(u,q) - t_v d(u,q) + t_v d(u,q) + t_v d(q,v) \\
&= (t_u - t_v) d(u,q) + t_v (d(u,q) + d(q,v)) \\
&\geq (t_u - t_v) d(u,q) + t_v d(u,v) \\
&= (t_u - t_v) d(u,p) + t_v d(u,v) \\
&= t_u d(u,p) + t_v (d(u,v) - d(u,p)) \\
&= t_u d(u,p) + t_v d(p,v) = \mathcal{C}(p)
\end{aligned}$$

So, any point q outside the shortest path increases the cost. □

When the two delivery points have the same multiplicity, not only the points of the shortest path are candidates, but all of them optimize the cost. Specifically,

Lemma 3. *Given $\mathcal{M} = \{u^{(t_u)}, v^{(t_v)}\}$, if $t_u = t_v \geq 1$, then, any point p along the shortest path between u and v can be the distribution point.*

Proof. When $t_u > t_v$, the best position of the distribution point is u, i.e., the vertex with the largest multiplicity. Namely, for any point p, with $p \neq u$ along the shortest path between u and v, it yields:

$$\begin{aligned}
\mathcal{C}(p) &= t_u d(u,p) + t_v d(p,v) \\
&= t_u d(u,p) + t_v d(u,v) - t_v d(u,p) \\
&= t_v d(u,v) + (t_u - t_v) d(u,p) \\
&> t_v d(u,v) = \mathcal{C}(u)
\end{aligned}$$

□

These preliminary observations motivate our further investigation in the next section.

2.3 Proposed Algorithms

In this section, we propose time-efficient approximation and heuristic algorithms that solve SDPP. In particular, we devise two algorithms that neglect the two mixed metrics and determine the distribution point just assuming the whole area as fully Euclidean (GEC) or as fully Manhattan (GMM). Then, we show two more algorithms that force the distribution point to reside in the Euclidean area (ECMB) or in the Manhattan area (MMEB). It is important to note that

to determine the distribution point, we only need to know the positions of the customers and the position of the border in EM-grid. The width of the Euclidean and Manhattan area do not play any role in determining the distribution point. Finally, we present an algorithm that returns the best among the previous four solutions (APX).

2.3.1 The GEC Algorithm

The *Global Euclidean Centroid* (GEC) algorithm is a heuristic that solves SDPP by pretending EM-grid as a full Euclidean grid. In the literature, for the *Euclidean space* the optimal distribution point is called as *geometric median* [33]. No any explicit formula nor an exact algorithm involving only arithmetic operations and k-th roots can exist in the Euclidean space [34] (note that in such a case the brute-force approach cannot work because there is an infinite number of candidates). In the Euclidean space, the geometric median has been approximated with the *centroid*, which is is the point that minimizes the sum of squared Euclidean distances between itself and each point in the set. The great advantage of the centroid is that it is very simple to be computed because it is the arithmetic mean position of all the input points. Moreover, experimentally, the cost associated with the centroid is in the vast majority of the cases very close to that of the geometric median [35].

By analogy to the Euclidean space, the *Global Euclidean Centroid* (GEC) algorithm solves SDPP in an EM-grid by selecting the *centroid* of the n customer locations. To be more precise, given a mission $\mathcal{M} \in G$, the centroid u_C is:

$$u_C = \left(\frac{\sum\limits_{u \in \mathcal{M}} t_u r_u}{n}, \frac{\sum\limits_{u \in \mathcal{M}} t_u c_u}{n} \right). \tag{2.4}$$

The GEC algorithm selects the centroid u_C as the distribution point. Note that since G is a discrete grid, we have to round each individual coordinate of u_C to the closest row and column, thus introducing a further error. Moreover, according to Eq. (2.4), u_C can be computed in $\mathcal{O}(n)$ time.

Observation 1. A lower bound for the approximation ratio of the GEC algorithm is 2. Consider the configuration illustrated in Figure 2.4.

The two vertices u and v (that represent two customers) on the same row have multiplicity k and $n - k$, respectively, with $k \ll n$. It is easy to see that $u^* = v$, while, by Eq. (2.4), u_C is closer to v than u. The cost of setting the distribution point at u_C is $k\left(\frac{n-k}{n}\right) d + (n-k)\left(\frac{k}{n}\right) d = 2k\left(\frac{n-k}{n}\right) d$, while the optimal cost of setting the distribution point at $u^* = u_M$ is kd. For large values of n, the ratio

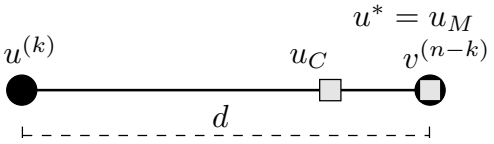


Figure 2.4: A particular instance when invoking GEC.

between this solution and the optimal solution tends to 2. Note that, in this configuration we neglect the position of the border K since GEC assumes the delivery area as a full Euclidean area. Also, this illustration does not guarantee that in general the *upper bound* for the approximation ratio is 2. For this consideration, GEC is not an approximation algorithm.

If we perform the GEC algorithm on the example of Figure 2.2, the average row among $2, 4, 4, 6, 3$ is 3.8, while the average column among $3, 2, 7, 10, 3$ is 5. By rounding the average row, we have that $u_C = (4, 5)$, which belongs to the Euclidean area. In this case, the total cost is 35.30 and the ratio $\frac{35.30}{31.98} = 1.104$, meaning that in this example, the cost of GEC is 10% more than the optimal cost.

2.3.2 The ECMB Algorithm

The *Euclidean Centroid by projecting M to B* (ECMB) algorithm solves SDPP by forcing the distribution point to reside in the Euclidean area. Consider a mission $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_M$. If the distribution point is selected in the Euclidean side, to serve each destination in the Manhattan area, the drone crosses the border B and then flies horizontally up to the destination (see Eq. (2.1)). Overall, the drone covers the distance $H = \sum_{u \in \mathcal{M}_M} (c_u - K)$ in M for serving the customers $\mathcal{M}_M$. The distribution point is then selected as the centroid of $\mathcal{M}' = \mathcal{M}_E \cup \mathcal{M}'_M$, where $\mathcal{M}'_M$ replaces each original customer $u = (r_u, c_u)$ in the Manhattan side with its projection $u' = (r_u, K)$ on the border B. Namely, one can see that all the points in $\mathcal{M}'$ reside in E. Hence, as explained in Section 2.3.1, a good distribution point is the vertex:

$$u_{\hat{C}} = \left(\frac{\sum\limits_{u \in \mathcal{M}} t_u r_u}{n_E + n_M}, \frac{\sum\limits_{u \in \mathcal{M}_E} t_u c_u + \sum\limits_{u \in \mathcal{M}_M} t_u K}{n_E + n_M} \right), \tag{2.5}$$

where $u_{\hat{C}}$ is the centroid of $\mathcal{M}'$. Eventually, the overall cost is given by H plus the sum of the distances traversed in E to move the points in $\mathcal{M}'$ to $u_{\hat{C}}$. Note that, $u_{\hat{C}}$ can be computed in $\mathcal{O}(n)$.

It is important to stress that ECMB would not return the optimal point u^* even if we know in advance that the optimal distribution point resides in the

Euclidean area since the centroid is not optimal for GEC, as previously discussed in Observation 1.

Observation 2. The approximation ratio of the ECMB algorithm in unbounded. In fact, consider Figure 2.4 and the value of the border K. Since ECMB forces the distribution point in the Euclidean area, if the border crosses the vertex v or it is to the right of v, then the analysis is exactly the same as before because u and v belong in E. However, the border can be also to the left of v. If the border crosses the vertex u, ECMB returns u because, after moving the $\mathcal{M}_M$ on K, all the items are in u. This solution has total cost $d(n-k)$. Recalling that the optimal solution has a cost of kd, the ratio would be $\frac{n-k}{k}$, and for large values of n this ratio tends to n. Even worse is the situation where the border is beyond u, on its left. Here, both u and v have to move because they reside in M, but the optimal cost is still kd, so the ratio can be arbitrarily large.

If we perform the ECMB algorithm on the example of Figure 2.2, we have initially to project c and d to the border. So, we would have to consider the projected points $c' = (4, 4)$ and $d' = (6, 4)$. Then, the arithmetic mean of the rows $2, 4, 4, 6, 3$ is 3.8 (as before in GEC), while the arithmetic mean of the columns (considering the projected points) $3, 2, 4, 4, 3$, is 3.2. By rounding, we have that $u_{\hat{C}} = (4, 3)$, which obviously belongs to the Euclidean area. In this case, the total cost is 32.47 and the ratio $\frac{32.47}{31.98} = 1.015$, which is very close to the best solution (only 1.5% more than the optimal cost).

2.3.3 The Approximation GMM Algorithm

The *Global Manhattan Median* (GMM) algorithm solves SDPP by returning the *median* of all the points, pretending the whole grid as a pure Manhattan grid, i.e., $G(R, C, 1)$. In the literature, for the *Manhattan space* the optimal distribution point is called as the *median* [24], and it will be denoted as:

$$u_M = (\tilde{r}_{\mathcal{M}}, \tilde{c}_{\mathcal{M}}), \tag{2.6}$$

where $\tilde{r}_{\mathcal{M}}$ and $\tilde{c}_{\mathcal{M}}$ are the median of the rows and the columns, respectively, of the customers, up to multiplicities, in $\mathcal{M}$. We can use the median u_M as the distribution point of an arbitrary delivery area, regardless of the value of K.

The median u_M can be efficiently computed [36] with time complexity $\mathcal{O}(n)$.

Theorem 1. *The approximation ratio performing the GMM algorithm is* $\sqrt{2}$.

Proof. When the distribution point is set to the optimal point u^*, let $\mathcal{C}_M(u^*)$ and $\mathcal{C}_E(u^*)$ be the total Manhattan and Euclidean cost for the drone, respectively, for

serving u^*. Since $d_E(u,v) \leq d_M(u,v)$, then $\mathcal{C}(u_M) \leq \mathcal{C}_M(u_M)$ and $\mathcal{C}(u^*) \geq \mathcal{C}_E(u^*)$. By the optimality of u_M for a full Manhattan grid, one has $\mathcal{C}_M(u_M) \leq \mathcal{C}_M(u^*)$. Recalling the Cauchy-Schwarz inequality, that is, $a + b \leq \sqrt{2}\sqrt{a^2 + b^2}$, one has $d_E(u,v) \leq d_M(u,v) \leq \sqrt{2} d_E(u,v)$ and therefore:

$$\frac{\mathcal{C}(u_M)}{\mathcal{C}(u^*)} \leq \frac{\mathcal{C}_M(u^*)}{\mathcal{C}_E(u^*)} \leq \sqrt{2}. \qquad \square$$

Observation 3. There is an instance such that the approximation ratio performing GMM tends to $\sqrt{2}$. Suppose to have a configuration like the one illustrated in Figure 2.5.

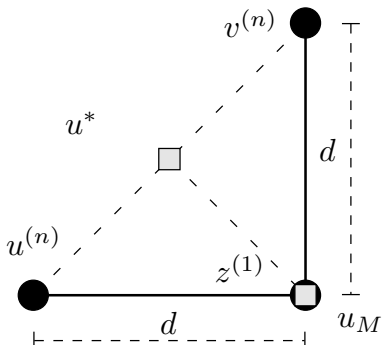


Figure 2.5: A particular instance when invoking GMM.

The three vertices $u = (0,0)$, $v = (d,d)$, and $z = (d,0)$ belong to the Euclidean part E of the EM-grid G. Each of the two vertices u and v have multiplicity n, while the third vertex has a single multiplicity. For large values of n, the distribution point is attracted by the straight line $\overline{uv}$. In fact, if we had only u and v without the point z, by Lemma 2, any point on $\overline{uv}$ would be optimal. However, because of z, only the closest point to z on the line $\overline{uv}$, i.e., $\left(\frac{d}{2}, \frac{d}{2}\right)$, is optimal. Hence, selecting $u^* = (\frac{d}{2}, \frac{d}{2})$ as the distribution point, it holds $\mathcal{C}(u^*) = \frac{d}{2}\sqrt{2} + nd\sqrt{2}$. Instead, GMM returns $u_M = z$ for any value of n and $\mathcal{C}(z) = 2nd$. Thus, for large values of n, $\frac{\mathcal{C}(z)}{\mathcal{C}(u^*)} \to \sqrt{2}$.

If we perform the GMM algorithm on the example of Figure 2.2, we have to individually consider the median of rows and median of columns. The median row among the sorted sequence $2, 3, 4, 4, 6$ is 4, while the median column among the sorted sequence $2, 3, 3, 7, 10$, is 3. Hence, we have that $u_M = (4, 3)$, which belongs in the Euclidean area. Note that incidentally $u_{\hat{C}}$ and u_M coincide. Hence, the total cost is 32.47 and the ratio $\frac{32.47}{31.98} = 1.015$.

2.3.4 The MMEB Algorithm

The *Manhattan Median by projecting E to B* (MMEB) algorithm solves SDPP by applying the same strategy as ECMB, only replacing Euclidean with Manhattan. In other words, MMEB forces the distribution point to reside in the Manhattan area. Given a mission $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_M$, to serve all the customers of $\mathcal{M}_E$ while forcing the distribution point to reside in the Manhattan side, the drone must fly passing through a unique point of the border B (see Eq. (2.1)). To visit all the Euclidean customers of $\mathcal{M}_E$ from the border B in position $v_i = (r_{v_i}, K)$ (with $1 \leq i \leq R$), the drone travels the distance $D_{v_i} = \sum_{u \in \mathcal{M}_E} d_E(u, v_i)$. The total Euclidean distance D_{v_i} depends on the unique junction position v_i for connecting the customers of $\mathcal{M}_E$ to the border. Let $\mathcal{M}'_E(i)$ be the i^{th} set $\mathcal{M}_E$ where each original customer $u = (r_u, c_u)$ in the Euclidean side is replaced by its projection $u' = (r_{v_i}, K)$ on the border, for $1 \leq i \leq R$. One can see that to serve the customers in $\mathcal{M}$, the overall cost is D_{v_i}, plus the cost for serving the customers in $\mathcal{M}'(i) = \mathcal{M}'_E(i) \cup \mathcal{M}_M$ in a pure Manhattan grid. Hence, the MMEB algorithm selects as the distribution point location for the drone the vertex:

$$u_{\hat{M}} = \arg\min_{1 \leq i \leq R} \mathcal{C}\left(\tilde{r}_{\mathcal{M}'(i)}, \tilde{c}_{\mathcal{M}'(i)}\right), \tag{2.7}$$

where $u_{\hat{M}}$ is the Manhattan median of $\mathcal{M}'(i)$, and $\tilde{r}_{\mathcal{M}'(i)}$ and $\tilde{c}_{\mathcal{M}'(i)}$ are the median of the rows and the columns, respectively, of the customers, up to multiplicities, in the i^{th} set $\mathcal{M}'(i)$. In Eq. (2.7), the value of $\tilde{c}_{\mathcal{M}'(i)}$ is the same regardless of the value of i, while $\tilde{r}_{\mathcal{M}'(i)}$ changes accordingly. Note that, $u_{\hat{M}}$ can be computed in $\mathcal{O}(nR)$.

If we perform the MMEB algorithm on the example of Figure 2.2, we have to sequentially project a, b, and c to the border, starting from the first row. So, for each $1 \leq i \leq 6$ we would have to consider the projected points $a' = b' = c' = (i, 4)$. Then, we compute the median in the Manhattan grid (as before in GMM) and compute the total cost. Eventually, we return the best i that minimizes Eq. (2.7). In this example, we have that $u_{\hat{M}} = (4, 4)$, which obviously belongs to the Manhattan area (border). The total cost is 33.30 and the ratio $\frac{33.30}{31.98} = 1.041$.

Observation 4. The MMEB algorithm would return, more efficiently, the global optimal solution for SDPP if it is known that u^* resides in the Manhattan grid, in just $\mathcal{O}(nR)$ time. Namely, instead of testing all the positions of the Manhattan grid (that are $R(C-K)$), MMEB considers only R distinct positions on the border and, for each of them, it computes the median of all the customers assuming the customers in E (i.e., the subset $\mathcal{M}_E$) projected on the border. By this observation, we can decrease the time complexity of the brute-force optimal algorithm OPT to $\mathcal{O}(nRK + nR)$.

2.3.5 The Approximation APX Algorithm

All the presented algorithms have peculiarities that make them suitable for particular settings. For instance, MMEB works well when the delivery area is mostly Manhattan, whereas ECMB works well when the area is mostly Euclidean. Therefore, this suggests a new algorithm, that we call APX, which selects as distribution point the *best* point u_B among the four previous ones, i.e.,

$$u_B = \arg\min \{\mathcal{C}(u_C), \mathcal{C}(u_{\hat{C}}), \mathcal{C}(u_M), \mathcal{C}(u_{\hat{M}})\}.$$

Theorem 2. *The approximation ratio performing the APX algorithm is* $\sqrt{2}$.

Proof. APX inherits the approximation ratio of Theorem 1, and trivially:

$$\frac{\mathcal{C}(u_B)}{\mathcal{C}(u^*)} \leq \frac{\mathcal{C}(u_M)}{\mathcal{C}(u^*)} \leq \frac{\mathcal{C}_M(u^*)}{\mathcal{C}_E(u^*)} \leq \sqrt{2}. \qquad \square$$

For completeness, in the example of Figure 2.2, the APX algorithm would compare the best distribution point among $u_C = (4, 5)$, $u_{\hat{C}} = (4, 3)$, $u_M = (4, 3)$, and $u_{\hat{M}} = (4, 4)$ whose costs are 35.30, 32.47, 32.47, and 33.30, respectively. So, the best distribution point selected by APX is $(4, 3)$.

In Table 2.1, we compare the presented algorithms evaluating their time complexities and guaranteed approximation bounds. Note that we have found two approximated solutions: GMM and APX. The former has time complexity linear in the number of the customers, while the latter has a time complexity that depends also on the size of the EM-grid. As we will see, although we can guarantee the same approximation ratio, the performance of APX is always better than that of GMM.

Table 2.1: Comparison between the algorithms.

Point	Type	Alg.	Time complexity	Apx. ratio
u^*	brute-force[2]	OPT	$\mathcal{O}(nRK)$	1
u_C	centroid	GEC	$\mathcal{O}(n)$	–
$u_{\hat{C}}$	centroid	ECMB	$\mathcal{O}(n)$	–
u_M	median	GMM	$\mathcal{O}(n)$	$\sqrt{2}$
$u_{\hat{M}}$	median	MMEB	$\mathcal{O}(nR)$	–
u_B	comparison	APX	$\mathcal{O}(nR)$	$\sqrt{2}$

[2]By Observation 4.

2.4 Performance Evaluation

In this section, we first numerically evaluate the goodness of our algorithms in EM-grids by varying the number of rows and columns, and the number of customers to be served. Then, with the help of the open air simulator BlueSky, we compare the cost of our best solution APX with the cost of a solution that serves the costumers from a fixed distribution point that emulates the position of the depot of a delivery company.

2.4.1 Settings and Parameters

We implemented our algorithms in Python language version 3.5, and run all the instances on an Intel i7-10genK computer with 16GB of RAM.

In order to evaluate our proposed algorithms for solving SDPP, we set different layouts by varying $R, C \in \{50, 100\}$ and $K \in \{1, \frac{1}{4}C, \frac{1}{2}C, \frac{3}{4}C, C\}$. We uniformly generate $n = |H|$ random positions (in which multiplicities can occur) inside the grid with $n = \{5, 10, 15, 20, 50, 100\}$, and then return the *average* ratio (along with the *standard deviation*) on 33 different random instances. Moreover, given a setting with n random points, we evaluate the algorithms when balancing the quantities n_E and n_M with respect to a certain fraction $p = \{0, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\}$ on n, such that $n_E = p \cdot n$ and $n_M = (1 - p) \cdot n$, with $n = n_E + n_M$.

We compare the algorithms with respect to OPT, and we plot the ratio $\rho = \frac{\mathcal{C}(H,\tilde{u})}{\mathcal{C}(H,u^*)} \geq 1$, where ρ is the ratio between the total cost for serving all the required customers assuming $\tilde{u}$ as distribution point returned by the tested algorithm, and the total cost outputted by the optimal algorithm assuming u^* as optimal distribution point, where $H \subseteq G$.

2.4.2 Numerical Analysis

Figure 2.6 reports the numerical evaluation of all the presented algorithms for solving SDPP with respect to the optimal algorithm OPT. In particular, we show the performance of the algorithms when fixing a layout (R, C, K) and varying the number n of deliveries.

The first row of Figure 2.6 shows a squared layout, the second row a layout with $R > C$, and the third row a layout with $R < C$. Let us now focus on the first row. We initially observe a symmetric behavior between the centroid-based and the median-based algorithms. The *centroid-based* algorithms such as GEC and ECMB work well when the Euclidean side is the largest, i.e, K tends to C, while the *median-based* algorithms such as MMEB and GMM get a better performance when the Manhattan side is the largest, i.e., for smaller values of K. In general, when the number of customers n to be served is small, the algorithms present very

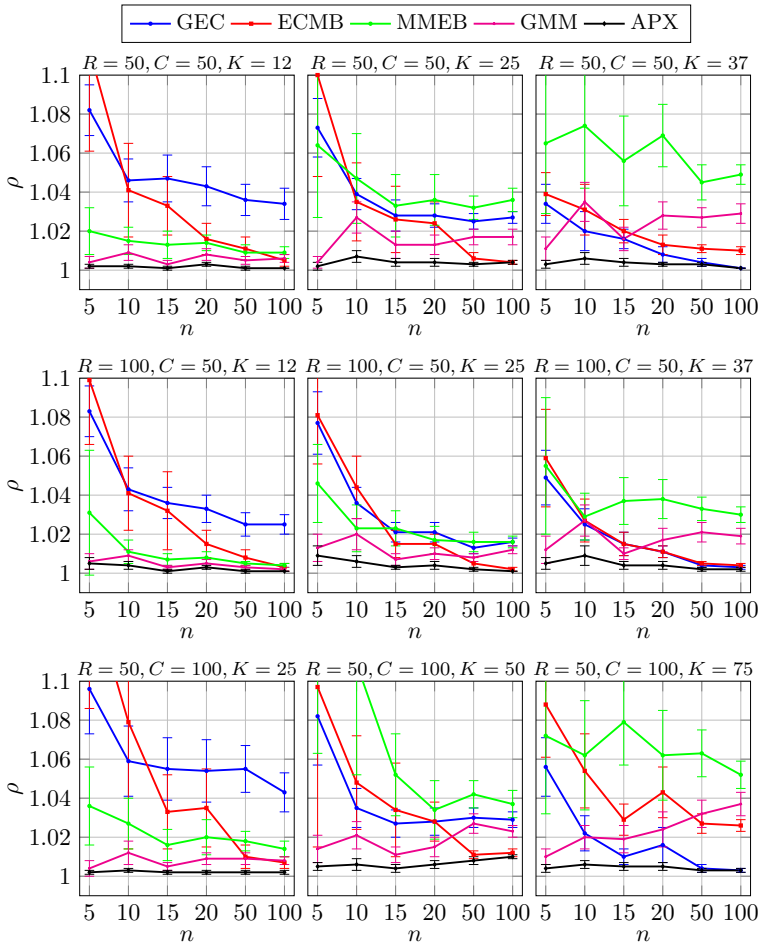


Figure 2.6: All the algorithms on different layouts when varying n.

variable results and hence the standard deviation is pretty large. This means that when n is small, it is easy to have instances for which the same algorithm can work well or very poorly. Nevertheless, APX obtains a good and stable performance regardless of the value of n and of the border K.

The best algorithm is APX whose average ratio is below 1.01, well below the guaranteed upper bound of $\sqrt{2} \approx 1.41$. For almost Manhattan grids, as expected, the worst performing algorithm is GEC, especially when n is large. For almost Euclidean grids, the worst performing algorithm is MMEB, which is worse than GMM. Concerning the second row of Figure 2.6, when $R > C$, all the algorithms obtain a good performance in the case of $K = \frac{C}{2}$. In principle, on this layout the algorithms behave almost the same as in the previous case with $R = C = 50$, but with much better performance. It is also interesting to note that in the opposite layout with $R < C$ (third row of Figure 2.6), the algorithms get a bad performance

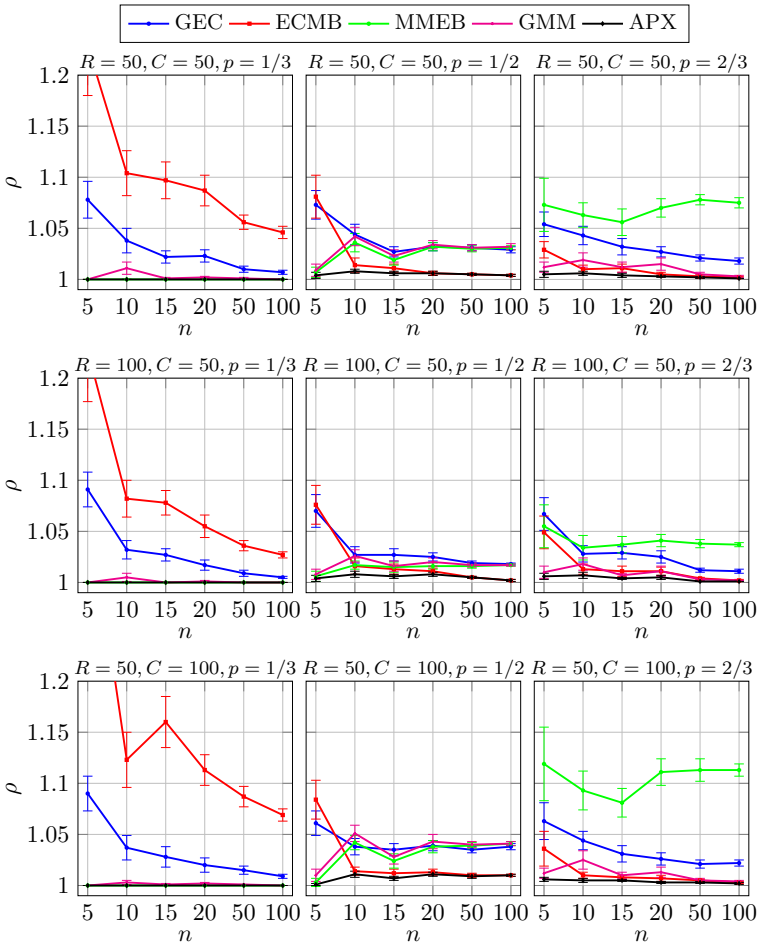


Figure 2.7: All the algorithms on different layouts when varying p.

with respect to OPT. This bad trend can be seen especially with GMM, which has a counter-intuitive behavior. The poor performance when there are more columns than rows is justified by the fact that the impact of the border is higher, and there is a quite large difference between the centroid-based and the median-based algorithms. Indeed, with a small n the performance of GMM is good, but when n starts to increase, its performance degrades immediately. This is accentuated when the grid is almost Euclidean ($K = 75$).

Figure 2.7 compares the performance of the algorithms when fixing a layout $G = (R, C, \frac{C}{2})$ and varying the quantity p of deliveries to each side of the EM-grid. As before, the first row depicts the squared layout, while the second and the third rows show the two different rectangular layouts. On each plot we fixed the value of the border K as $\frac{C}{2}$, i.e., we zoom in the central column of Figure 2.6. In the first column of Figure 2.7, a third of the deliveries are on the Euclidean area, i.e, $p = \frac{1}{3}$,

in the second column the deliveries are equally halved to each sub-area, i.e, $p = \frac{1}{2}$, and finally in the third column a third of the deliveries are on the Manhattan area, i.e, $p = \frac{2}{3}$.

In general, the performance are more variable than those already seen in Figure 2.6, so here we have a different scale on the y-axis for ρ up to 1.2. On these plots, ECMB performs poorly, especially for small values of p. This is due to the fact that ECMB forces the distribution point in the Euclidean area, and having two thirds of the deliveries in the Manhattan area, the total cost for the drone blows up accordingly. It is interesting to see how all the algorithms decently perform when the n deliveries are equally distributed on both the areas. In these cases, the best algorithms are GMM and, obviously, APX. As expected, the centroid-based algorithms work well when most of the points belong to the Euclidean grid, and the median-based algorithms work well when most of the points belong to the Manhattan grid.

In the next section we move toward a much realistic environment, evaluating the performance of APX by using an open air simulator.

2.4.3 Simulation Results

For evaluating our algorithms on a simulated environment, we rely on BlueSky. BlueSky[3] is an open Air Traffic Simulator (ATS), and is meant as a tool to perform research on Air Traffic Management and Air Traffic Flows. It is distributed under the GNU General Public License v3. The goal of BlueSky is to provide a free tool in order to visualize, analyze, or simulate air traffic without any restrictions, licenses, or limitations [23]. Although BlueSky is an open air simulator, we simply refer to it as drone simulator. A screenshot is shown in Figure 2.8.

In the simulated environment, we set $R = C = 50$ and $K = \{12, 25, 37\}$ which represent a grid with a third, a half, and two thirds of Euclidean columns, respectively. Overall, we fix a reasonable number of deliveries $n = 50$. Moreover, we set the speed of the drone to 10m/s and its altitude 100m above the ground. Finally, we set the unitary distance between two grid points as 100m, so the whole area is $5 \times 5\text{km}^2$. Figure 2.8 shows a random instance of BlueSky where the labels in green (with an associated arrow that indicates the drone's heading) represent the drones, while the labels in gray depict the customers' destinations. In the simulations, we evaluate the total drone's covered distance and elapsed time for accomplishing all the deliveries. Namely, the focus of our research is to minimize the mission's cost by selecting a single starting point from which all the deliveries are served. In order to expedite the simulations, we virtually set a drone for each individual deliver, and then we run in parallel on BlueSky all the flights and sum

[3]https://github.com/TUDelft-CNS-ATM/bluesky

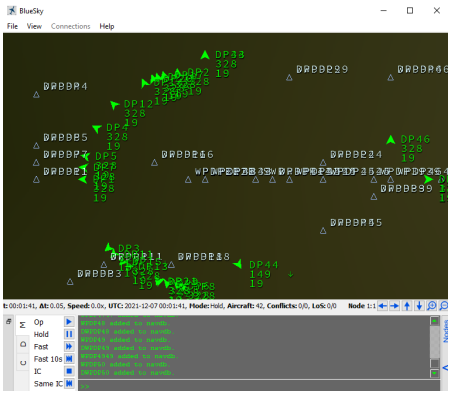


Figure 2.8: A screenshot of the drone simulator BlueSky. The numbers are the labels assigned to the drones and customers.

all the distance and time values.

We also compare our solution that uses a distribution point selected ad-hoc for the current set of deliveries with the solution that uses a fixed, static, predefined distribution point, independent of the current deliveries. A fixed distribution point simply models a fixed warehouse built in a specific location which is the standard (current) solution for a delivery company. As already said, a distribution point that varies with the given set of customers, instead, implies to have a mobile pod that moves the packages to be delivered at a certain position of the grid. The (non fixed) distribution point, that depends on the set of customers, introduces the additional cost of moving the pod, paid just once for all the deliveries of the same set. At the moment, we neglect such a cost because our goal is to see if it is advantageous to use the ad-hoc distribution point over the fixed one in terms of distance covered and time spent by the drone to serve the given set of customers. For this reason, we compare the cost of the distribution point returned by the best algorithm APX presented in this Chapter with OPT, and with the following three fixed points:

- u_E (denoted as FixE) is the center of the Euclidean area, i.e., $u_E = (\lfloor \frac{R}{2} \rfloor, \lfloor \frac{K}{2} \rfloor)$;
- u_B (denoted as FixB) is the center of the border, i.e., $u_B = (\lfloor \frac{R}{2} \rfloor, K)$;
- u_M (denoted as FixM) is the center of the Manhattan area, i.e., $u_M = (\lfloor \frac{R}{2} \rfloor, \lfloor \frac{C+K}{2} \rfloor)$.

In this section, in addition to the distance covered by the drone which was also evaluated with the analytic experiments in Section 2.4, we evaluate the time spent by the drone during the missions. First, Figure 2.9 illustrates the qualitative assessment of our new solution by plotting the ratio between the distance and time

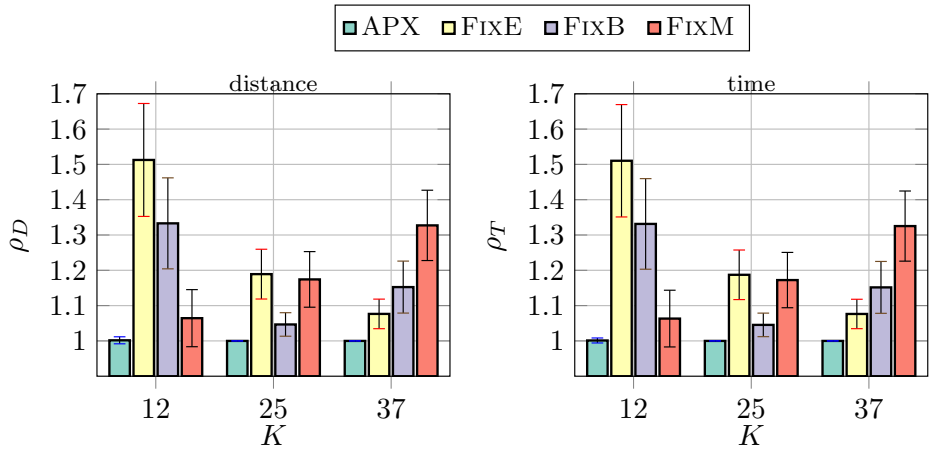


Figure 2.9: Results by using BlueSky simulator comparing the distance and time with respect to the optimal one.

cost of the APX and of the fixed points FixB, FixB, and FixM with respect to that of OPT. Then, Figure 2.10 gives the absolute distance and time values to quantitatively assess our new solution.

Figure 2.9 plots in the y-axis the ratio $\rho_D = \frac{\mathcal{C}(H,\tilde{u})}{\mathcal{C}(H,u^*)} \geq 1$ (on the left) and the ratio $\rho_T = \frac{\mathcal{T}(H,\tilde{u})}{\mathcal{T}(H,u^*)} \geq 1$ (on the right) where $\mathcal{T}$ represents the time required for performing all the round trips from the distribution point to the customers. Basically, on the 33 random simulated scenarios with BlueSky, APX and OPT almost always coincide on both distance and time. The ratio is thus very close (slightly better) to that of the analytic experiments in Figure 2.6 (first row, $n = 50$). Hence, fixed the source, destination, and intermediate points, in absence of external agents (e.g., wind[4]), the drones are routed in BlueSky along straight lines, without deviations and noises, as we assumed in Section 2.4.2. The differences between ρ_D and ρ_T are pretty negligible. As one can now evaluate in Figure 2.10, the time obeys the physical law, that is, it is equal to the distance divided by the speed. Hence, without external wind, the drones in BlueSky constantly move at the specified speed. Although this aspect was not so obvious at the beginning when we started to consider this simulator, it is quite important because we can now reasonably assume that not only the distance but also the time can be analytically estimated.

The main aim of Figure 2.9 is to evaluate the performance of the mobile distribution points versus that of the fixed distribution points. As expected, when K is small FixM is preferable because the Manhattan side dominates in EM-grid,

[4]BlueSky allows to set a global wind speed and direction parameter when simulating the missions, influencing so the flight of drones.

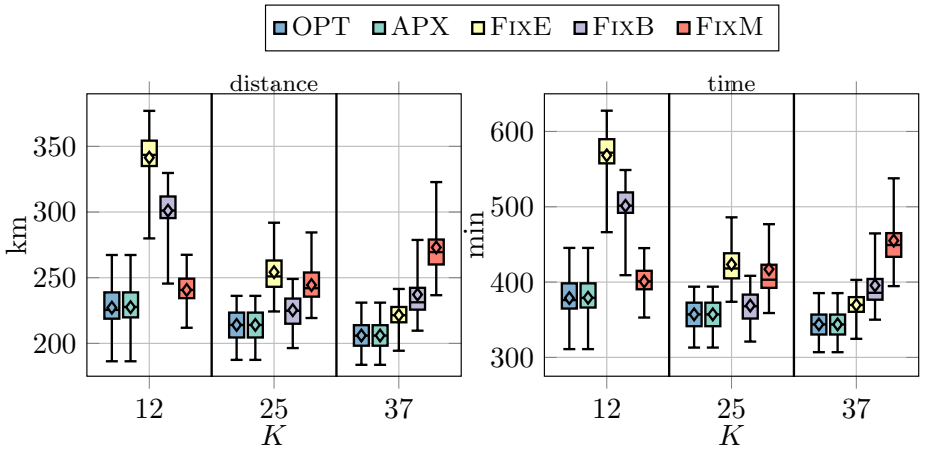


Figure 2.10: Results by using BlueSky simulator evaluating the distribution (distance and time) of all the deliveries.

and when K is large FixE is preferable because the Euclidean side dominates in EM-grid. In both cases, FixB is in between. When $K = \frac{C}{2}$, the worst result occurs when apply FixE or FixM (middle of Euclidean or Manhattan grid, respectively) whose distribution points are $u_E = (25, 12)$ and $u_M = (25, 37)$, respectively, while the other fixed point FixB with $u_B = (25, 25)$ (middle of border) gets a quite good performance, just a bit worse than the optimal. This is reasonably expected since u_E and u_M are specular each other with respect to the border that halves the EM-grid, and both are equidistant from the border. The point u_B is closer to the optimal point u^* than to the other two points, and therefore the cost when applying FixE or FixM is larger.

Finally, in Figure 2.10 we plot the simulated results in a much more thorough manner. In these plots we make use of *box plots* [37] which report the *minimum*, the *first quartile*, the *median*, the *third quartile*, and the *maximum*. Also the *mean* is reported as the diamond. In particular, in the first plot we highlight the total covered distance by the drone (kilometers, km), while in the second plot we show the total makespan of the drone for performing all the deliveries (minutes, min).

We initially observe that the optimal solution OPT requires approximately 210-230km as total distance flown for 50 deliveries. Therefore, each delivery requires an average of 4.2-4.6km for a complete round trip to/from the distribution point. Obviously, the total covered distance increases when the value of K decreases due to the fact that the drone has to connect more intermediate points provided by the Manhattan metric. Concerning APX, its performance almost matches the one of OPT on these random instances. It is interesting to note that the mean and the median coincide for APX and OPT.

The three static points show the same behavior as before (see Figure 2.9), i.e.,

FixE is better when K is large, while FixM is better when K is small. Moreover, note that the mean of FixE is smaller than its median when $K = 12$. Similarly, the mean of FixM is larger than its median when $K = 37$. The span of values of FixE is larger than that of FixM.

The worst performance can be observed for FixE when $K = 12$ in which the drone has to fly for almost 350km (7km for a single delivery). The overall duration of the mission is about 600min, i.e., about 10hours (12min for a single delivery). With respect to the worst case, the length of the missions in APX save up to 100km and 150min. Finally, in OPT and APX the drone flies for approximately 350min (7min for a single delivery), thus halving the delivery time with respect to the worst scenario. As expected, the missions in OPT are shorter when the Euclidean side dominates in EM-grid, and longer when the Manhattan side dominates in EM-grid. The OPT and APX missions saves from a minimum of 10 to a maximum of 150km, and from 10 to 150min with respect to the fixed distribution points.

In conclusion, FixM (respectively, FixE) has a reasonable performance if it is applied when the Euclidean side is small (respectively, large), while its performance is very poor if it is applied when the Euclidean side is large (respectively, small). That is, the behavior of FixM and FixE strongly depends on which side dominates in EM-grid. Therefore, both FixM and FixE can perform very poorly depending on the distribution of customers among both the sides. So, when the width of the Manhattan and Euclidean sides is not defined a priori, APX substantially beats all the other algorithms.

Concluding, in this Chapter, we considered a drone which is in charge of delivering small packages to customers in a delivery area, modeled as an EM-grid. We proposed approximation and heuristic algorithms for computing the distribution point that solves SDPP. Although we can guarantee the same approximation ratio for the two approximation algorithms, the performance of APX is always better than that of GMM. By using the BlueSky simulator, we compare our best solution APX with fixed distribution points, selected as a function of the size of the EM-grid. Our solution saves from a minimum of 10 to a maximum of 100km of traveled distance, and from 10 to 150min of elapsed time with respect to the fixed distribution points.

As a future work, we would like first to extend our solution to multiple drones. This requires to partition the customers among the drones, and then apply our proposed solutions to each partition. Also, we would like to further investigate the symbiotic cooperation between a truck, pods, and drones. In this case, we could set different distribution points for subsets of customers, and then use a truck that sequentially visits each of these. Other possible extensions would include synchronization issues, in terms of time, on a multi-drone scenario.

Chapter 3

Winds make the difference in drone delivery[1]

Deliveries with drones are potentially faster than with trucks because drones can connect points in the plane by traversing straight lines, without traffic or other impediments. However, delivery companies must be aware that the drone ranges must be short because the drones are battery-powered and the battery is limited in capacity due to the drone size. Also, strong winds can raise concerns during drone deliveries.

While the limitations due to strong gusts have been a primary concern so far for planning drone missions [11], to the best of our knowledge, not much attention has been given to the possibility of exploiting favorable wind conditions [25] to reduce the energy consumption of the drone. In this Chapter, we explore this possibility. Islands, like Corse, are very windy and favorable winds can be a great opportunity. Precisely, we propose to select the drone's trajectory for the last leg of the delivery route so that the wind is as favorable as possible. However, favorable wind over a too long route can make the energy saving for unit distance futile. So, the wind cannot be selected independently of the length of the traversed distance.

Due to the above considerations, our solution proposes to use a truck that brings the drone close to the delivery point. Then, pondered both the direction and the distance, we fly the drone along the trajectory that uses the least amount of energy for serving the delivery point. Using the drone, the truck avoids the detour from the main route up to the delivery point. Using the truck, the drone can operate in a short range. In a moderately dense ground environment, like a residential area neighborhood, already at a low altitude, the drone can freely move

[1]This work has been published to DCOSS 2021 - the 18th Annual International Conference on Distributed Computing in Sensor Systems and an extension as been submitted to IEEE T-ITS - Transactions on Intelligent Transportation Systems journal ⟶ Publications n. [3] and n. [4]

in the air having no obstacles on its way. Consequently, we assume that the drone serves the delivery point just following two straight lines from the take-off to the delivery point and from the delivery point to the landing point. Our goal is then to determine the take-off and landing points on the truck's route that minimize the battery consumption exploiting the most favorable wind conditions.

3.1 The related work

In the last five years, the problem of delivering goods with drones has been approached by several papers [19, 20]: an excellent and concise summary of the state of art is reported in [38]. Almost all the papers assume that the drone has unit-capacity and has to return to the warehouse after every single delivery due to the payload and energy budget constraints. Many solutions in the literature consider drones working in tandem with a mobile ground device (truck, van) and there are many ways the two means of transportation collaborate. Some solutions partition the deliveries between the two, some make them cooperate in every single mission [38, 39, 40] In our solution, truck and drone cooperate in every mission. From our point of view, the truck is an autonomous vehicle, not limited in energy, moving on a predefined route, with no possibility to turn around and to detour from its route. It is at drone's disposal and brings the drone close to the delivery point. The drone completes the last leg of the delivery and is limited in energy budget. The truck and the drone aim to serve the end customer minimizing the energy spent by the drone. Although selecting the shortest trajectory between the truck's route and the delivery point could seem the winning strategy, this is not always the solution that minimize energy as we show in this Chapter. In a windy day, the drone can save energy if it serves the delivery point connecting the truck's route and the delivery point not with the oblique lines. Such oblique drone trajectory is also used in [39, 40, 41], but there the drone's route is given in input to the problem. In this Chapter, instead, the drone's trajectory is the output of our algorithm.

To the best of our knowledge, the impact of favorable winds on the planning of the drone trajectory has not received much attention so far. We believe that one of the reasons that have held back the researchers from studying such an impact is the difficulty of having simple and reasonable models that describe the drone energy consumption in terms of wind intensity and direction change. Recently, in [18], an energy model that permits to tabulate the expected energy consumed given the sort of drone used (quadcopter, octocopter), the speed of the drone, the payload, and the relative wind condition has been developed. From this model springs a line of research that considers wind conditions. In [42], the problem of finding which is the percentage of flight missions that can be successful accom-

plished with a given energy budget knowing the wind conditions on-the-fly was studied. The delivery problem has been modeled as finding the shortest path on a time-dependent weighted graph. The edge weights represent the energy spent to traverse the edges and the weights are time-dependent because the wind conditions vary during the time. However, in [42], the drone moves along predefined routes represented by the edges of the graph. In this Chapter, instead, we exploit the freedom of selecting the drone trajectory that has the wind in favor to improve the energy efficiency. We found that, in presence of wind, often a longer distance is less energy demanding than the shortest one.

Contribution: Our results are summarized as follows:

- We calculate the relative wind on any straight line that passes through the delivery point by building a coordinate system whose origin is the delivery point itself and the x-axis is parallel to the truck's route;

- We propose a constant time algorithm (OSR) to plan the *optimal service route* when the truck moves on a line given the desired drone speed and the wind conditions;

- We propose the algorithm *Multi-Side* OSR (MS-OSR), to plan a minimum-energy service route when the truck moves on a multi-line route that bounds the delivery area;

- We simulate OSR and MS-OSR under two different set of winds: the exhaustive winds, which average the consumption on all the possible winds of the compass rose, and the real winds collected in few locations in Corse. We found that, OSR and MS-OSR save for each delivery, respectively, the 30% and 60% of energy with respect to the shortest drone's trajectory that connects the delivery point and the truck route. Recall that the shortest drone's trajectory coincides with the Euclidean distance between the delivery point and its projection on the truck's route.

Organization: The rest of the Chapter is organized as follows. Section 3.2 defines the relative winds and the compass rose, surveys the energy model, and computes the levels of energy associated with the winds of the compass rose. Section 3.3 proposes the OSR algorithm that determines the take-off and landing points of the drone on the truck's route that minimizes the energy spent to serve the end customer. Section 3.4 extends OSR to the multi-line scenario. Then the performance of OSR and the MS-OSR extension are evaluated.

3.2 Model

In this section, we describe the main concepts we use to model the problem. First, we explain the wind system in Section 3.2.1, then the energy model and the energy levels associated with the compass rose in Section 3.2.2, and finally the truck-drone collaborative system in Section 3.2.3.

3.2.1 The Wind

Wind is an important variable for drone energy consumption. The weather stations record the meteorologic wind which, during the day, can change speed and direction. The energy consumption increases if the drone movement and the wind are opposite, while decreases if the drone movement and the wind are equal. To evaluate the relative wind direction for the drone movement, we must convert the meteorological wind direction into the mathematical wind direction, as explained below.

Meteorologic and Mathematical Wind Directions: Weather stations record the *meteorologic direction* of the wind, that is, the direction from which the wind originates, assuming North as the 0 degree direction. For instance, if the wind blows from the north to the south, the weather station records a direction of 0 degrees, while if the wind blows from the east to the west, the weather station refers to 90 degrees. The meteorologic direction of the wind grows clockwise in a Cartesian coordinate system xOy whose x-axis is the North direction and the origin O is the destination (and not the source) of the wind. The *mathematical direction* of the wind, instead, considers the origin O as the source of the wind. The mathematical wind directions grow counterclockwise from the usual x-axis which is equal to the East direction. Hence, the conversion rule from the meteorological direction of the wind to the mathematical one is [43]:

$$w_d^{\text{ma}} = \big|\underbrace{|-w_d^{\text{me}}+360|_{360}}_{clockwise}\ \underbrace{+90}_{phase}\ \underbrace{-180}_{O}\big|_{360} = |-w_d^{\text{me}}+270|_{360}$$

Fig. 3.1 shows the direction of the meteorological wind $w_d^{\text{me}} = 225$ that is directed towards O and the corresponding mathematical direction $w_d^{\text{ma}} = 45$ that has O as the source.

From now on, in this Chapter, we only refer to the mathematical directions of the wind. All the directions of the wind collected by the weather stations in Corsica in Section 3.4.2 are converted in mathematical winds before applying the OSR and MS-OSR algorithms.

Global Wind and Relative Wind: The *global wind* ω is the wind that occurs in the area where the drone operates. The global wind $\omega = (\omega_d, \omega_s)$ is represented by the mathematical *direction* ω_d and its *speed* ω_s. The *relative wind* is the wind that the drone faces when it moves on a line r with direction γ_r under a global wind ω. The relative wind has *mathematical direction* $\varphi(\gamma_r, \omega)$ and speed φ_s. The relative direction $\varphi(\gamma_r, \omega) = |\omega_d - \gamma_r|_{360}$ is the difference mod 360 between the direction of the global wind and the direction of the drone movement (Fig. 3.2). In this Chapterwe assume $\varphi_s = \omega_s$, although in practice the flight altitude and the kind of the ground environment (i.e., urban, suburban, rural) can modify ω_s making $\varphi_s \geq \omega_s$. From now on, since we do not consider that the wind changes during a drone mission, while we consider different drone trajectories, we denote the relative direction $\varphi(\gamma_r, \omega)$ of the relative wind as $\varphi(r)$.

The Compass Rose: To define the *sector* winds and the *compass rose*, we divide the turn angle at O of a conventional Cartesian coordinate system xOy into $4t$ sectors, each of width $\sigma = 180/2t$ degrees. Conventionally, the *sector wind* S_i will contain the winds whose relative wind direction verifies

$$\left(i\frac{180}{2t}\ ;\ (i+1)\frac{180}{2t}\right] \quad 0 \leq i \leq 4t-1 \tag{3.1}$$

A *representative direction* ρ_i, with $0 \leq i \leq 4t-1$, will be associated to each sector wind S_i. The representative ρ_i will be used to compute the energy consumption of any relative wind direction that falls in S_i. In substance, the compass rose is used to limit to $4t$ the number of the possible energy levels, as we will see in the next section.

3.2.2 Energy Model and Levels of Energy

Recently in [18], a model to evaluate the energy $\mu(\upsilon_d, \omega_s, \varphi(r), \kappa)$ depleted to keep a drone travelling for 1m, at a constant ground speed υ_d, along a trajectory r in direction γ_r, carrying a payload κ, and under a global wind ω was proposed. The energy is a function of the power Π required for producing the needed thrust to let the drone fly. Π depends on the drone's rotor characteristics, and on the square of the drone's relative air speed $\upsilon_a(r) = |\sqrt{\upsilon_d^2 + \omega_s^2 - 2\upsilon_d\omega_s\cos(\varphi(r))}|$. As reported in [42,44], since Π is calculated by solving an implicit equation, we cannot give a close formula for the power and for the energy as well. However, the energy can be tabulated for different drone and wind speeds, different payloads, and for

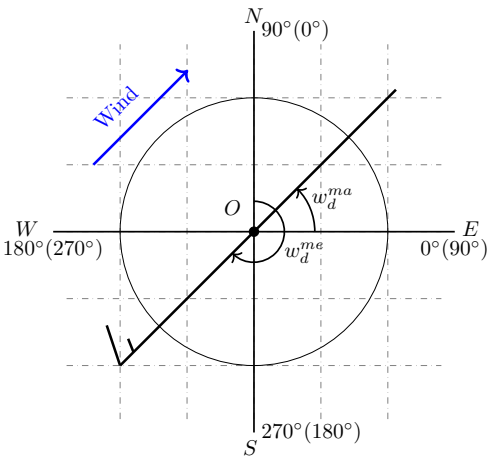


Figure 3.1: The x and y axes are labelled with the cardinal, mathematical, and meteorologic (in brackets) directions.

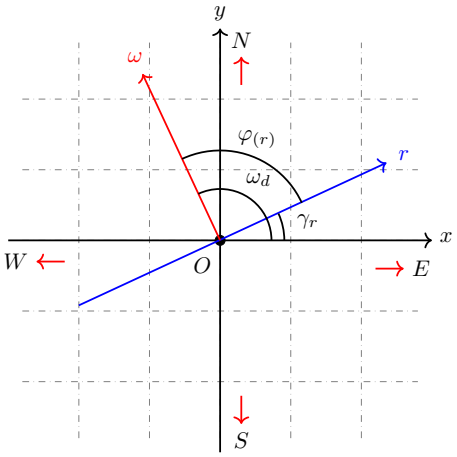


Figure 3.2: Relative wind direction $\varphi(r) = 90°$ on the line r when $\gamma_r = 25°$ and $\omega_d = 115°$.

the $4t$ different relative winds of the compass rose.

In this Chapter, we will use energy values tabulated under different conditions. When v_d, ω_s, and κ are clear from the context, we simply denote the energy spent under the sector wind S_i as μ_i, with $0 \leq i \leq 4t - 1$. Table 3.1 reports the 12 different energy levels of a compass rose with $t = 3$, and thus $\sigma = 30°$. For computing the energy consumption, we select as ρ_i the angle in S_i whose cosine is maximum in absolute value (e.g., $\rho_0 = 1°$ in S_0 and $\rho_{11} = 0°$ in S_{11}. With this representative, μ_0 and μ_{11} are almost the same). When the cosine of the representative wind is negative (see for example S_3 with respect S_2), μ_i increases because $v_a(r)$ increases. Table 3.1 also reports the different energy levels below different payloads.

Finally, the energy spent by a drone to traverse a given distance δ facing the sector wind S_i is given by the product of $\mu_i\delta$, with $0 \leq i \leq 4t - 1$.

Table 3.1: Energy coefficients μ_i with different payloads κ and $v_d = 10$m/s, $\omega_s = 10$m/s.

μ_i	Payload κ (Kg)		
S_i	0	2	6
0	0.1235395549	0.1511641188	0.2126169706
1	0.1487165204	0.1770355558	0.2398070629
2	0.2216461515	0.2511850333	0.316473693
3	0.4462761438	0.4787047231	0.5491670144
4	0.534990835	0.5692753882	0.6425861937
5	0.5677098232	0.6027887907	0.6773260158
6	0.5676725759	0.6027506094	0.6772863865
7	0.5328310777	0.567065102	0.640298223
8	0.4426247141	0.4749871414	0.5453470678
9	0.2184914376	0.2479916775	0.3131969116
10	0.1471049048	0.1753853793	0.2380818085
11	0.1235135924	0.1511373263	0.21258864

3.2.3 The OSR problem

In this Chapter, the drone D is carried by a truck, named MT. When a delivery point P that has to be served by the drone approaches, the drone selects on the truck path the take-off and landing points, called X_T and X_L, respectively. The drone flies on the straight line from X_T to P carrying the payload. After dropping off the payload, the drone empty moves back on the straight line from P to X_L, as illustrated in Fig. 3.3. The drone and the truck always move forward in the sense that the projection H of P onto the MT path must follow X_T and precede X_L. As said, the MT never turns back to pick up the drone. Changing X_T and X_L, the slope of the segments X_TP and PX_L changes and the relative winds $\varphi(X_TP)$ and $\varphi(PX_L)$ change too. Also, the lengths of $\overline{X_TP}$ and $\overline{PX_L}$ change. In this Chapter, we want to find the *optimal service route* X_TPX_L, that is, the last mile trajectory that minimizes the energy consumed by the drone to serve P.

3.3 The optimal service route algorithm

In this section, we assume to have in input the MT's route represented by a straight line r and the truck's movement direction γ_r, a delivery point P, and a global wind ω. Our goal is to find the *optimal service route* OSR that minimizes the energy spent by the drone to fly towards P and return back on the MT's path. OSR is uniquely identified by the take-off point T and the landing point L on r.

We reason as follows. On r there are many points from which the drone can take off and land. Fixed the global wind in the area, the drone faces a relative wind denoted as $\varphi(\overrightarrow{X_TP})$ and $\varphi(\overrightarrow{PX_L})$, respectively, for each take-off or landing

trajectory. In principle, there are infinite relative winds (one for each take-off X_T or landing X_L point on r), but since we have grouped the relative wind directions, there are at most $4t$ sector winds to be considered. Each sector wind requires a different energy level. Among all the X_T's or X_L's points that fall into the same sector wind and thus require the same energy for unit distance, the one that consumes less energy is the closest to P. Among the $4t$ different shortest take-off and landing trajectories associate to the $4t$ sector winds, OSR selects the take-off and landing trajectories that consume the minimum energy to serve P. In the rest of this section, we implement this reasoning. We first show how to compute the relative wind directions on the possible D's routes, then we determine the sector winds corresponding to them, and then we find the best route for each sector wind. Finally, we summarize the procedure to compute OSR in Alg. 1.

3.3.1 Relative Wind Determination

From now on, let $r = mx + q$ be the MT's route. Consider the Cartesian coordinate system xPy of the mathematical wind, with origin in P. We distinguish two cases [2]:

1. P is on the left of MT when it moves along r, or
2. P is on the right of MT when it moves along r.

Let H be the projection of P on r. We assume that any *candidate take-off point* $X_T \in r$ is on the left of H, i.e., $X_T \leq H$. Similarly, any *candidate landing point* X_L is on the right of H, i.e., $X_L \geq H$.

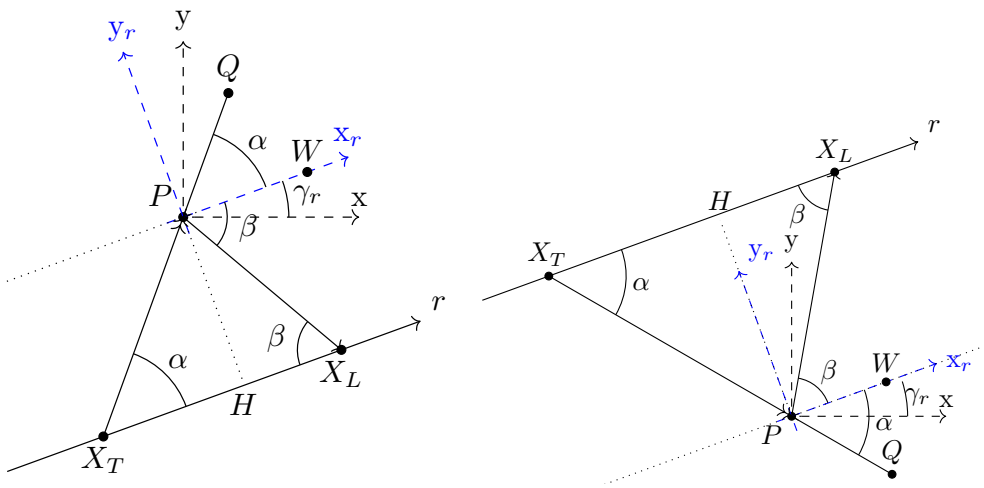
Theorem 3. *The relative winds on* $\overrightarrow{X_T P}$ *and* $\overrightarrow{PX_L}$ *are:*

$$\varphi(\overrightarrow{X_T P}) = \begin{cases} |\omega_d - (\gamma_r + \alpha(HX_T P))|_{360} & \text{if } P \text{ on the left of } r \\ |\omega_d - (\gamma_r - \alpha(HX_T P))|_{360} & \text{if } P \text{ on the right of } r \end{cases} \tag{3.2}$$

$$\varphi(\overrightarrow{PX_L}) = \begin{cases} |\omega_d - (\gamma_r - \alpha(PX_L H))|_{360} & \text{if } P \text{ on the left of } r \\ |\omega_d - (\gamma_r + \alpha(PX_L H))|_{360} & \text{if } P \text{ on the right of } r \end{cases}$$

Proof. We prove (3.2) when P is on the left of r (see Figure 3.3(a)). Consider the rotate Cartesian system $x_r P y_r$ with origin in P, x_r-axis parallel to r and oriented in the same direction as r. Recall that the y_r-axis forms a 90° counter-clockwise angle with x_r. Note that the angle between x and x_r has width γ_r. In Figure 3.3(a), for clarity, we denote $\alpha(HX_T P)$ simply as α and $\alpha(PX_L H)$ as β. Fixed the position

[2]The position of P with respect to r can be easily checked in constant time by testing if r moves counterclockwise or clockwise to reach P.



(a) P is on the left of r. (b) P is on the right of r.

Figure 3.3: The relative winds on X_TP and PX_L.

of X_T, we can calculate $\alpha = \arctan \frac{\overline{PH}}{\overline{X_TH}}$. Observed that r and x_r are parallel, $\overrightarrow{X_TP}$ forms the same angle $\alpha = \widehat{WPQ}$ on x_rPy_r and the angle $(\gamma_r + \alpha)$ on xPy. Hence, the direction of the relative wind on xPy is $\varphi(\overrightarrow{X_TP}) = \omega_d - (\gamma_r + \alpha(HX_TP))$.

Fixed the position X_L on r, it holds $\beta = \arctan \frac{\overline{PH}}{\overline{HX_L}}$. Observed that r and x_r are parallel, the returning path PX_L forms the angle $-\beta$ on x_rPy_r. Hence, the direction of the relative wind on xPy is $\varphi(\overrightarrow{PX_L}) = \omega_d - (\gamma_r - \beta) = \omega_d - (\gamma_r - \alpha(PX_LH))$.

The case where P is on the right of r is depicted in Figure 3.3(b). Eq. 3.2 can be proved similarly to the left case. □

3.3.2 Grouping Relative Winds

We study in which sector wind the relative winds of the D's trajectories fall. First, observe that the relative wind on the MT's route is $\varphi(r) = |\omega_d - \gamma_r|_{360}$, and, fixed the compass rose cardinality t, let S_τ where $\tau = \left\lfloor \frac{|\varphi(r)-1|_{360}}{\sigma} \right\rfloor$ be the relative sector wind on the MT's route. Note that τ depends on $\varphi(r) - 1$ instead of $\varphi(r)$ because the sectors in (3.1) are defined open on the left extreme and close on the right extreme.

Denote the angle formed by the D's route with the MT's route as $\widehat{D}$ and its width as $\alpha(D)$. When X_T and X_L move on r until H (see Fig. 3.3), it is easy to see that $\alpha(D)$ vary in the interval $(0, 90]$. Precisely, $\alpha(HX_TP) = \alpha(PX_LH) = 90$

Algorithm 1: OSR

INPUT: r - MT's route; ω - global wind; P - delivery point;
t - cardinality of sector winds;
$\mu_0^{\kappa}, \mu_1^{\kappa}, \ldots, \mu_{4t-1}^{\kappa}$ - precomputed energy levels with payload; $\mu_0, \mu_1, \ldots, \mu_{4t-1}$ - precomputed energy levels without payload
OUTPUT: T and L on r, along with the energy $E(TP)$ and $E(PL)$
begin;
$\varphi(r) = |\omega_d - \gamma_r|_{360}$;
$\tau = \left\lfloor \frac{|\varphi(r)-1|_{360}}{\sigma} \right\rfloor$;
$\overline{HP}$ is the distance of P from r;
if *P is on the left of r* **then**
    ▷ `Find` T `and` $E(TP)$;
    $\varphi(X_T P)$ scans the IV Q ; ▷ see Eq. 3.3
    Let $j^* = \arg\{\min_{0 \le j \le t}\{\frac{\mu^{\kappa}_{|\tau-j|_{4t}}}{\sin(\Lambda(|\tau-j|_{4t}))}\}\}$; ▷ Tab. 3.2
    Set T : $\overline{TH} = \overline{PH}\cot(\Lambda(|\tau - j^*|_{4t})$;
    $E(TP) = \left(\mu^{\kappa}_{|\tau-j^*|_{4t}}\right) \frac{\overline{PH}}{\sin(\Lambda(|\tau-j^*|_{4t}))}$;
    ▷ `Find` L `and` $E(PL)$;
    $\varphi(PX_L)$ scans the I Q ; ▷ see Eq. 3.3
    Let $j^* = \arg\{\min_{0 \le j \le t}\{\frac{\mu_{|\tau+j|_{4t}}}{\sin(\Lambda(|\tau+j|_{4t}))}\}\}$; ▷ Tab. 3.2
    Set L : $\overline{HL} = \overline{HP}\cot(\Lambda(|\tau + j^*|_{4t})$ $E(PL) = \left(\mu_{|\tau-j^*|_{4t}}\right) \frac{\overline{PH}}{\sin(\Lambda(|\tau-j^*|_{4t}))}$;
if *P is on the right of r* **then**
    ▷ `Find` T `and` $E(TP)$;
    $\varphi(X_T P)$ scans the I Q ; ▷ see Eq. 3.3
    Let $j^* = \arg\{\min_{0 \le j \le t}\{\frac{\mu^{\kappa}_{|\tau+j|_{4t}}}{\sin(\Lambda(|\tau+j|_{4t}))}\}\}$; ▷ Tab. 3.2
    Set T : $\overline{TH} = \overline{PH}\cot(\Lambda(|\tau + j^*|_{4t})$;
    $E(TP) = \left(\mu^{\kappa}_{|\tau+j^*|_{4t}}\right) \frac{\overline{PH}}{\sin(\Lambda(|\tau+j^*|_{4t}))}$;
    ▷ `Find` L `and` $E(PL)$;
    $\varphi(PX_L)$ scans the IV Q ; ▷ see Eq. 3.3
    Let $j^* = \arg\{\min_{0 \le j \le t}\{\frac{\mu_{|\tau-j|_{4t}}}{\sin((\Lambda(|\tau-j|_{4t}))}\}\}$; ▷ Tab. 3.2
    Set L : $\overline{HL} = \overline{PH}\cot(\Lambda(|\tau - j^*|_{4t})$;
    $E(PL) = \left(\mu_{|\tau-j^*|_{4t}}\right) \frac{\overline{PH}}{\sin(\Lambda(|\tau-j^*|_{4t}))}$;
    end
return *T and L*

when $X_T H = H X_L = 0$, while $\alpha(HX_T P) = \alpha(PX_L H) = 0$ when $X_T H = H X_L \to \infty$. Then, the relative winds on the take-off and landing D's trajectories, scan the first quadrant (I Q) or the fourth quadrant (IV Q) of the Cartesian coordinate system with origin in P and whose x-axis coincides with the relative wind $\varphi(r)$ (illustrated in blue in Fig. 3.4). From that, (3.2) can be rewritten as:

$$\varphi(\overrightarrow{X_T P}) = \begin{cases} |\varphi(r) - \alpha(HX_T P))|_{360} & \text{if } P \text{ on the left of } r \quad IVQ \\ |\varphi(r) + \alpha(HX_T P))|_{360} & \text{if } P \text{ on the right of } r \quad IQ \end{cases} \tag{3.3}$$

$$\varphi(\overrightarrow{PX_L}) = \begin{cases} |\varphi(r) + \alpha(PX_L H))|_{360} & \text{if } P \text{ on the left of } r \quad IQ \\ |\varphi(r) - \alpha(PX_L H))|_{360} & \text{if } P \text{ on the right of } r \quad IVQ \end{cases}$$

By (3.3) and (3.1), it holds:

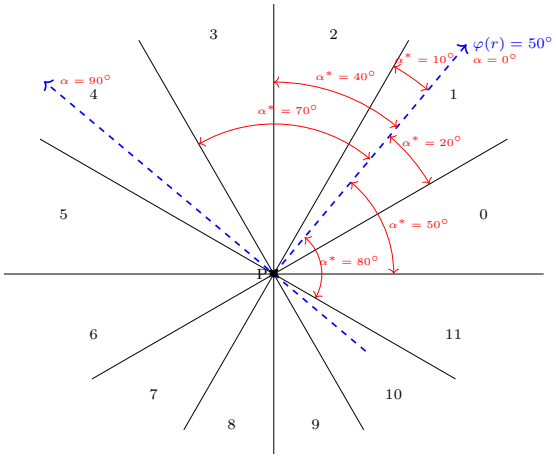


Figure 3.4: The winds of the compass rose scanned by the different drone trajectories $X_T P$ and $P X_L$ when $\varphi(r) = 50°$.

Theorem 4. *When $\varphi(\overrightarrow{X_T P})$ or $\varphi(\overrightarrow{P X_L})$ scan the first quadrant (I Q in (3.3)), the drone faces the winds S_i of the compass rose whose indices i are:*

$$\tau \leq i \leq |\tau + t|_{4t} \tag{3.4}$$

When $\varphi(\overrightarrow{X_T P})$ or $\varphi(\overrightarrow{P X_L})$ scan the fourth quadrant (IV Q in (3.3)), the indices of the scanned winds S_i are :

$$|\tau - t|_{4t} \leq i \leq \tau \tag{3.5}$$

As an example, in Fig. 3.4, the sector winds scanned by the D's trajectories with $t = 3$, $\sigma = 30°$ and $\varphi(r) = 50°$ are S_1, S_2, S_3, and S_4 in the first quadrant, and S_1, S_0, S_{11}, and S_{10} in the fourth quadrant.

To complete our analysis, it remains to find for each sector wind scanned by the D's routes, the shortest take-off and landing paths. Since the length of the D's route is $\frac{\overline{PH}}{|\sin(\alpha(D))|}$ and since $\alpha(D) \in (0, 90]$, the shortest take-off and landing D's routes in each S_i correspond to the largest values $\alpha(D)$.

For the $t + 1$ sector winds scanned by the D's trajectories, Table 3.2 lists the smallest $\xi(i)$ and the largest $\Lambda(i)$ values that fall in S_i. $\Lambda(i)$ gives the shortest route in S_i. The sector wind S_{i^*} that requires the minimum energy to reach P, that is $i^* = \arg\min\{\frac{\mu_i}{\sin(\Lambda(i))}\}$, is the most favorable sector wind to be selected. Namely, the energy in S_i to reach P is given by the distance $\frac{PH}{\sin(\Lambda(i))}$ multiplied by μ_i.

Last, note that, since the drone flies towards P loaded and returns unloaded towards the MT, two different levels of energy, one including the payload μ_i^{κ} and

Table 3.2: The angles $\alpha(D) \in (0, 90]^\circ$ associated to each wind $S_{|\tau+j|_{4t}}$ of the I and IV quadrant of $\varphi(r)$

I Quadrant			
$\alpha(D)$			
Smallest $\xi(\lvert\tau + j\rvert_{4t})$	**Largest** $\Lambda(\lvert\tau + j\rvert_{4t})$	**Wind**	j
1	$(\sigma - \lvert\varphi(r)\rvert_\sigma)$	S_τ	0
$-\lvert\varphi(r)\rvert_\sigma + j\sigma + 1$	$-\lvert\varphi(r)\rvert_\sigma + (j+1)\sigma$	$S_{\lvert\tau+j\rvert_{4t}}$	$1 \leq j \leq t-1$
$-\lvert\varphi(r)\rvert_\sigma + t\sigma + 1$	90	$S_{\lvert\tau+t\rvert_{4t}}$	t
IV Quadrant			
$\alpha(D)$			
Smallest $\xi(\lvert\tau - j\rvert_{4t})$	**Largest** $\Lambda(\lvert\tau - j\rvert_{4t})$	**Wind**	j
1	$\lvert\varphi(r)\rvert_\sigma - 1$	S_τ	0
$\lvert\varphi(r)\rvert_\sigma + j\sigma$	$\lvert\varphi(r)\rvert_\sigma + (j+1)\sigma - 1$	$S_{\lvert\tau-j\rvert_{4t}}$	$1 \leq j \leq t-1$
$\lvert\varphi(r)\rvert_\sigma + t\sigma$	90	$S_{\lvert\tau-t\rvert_{4t}}$	t

Table 3.3: The angles $\alpha(D) \in (0, 90]^\circ$

I Quadrant				IV Quadrant			
$\alpha(D)$		**Wind**	j	$\alpha(D)$		**Wind**	j
1	10	S_1	0	0	19	S_1	0
11	40	S_2	1	20	49	S_0	1
41	70	S_3	2	50	79	S_{11}	2
71	90	S_4	3	80	90	S_{10}	3

one without payload μ_i, are used to compute the overall drone's energy consumed to complete the mission.

Table 3.3 reports the sector winds and the range of values of $\alpha(D) \in (0, 90]^\circ$ associated to each sector wind of the I and IV quadrant when $\varphi(r) = 50^\circ$, $\tau = 1$, and $t = 3$, as in the Example in Fig. 3.4. Our discussion to find OSR is recap in Alg. 1. In conclusion:

Fact 1. *Given the* MT *route r and γ_r, the delivery point P, the wind ω, the drone speed υ_d, and fixed t of the compass rose,* OSR *computes in $O(1)$ time the two straight line TP and PL that consume the minimum energy to serve P from r.*

3.4 Multi-line extension and Simulations

We evaluate the impact of selecting the optimal service route OSR on energy consumption. We study the energy cost when the drone operates from a single-line (case study *line*) and it can select from which line to operate among multiple lines (case study *multi-line*). In the first case, the MT travels on a straight line (i.e., highway) and the deliveries occur in the area in front of the highway, in the second case the MT circumnavigates the area where the deliveries occur.

We consider two types of winds in the experiments: *exhaustive winds* EW and *real winds* RW. In EW, we vary ω_d with regular steps between 1° and 360° and consequently the relative wind $\varphi(r)$ on r varies. In RW, the wind varies according to the data collected at different hours of the day in two locations in Corsica during winter days: the wind of Corte (inland) is characterized by low wind speed, while those of Cap Corse (seaside) have high wind speed.

Section 3.4.1 describes the generation of our tests, while Section 3.4.2 presents the results.

3.4.1 Description

Line

We consider the MT's route $r: \ y = 0$ with $\gamma_r = 0$ and a single fixed point P to be served at distance 6Km from r. To serve P we compare two algorithms:

- Optimal trajectory (OSR): Alg. 1 is applied to r and P to find the take-off T and landing L points.
- Shortest trajectory (E) : P is served selecting the shortest route HP from r to P, that is, the take-off and the landing points coincide with H.

Multiline

In this scenario, we consider a delivery area, which consists of a circle of radius $R = 5$km. Inside, the circle we generate a convex polygon with $n = 8$ sides. Setting the starting point of the MT's route as one of the vertices of the polygon, the route follows by the sequence of vertices $\{s_0, s_1, ..., s_7\}$ traveled in a clockwise direction. An example is illustrated in Fig. 3.5. Even if the wind does not change, the relative winds on the lines change, and so the optimal service route may differ from line to line. We consider a set of 20 delivery points randomly generated inside

the polygon. For each delivery, the optimal service route is obtained by applying Algorithm OSR from each polygon side. Four different algorithms are proposed:

- Same Side Optimal trajectory (SS-OSR): This approach applies Alg. 1 to each side of the polygon and returns the pair T and L on the side that consumes the minimum energy. Both T and L are forced to belong to the same line. To be precise, we apply OSR only to those sides that contain the projection H of P. Since each polygon's side has a limited size, not all the values in $(0, 90]$ can be taken by $\alpha(D)$. SS-OSR is applied to serve P_0 in Fig. 3.5.

- Multiple Optimal trajectory (MS-OSR) : It is similar to SS-OSR, but T and L are not forced to belong to the same side. The computations of TP and PL in OSR are uncoupled. First, OSR is applied to each side of the polygon to find the minimum energy take-off trajectory TP, and then OSR is applied to each side of the polygon to find the minimum energy landing trajectory LP. The constraint that the MT cannot go back is respected selecting T that (clockwise) precedes L on the MT route. As an example, MS-OSR is applied to find T and L to serve P_1 in Fig. 3.5.

- Same Side Shortest distance (SS-E): it is like SS-OSR but it invokes for each side, algorithm E instead of OSR. For each side, it is considered the trajectory that serves P by the projection H of P on that side. Then, P will be served starting from the side that minimizes the energy consumption. Recall that this is not necessarily the side closer to P because the energy spent depends also on the relative wind on the take-off and landing trajectory. SS-E is applied to serve P_2 in Fig. 3.5.

- Multiple Shortest trajectory (MS-E): it is like MS-OSR but it applies E instead of OSR. This solution serves P using only the shortest Euclidean routes, as SS-E. However, it allows that the take-off and landing point lie on different sides, as MS-OSR. MS-E is applied to serve P_3 in Fig. 3.5.

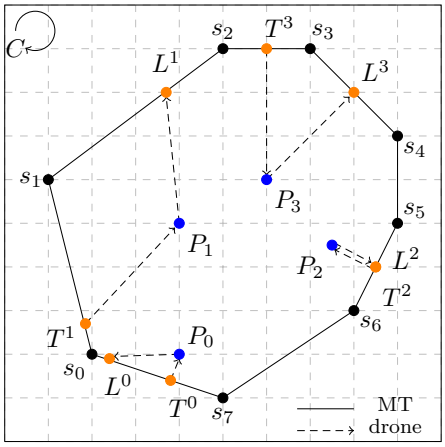


Figure 3.5: Illustration of the possible D trajectories.

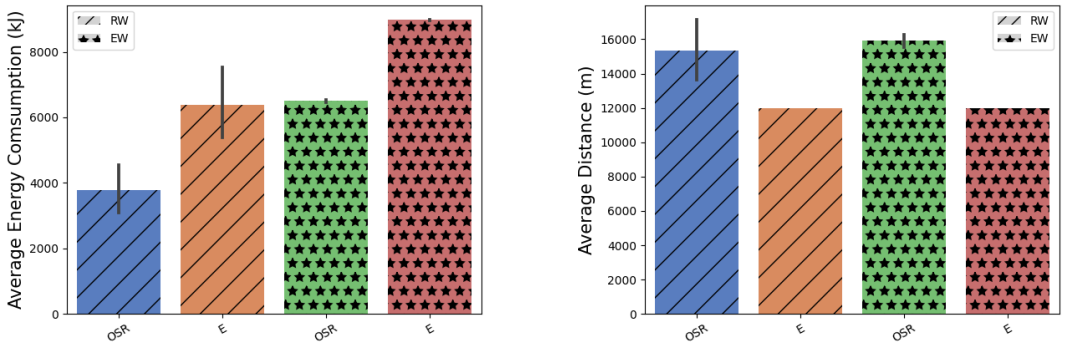
3.4.2 Evaluation

We evaluate our solution computing the energy and the distance to serve a single delivery point using the above algorithms. We evaluate the distance to show that running in a favorable wind, D can save energy even if the traveled distance is longer than the Euclidean distance between the MT's route and the delivery point P.

In the line scenario, P is fixed. We repeat the mission under different winds. For each mission, we compute the energy and distance costs and we average the costs over all the evaluated different winds. In the multi-line scenario, for each mission, we fix a global wind w and we serve all the 20 points. We compute the average delivery energy for each mission as the total depleted energy divided by 20. Then, we repeat the mission under different winds and again we average the delivery energy over all the missions.

Line

Fig. 3.6 compares the average energy and distances depleted by Algorithms OSR and E in a single line scenario. With Algorithm E, the drone always travels the same distance 12Km as P is at 6Km from r. For each mission, we compute the energy consumed by OSR and E. Note that E does not spend the same amount of energy going to P and coming back not only because going the drone carries a payload while coming back the drone is empty, but also because the sector wind may be different due to the different verses. The plots report the average energy consumption when $v_d = 20$m/s, $\gamma_r = 0$, $\kappa = 6$, and $t = 3$. The results labelled EW



(a) Average energy.

(b) Average distance.

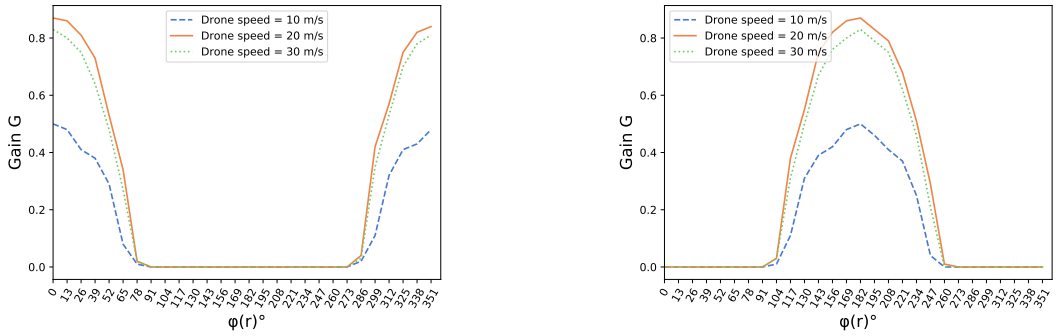
Figure 3.6: The performance of OPT and E in the line scenario with $v_d = 20$m/s and $\kappa = 6$.

are obtained by fixing $\omega_s = 20$m/s and by varying ω_d among all the multiples of 13° and 30° in $(0, 360]$. Instead, the results labeled RW are obtained by varying ω, both speed and direction, on a set of winds collected in Corse in 10 days.

The results in Fig. 3.6 are very interesting: under RW, for each mission, OSR saves forty percent of the energy of E although it crosses one-fourth longer distances than E. Under EW, OSR saves thirty percent of the energy with respect to E and travels one-third more than E. These results show that our investigation of the impact of the wind on flight energy is worthy. The EW and RW saving are quite similar, although EW saves more than RW because EW spans all the possible winds and half of them are favorable winds, while RW follows the weather and not all the winds have the same probability to occur. Consequently, EW is more stable than RW experiments as shown by the confidence intervals.

The EW tests scan all the possible winds. Thus, we plot the *energy gain* G which is defined as the ratio between the energy saved by Algorithm OSR with respect to E over the energy spent by E. Fig. 3.7 plots the gain when ω_d is a multiple of 13° in $(0, 360]$. By definition,G does not depend on the distance PH but only on the energy spent per unit of distance.

Inspecting Fig. 3.7(a), OSR gains with respect to E in half of the exhaustive winds. G is superior to 0 when $\varphi(r) = |\omega_d - \gamma_r|_{360} = \omega_d \in (0, 90] \cup (270, 360]$ and it can reach up to the 80% under relative winds. G decreases when ω_d approaches 90° or 270°. In all the other cases, $90 < \omega_d \leq 270$, OSR and E coincide and $G = 0$, that is, the shortest path is the best route. We conclude that the savings are



(a) $\gamma_r = 0°$ (b) $\gamma_r = 180°$

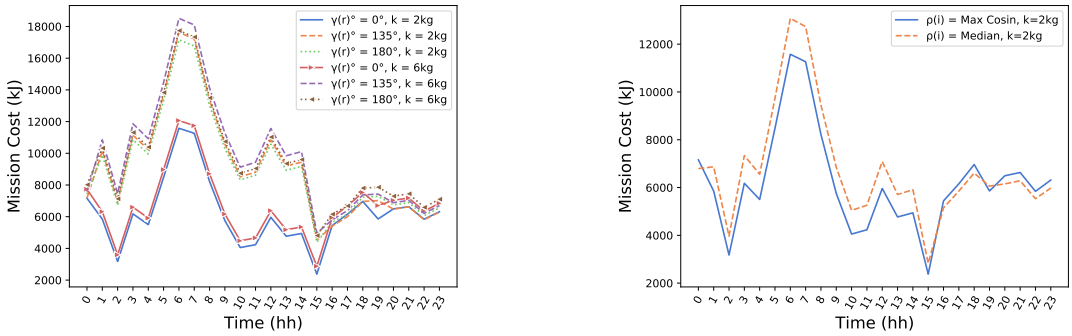
Figure 3.7: Tests EW with $\omega_s = 20$m/s and $\kappa = 6$kg.

higher in the line scenario the more $\varphi(r)$ falls in $(0, 90] \cup (270, 360]$.

Note that changing γ_r, the curve of the gain just rotates. For example, when $\gamma_r = 180°$ in Fig. 3.7(b), OSR will beat E when ω_d falls in $(180, 270] \cup (90, 180]$, while OSR and E coincide when ω_d in $(0, 90] \cup (270, 360]$.

In Fig. 3.7, we also evaluate the impact of different drone speeds on the gain. Observe that the gain is maximum when $\omega_s = \upsilon_d$, which is the case when the thrust Π that has to be provided to let the drone fly is the minimum one. It is worthy to note that the gain decreases more when the drone has to be slowed down ($\upsilon_d = 10$m/s) than when the drone must accelerate ($\upsilon_d = 30$m/s) with respect to the wind speed. So a stronger thrust is required to decelerate than to accelerate.

Finally, Fig. 3.8 reports the energy spent under different payloads and different representatives. To evaluate the impact of these parameters we select the real winds of a typical day of the seaside in Corse. As one can see, the impact is moderated in both cases. In particular, the increase on the depleted energy with different payloads is almost independent of the line direction γ_r and coherent with the values tabulated in Table 3.1. The mission energy cost, when the representative $\rho(i)$ is the median angle in S_i, is slightly larger than the energy cost when $\rho(i)$ is the angle in S_i whose cosine is maximum in absolute value. However, a deeper investigation of real data is required to decide which is the best representative.



(a) Different payloads

(b) Different representative $\rho(i)$.

Figure 3.8: Test RW with $v_d = 20\text{m/s}$ and $t = 3$.

Multiline

Here, the drone has the freedom to decide from which side of the polygon to serve the delivery point.

Fig. 3.9 shows the energy consumed and the distance traversed to serve a single mission under EW and RW. Precisely, the four algorithms SS-OSR, MS-OSR, SS-E, and MS-E are examined.

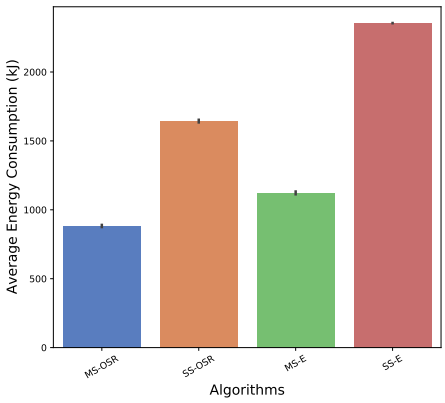
MS-OSR is the most energy-efficient algorithm in both wind scenarios, although it travels on average the longest distance. Under the EW winds, MS-OSR more than halves the energy of SS-OSR for a single delivery by the freedom of selecting T and L on two different sides. The same happens for SS-E and MS-E. As seen for the single line, SS-OSR saves about 40% of the energy of SS-E and MS-OSR consumes one-third of the energy of SS-E, thus saving at least 60%. MS-OSR saves 20% of energy with respect MS-E. It is worthy to note the good performance of MS-E with respect to SS-OSR: although SS-OSR depletes less energy than MS-E, (precisely, SS-OSR consumes 20% less than MS-E), one cannot forget to read the MS-E performance in light of its simplicity: MS-E that just selects its trajectories among the shortest paths. Regarding the traveled distances, algorithms that save more energy travel more. Note that, since the polygon is inscribed in a circle of radius $R = 5\text{Km}$, each delivery point is at most at distance 10Km from the polygon's side, but all the algorithms serve the delivery traversing on average a distance between 3Km and 4.5Km. Under the RW winds, the algorithms follow exactly the same behavior as under EW. However, as seen for the single line scenario, the algorithms save slightly less energy, the traversed distance is shorter,

and the results are more variable as evidenced by the wider confidence interval.

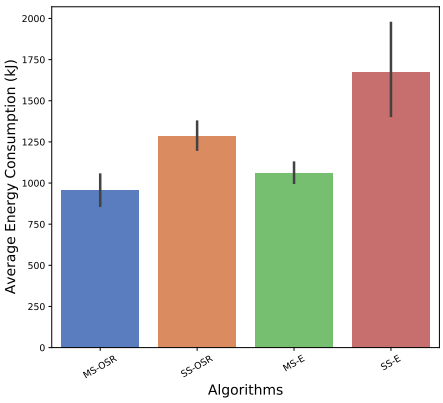
In conclusion, being able to choose between several sides of the polygon we save a lot. In fact, not only do we choose the side with the most favorable wind, but also the distance decreases significantly: a couple of deliveries can be fulfilled easily with regular batteries of 5kJ in use in nowadays COTS drone against the single delivery achievable in the line scenario.

Concluding this Chapter, for the first time at the best of our knowledge, we include the wind in the drone routing problem. We investigate the problem of finding the best route for a delivery drone that takes off from a truck and must deliver some goods to a know destination. Two algorithmic solution are proposed: for when the truck moves on a line (as on a highway) in front of the deliveries, and for when the truck moves on a multi-line that bounds a convex area where the deliveries take place.

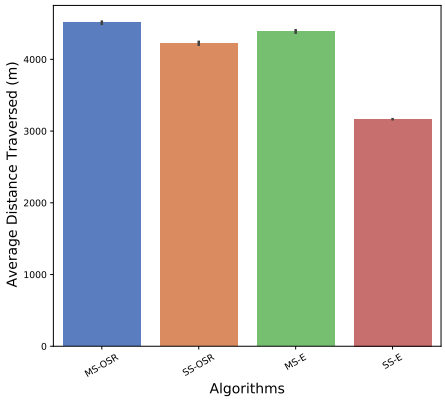
In the future, we would like to confirm our findings by using flight simulators, and then by extending our investigation to a test-bed made of COTS drones. We also will include discussions on what can go wrong, e.g. time of day/night when the winds may be unstable, and on how to include the flight time scheduling because there are different winds at different hours of the day.



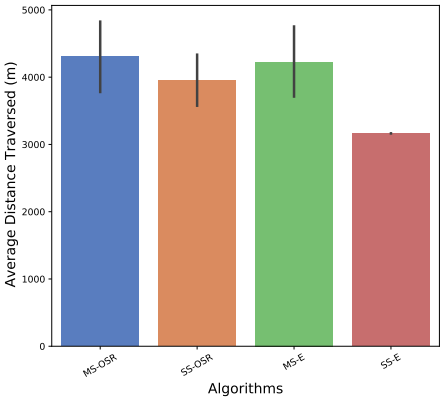
(a) Average energy with EW.



(b) Average energy with RW.



(c) Average distances with EW.



(d) Average distances with RW.

Figure 3.9: Multi-line tests with $\upsilon_d = 20\text{m/s}$, $\kappa = 6\text{kg}$, and $t = 3$.

Part II

Localization of Sensors with Drones: Troubles in practical implementation

In the last period, unmanned aerial vehicles such as drones, are extensively and widely used in many civilian applications [45,46,47,48,49,50]. In fact, due to their ability to perform very challenging tasks, in this thesis, drones are considered when employed for localizing missing people [6, 51, 52, 53] or managing the search and rescue operations [54,55,56] after disastrous events, in precision agriculture [57], or for delivering small packages in a last-mile delivery scenario [21,30,31,58,59,60,61].

Localization of sensors, presented in this Part, can help to search and rescue people after natural disasters, to monitor the structural health of building, or even to track people during the COVID-19 pandemic in order to decrease the circulation of the virus.

Considering also the increasing interest in the Internet of Things (IoT), and in general in the Wireless Sensor Networks (WSNs), the problem of localizing sparse wireless sensors is more and more appealing to researchers. Due to their limited size and low cost, those sensors can be installed everywhere for any specific task.

Technically speaking, the localization problem aims at estimating the position of *ground devices* (GDs) deployed on the WSN. For this important task, special devices whose positions are known a-priori, called *anchors*, are in charge of localizing the GDs. In the literature, localization can be classified in respect to the type of anchors deployed, i.e., *static anchors* (SAs) or *mobile anchors* (MAs). The former scenario requires a discrete amount of SAs with fixed positions and not only it is not affordable in terms of costs when the WSN is large, but such fixed infrastructure cannot be quickly reused elsewhere. The latter one instead, only requires a single MA (e.g., rover, drone) with GPS capabilities and fresh coordinates while moving. Another important difference is that localization through the use of a MA is inherently more accurate compared to the SAs, hence, in the next chapter of this thesis, Chapter 4, only the localization with a single MA (i.e., a drone) is considered. In literature, the localization can also be classified with respect to the type of algorithms used, thus, broadly categorized as *range-free* (RF) or *range-based* (RB) approaches [62]. In the former, the position is estimated only by discovering if the GD and MA are one in the range of the other. In the latter, the position of the GD is estimated by taking measurements (e.g., distance, angle, signal strength) between it and the MA. Usually, RB algorithms are known to be more accurate than RF ones but at the cost of additional specialized hardware. Finally, among the RF algorithms, the *radius-based* algorithms assume the knowledge of the transmission radius, while the radius-free ones do not. In the next Chapter, we evaluate on a test-bed the accuracy of many RF algorithms that exploit the *heard/not-heard* (HnH) method, consisting of detecting two consecu-

tive messages transmitted by the MA, one heard and one not-heard from the GD, called endpoints.

Following, the state-of-the-art about localization in WSNs, starting from a brief overview about known techniques with SAs, and then surveying more in detail techniques using a MA. Finally, few works that implemented real test-beds about localization.

The Static Anchors Scenario: In the literature, different algorithms exist to tackle the issue of localization of GDs using SAs. DV-HOP based techniques [63] and Amorphous algorithms [64] approximate the distance between GDs using the number of hops between them and estimating the average hop distance inside the WSN. Then, using the trilateration method, each GD computes its own position. The Centroid algorithm used in [65] is another technique where each SA broadcasts its position to the surrounding, and each GD computes its own position calculating the average of all the coordinates of the SAs that it can hear. Other works propose solutions based on the Approximate Point in Triangulation (APIT) [66] whose goal is to divide the area in triangles in which unknown GDs reside. In each overlapped section (i.e., a polygon) of triangle area, the position of the unknown GD is computed calculating the centroid. The main issue for these solutions is the relatively high number of SAs required for an acceptable localization error.

The Mobile Anchor Scenario: All the following algorithms emulate multiple SAs with a single MA that continuously broadcasts its position. For localizing GDs, a MA has to plan a route (static path) in advance inside the WSN. On that route, the MA estimates the distance between itself and the GDs in range, and eventually, the GDs' positions are computed performing trilateration. In [67] three different 2D movement trajectories have been studied, i.e., SCAN, DOUBLE-SCAN, and HILBERT. The distance between two consecutive segments of the trajectories is defined as the resolution. The simplest algorithm is SCAN, in which the MA follows a path formed by vertical straight lines interconnected by horizontal lines. Essentially, the MA sweeps the area along the y-axis. The main drawback is that it provides a large amount of collinear endpoints. In order to resolve the collinearity problem, DOUBLE-SCAN sweeps the sensing area along both the x-axis and the y-axis. However, in this way the path length is doubled compared with SCAN. Finally, a level-n HILBERT curve divides the area into 4^n square cells and connects the centers of those cells using 4^n line segments, each of length equal to the length of the side of a square cell. Generally, HILBERT provides more non-collinear endpoints but the path length can be very long if the resolution

increases. All the above techniques are based on straight lines and suffer from collinearity problem. In order to heavily reduce collinearity, S-CURVES [68] has been introduced, which is similar to SCAN except that it uses curves rather than straight lines. Even though the collinearity problem is almost resolved, the main problem is that it does not properly cover the four corners of the squared sensing area. One of the best techniques is LMAT [69]. The main idea is to plan a series of MAs in such a way as to form equilateral triangles, avoiding collinearity problems. Each GD inside a triangle is localized by a trilateration procedure using the three vertices.

All the previous algorithms are range-based since they rely on measurements (signal strength, distance, etc.). Only a few range-free techniques have been proposed in the literature. Among them, the ones proposed by Xiao [70] and Lee [71] rely on a ground MA, while the one proposed by Betti [72], relies on a flying MA.Briefly, Xiao exploits the HnH method and the radius of the antennas, to find a specific area that contains the GD. While moving over a straight line, the MA records the first pair of heard endpoints called arrival and departure points. From those points, it also computes (knowing the interval between successive transmission signals) a pair of non-heard points, namely the pre-arrival and post-departure positions. Using those four points as centers, four identical circles are drawn (using the radius of the antenna) and the resulting intersections form an area, where the GD resides. Lee works very similarly to Xiao, except that four circles are drawn from the pairs of heard endpoints, using different radii, creating again an intersection area where the GD resides. Drf instead, does not use the radius of the antenna but uses the notion of chord and its geometrical property that state that each perpendicular chord's bisector, passes through the center of the circle. Therefore, a bisector of another non-parallel chord and the previous one, intersect in the center of the circle. Using the HnH method moving in a straight line, the MA finds multiple points belonging to the circumference of the GD transmissions area. Then, it connects those points with segments, thus acquiring chords, from which it calculates the center of the circle (i.e., the GD).

Implemented Test-beds: To the best of our knowledge, only a few test-bed implementations have been done aimed at comparing the performance among RF algorithms with a MA. Recently in [73], a test-bed using inexpensive UWB antennas and a drone as MA evaluates the localization accuracy of the RF Drf algorithm proposed in [72]. Such an algorithm strictly relies on the good quality of the antenna radiation pattern and requires simple geometrical rules for estimat-

ing the GD's position. Unfortunately, in practice, the experimental localization error obtained by the implemented DRF algorithm is large. Another recent study in [74] shows that the irregularities of the hardware antenna radiation pattern can heavily affect the air-to-ground (A2G) link quality between a MA (drone) and a GD. The authors study the A2G link quality of BroadSpec UWB antennas from Time Domain Inc by observing the Received Signal Strength Indication (RSSI). They show a dependency of the link quality on the antennas' orientations, their elevation, and their distance. The authors in [75] report the UWB A2G propagation channel measurements in an open field using a drone. Three scenarios with different obstacles are considered while a drone was orbiting above a GD at different altitudes and ground distances. Also, different antenna orientations are considered for the drone. Experimental results show that the received power is highly dependent on the antenna gain of the line-of-sight (LoS) component in the elevation plane when the antennas are aligned (same orientation). In a work proposed in [76], authors investigate a time difference of arrival (TDoA) based approach for localizing drones taking into account a simple A2G 3D antenna radiation pattern. Experimental results show that accounting for antenna effects makes a significant difference and reveals many important relationships between the localization accuracy and the altitude of the drone. Most importantly, they finally show that the localization performance varies in a non-monotonic pattern with respect to the drone altitude.

Concluding, considering that the RF algorithms have been around for a long time, that the algorithm tested in [73] on the field using a drone (i.e., flying MA) resulted in bad performance, and that the irregularities showed in [74, 75, 76] are for A2G links, we started to think that altitude could be one of the main causes of poor results for the algorithm tested in [73] and in general for RF algorithms. Therefore, these considerations motivate us to investigate more about the accuracy of RF algorithms at different altitudes and the quality of antennas, and our findings are shown in the next Chapter.

Chapter 4

Implementation Issues for Range-free Localization Algorithms[1]

In literature, all the algorithms are described under the ideal model, that assumes that both MA and GD are equipped with an isotropic antenna whose radiation pattern is a perfect sphere. Consequently, the transmission and receiving areas, which are given by the intersection of the sphere with the earth surface, are perfect circles. The ideal model can be emulated in reality only with expensive hardware on noiseless areas, which are not the usual operating conditions of most localization applications that occur, for example, in precision agriculture or search-and-rescue context. To reproduce the typical operating conditions, we adopt for our test-bed a set of inexpensive DecaWave DWM1001 Ultra-Wide-Band (UWB) antennas and an off-the-shelf 3DR Solo drone as the flying MA. Our pursued goals can be summarized as follows:

- We experimentally evaluate the performance of many localization algorithms, with very little knowledge on the antenna radiation pattern;
- We study how the MA's altitude can affect the localization performance.

The experimental results raised difficulties, doubts, and questions. Firstly, collecting a large set of endpoints on the field has proven to be challenging. So, we search for the statistical distribution that best fits the already observed endpoints

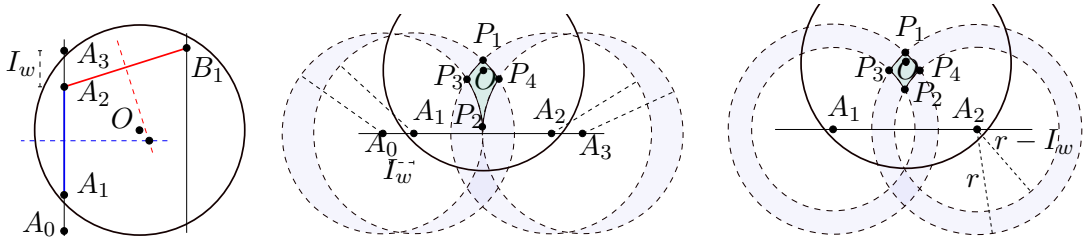
[1]This work has been published to Pervasive and Mobile Computing Journal ⟶ Publications n. [5,6]

for generating a large new set of synthetic endpoints. Such a set is used for testing the compared algorithms. To mitigate the poor observed performance in the real scenario, we decide to exploit the DWM1001's capability of taking distance measurements to gain awareness of the antenna radiation pattern. Despite this strategy makes the RF algorithms actually RB, the original algorithmic rules are kept, and the awareness of the exact antenna radiation pattern, though incomplete, significantly improves the performance.

4.1 The Range-free Algorithms

Following are described the RF algorithms that we compare to test their feasibility in a real context. We start defining a *rover* as a MA that moves on the ground, and a *drone* as a MA that flies in the sky. During the localization procedure, called mission, the MA visits specific points in its path, called *waypoints*. Along this path, the MA continuously transmits a *beacon* that can be used by any GD within the communication range for estimating its position. The beacon includes the current MA's GPS position and it is sent at regular intervals of time. The distance among any two consecutive transmitted beacons is called *inter-waypoint* distance I_w and depends on the MA's current speed. All the algorithms that we study exploit the HnH method relying on the detection of endpoints. Performing HnH, the GD learns that the MA is currently transmitting at its transmission area border. Moreover, the GD learns that the last (first) not-heard beacon is at a distance I_w from the first (last) heard beacon (endpoint), and hence at a distance no more I_w from the edge of the transmission area. In general, three applications of HnH are required for localizing the GD.

In the following, we describe three RF algorithms that we evaluate, namely, DRF [72], XIAO [70], and LEE [71]. We also introduce a new variant of DRF, called DRFE. In these algorithms, the MA travels along a static path for localizing the GD. The static path is formed by a sequence of consecutive straight segments in which the MA regularly broadcasts message beacons (its current position) that the GD is listening to. Once the GD has collected enough information, it can finally compute and estimate its position according to the performed algorithm.



(a) Drf. (b) Xiao. (c) Lee.

Figure 4.1: The Drf, Xiao, and Lee localization algorithms. In Xiao e Lee there are two symmetric intersection areas: a third point (not illustrated) is required to find and disambiguate the intersection area where GD resides.

4.1.1 The Drf Algorithm

Drf [72] is a lightweight RF radius-free algorithm designed for drones. This algorithm is based on the notion of *chord*. In geometry, the perpendicular bisector of any circle's chord passes through the center O of the circle itself. So, the bisector of another non-parallel chord and the previous one intersect at O point. In Fig. 4.1(a), the GD is located at the point O, while initially the MA travels along the segment that intersects the points in sequence A_0 first and then A_3. The radio receiving area of the GD is identified by the circle centered at O, so if the MA transmits a message outside that circle (e.g., in A_0), the GD cannot hear any message. However, when the MA crosses such a circle and transmits in A_1, the GD can now receive and record messages, because the relative distance between them is less than or equal to the transmitting/receiving radius. The same reasoning can be applied for the points A_2 (heard) and A_3 (not heard). Accordingly, the first chord is denoted by the segment with endpoints A_1A_2. It is easy to understand that when the MA crosses the GD's receiving area along another segment (e.g., the one that intersects the point B_1), another endpoint is detected, and eventually two chords are identified by the pairs A_1A_2 and A_2B_1. Finally, the GD starts to estimate its position once it has detected these two chords computing first the associated perpendicular bisectors (dashed lines) and then their intersection point (which can be different from point O). The detection of chords incurs several problems that eventually affect the localization accuracy. Recalling that the MA regularly broadcasts its current position (waypoint) at discrete intervals of time and that two consecutive waypoints are at distance I_w, the endpoints of the chords may not exactly fall on the circumference of the receiving disk, even if the receiving disk is a perfect circle (e.g., A_2 and A_3).

4.1.2 The Xiao Algorithm

Xiao [70] is a RF radius-based localization algorithm initially developed for ground MAs. Like Drf, the Xiao algorithm exploits the HnH method in order to detect special points used for building a constrained area that bounds the GD's position. Unlike Drf, Xiao also uses the value of the communication radius r. In Fig. 4.1(b), the GD is located at the point O while the MA travels along the segment that intersects first the point A_0 and then A_3. Once applied the HnH method, the GD initially detects the first pair of heard endpoints A_1 and A_2. However, since the segment lies on a straight line and the value of I_w is known, the GD can also compute two additional non-heard beacons, i.e., A_0 and A_3, associated with A_1 and A_2, called pre-arrival and post-departure. Then, four circles of radius r centered at each of these four points are drawn. Those circles create two symmetrical intersection areas (e.g., the first one is bounded by the points P_1, P_4, P_2, P_3) where the GD may reside. Hence, the GD's position can be at the "center" of one of the two intersection areas. In order to disambiguate in which intersection area the GD resides, a third HnH beacon is required, and the final estimated position is the one which has the closest distance, from that third point, to the radius. The definition of center varies depending on whether the intersection area is delimited by four or five vertices [70].

4.1.3 The Lee Algorithm

Lee [71] is a RF radius-based algorithm very similar to Xiao. It builds a constrained area using the HnH method and the knowledge of both r and I_w, similarly to Xiao. In Fig. 4.1(c), once the GD has detected the two extreme endpoints (A_1 and A_2), it traces two circles of radius r and $r-I_w$ on both the points, which create two annuli, intersecting in two distinct and symmetrical intersection areas (e.g., one is bounded by the points P_1, P_4, P_2, P_3). Finally, GD resides at the center of one of such areas, and a third endpoint is used to disambiguate the correct one.

4.1.4 The Proposed DrfE Algorithm

Now we present a new RF algorithm, called DrfE, that shares the "chord" idea with the Drf algorithm and the radius information r with Xiao and Lee. In the special case that the two endpoints A_1 and A_2 that delimit the chord exactly lie on the circumference, the GD resides on the point O of the perpendicular bisector

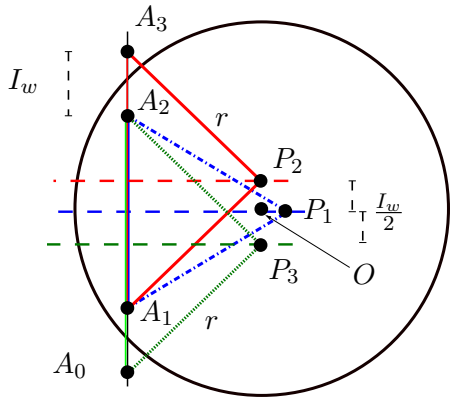


Figure 4.2: The points P_1, P_2, and P_3 of the DRFE algorithm.

that is distant r from the two endpoints. Precisely, there are two possible locations for O, one on the left and one on the right of the chord. Usually, using a third endpoint non-collinear with A_1 and A_2, it is possible to disambiguate the correct intersection.

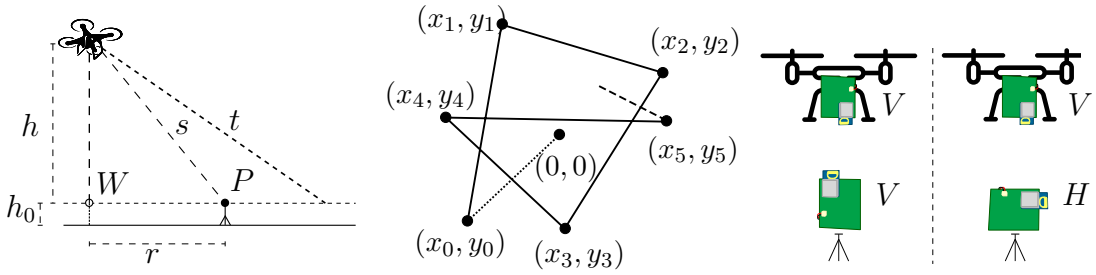
In general, since the MA's path is sampled with discrete beacons at distance I_w among them, A_1 and A_2 may not lie on the circumference and thus GD may not exactly reside on the perpendicular bisector of the chord $\overline{A_1A_2}$, but in its vicinity. So, to find the GD's position, DRFE repeats the above construction for the three chords $\overline{A_1A_2}$, $\overline{A_0A_2}$, and $\overline{A_1A_3}$, where A_0 and A_3 are the two non-heard beacons associated with the endpoints A_1 and A_2, as illustrated in Fig. 4.2. From the three chords $\overline{A_1A_2}$, $\overline{A_1A_3}$, and $\overline{A_0A_2}$, three intersection points P_1, P_2, and P_3 are obtained at distance r from the endpoints of their chord. In other words, P_1, P_2, and P_3 form three isosceles triangles $\triangle(A_1A_2P_1)$, $\triangle(A_1A_3P_2)$, and $\triangle(A_0A_2P_3)$ with two oblique sides of equal length r. As before, this construction finds three vertices on the left of the chord and three vertices on the right that will be disambiguated using another non-collinear endpoint (not illustrated). Let us suppose that GD is on the right of the chord $\overline{A_1A_2}$. DRFE places GD at the centroid of P_1, P_2, and P_3 on the right of the chord $\overline{A_1A_2}$. Since the RF radius-based algorithms require the knowledge of the transmission radius, in the next next we describe how to collect such information in a real test-bed.

4.2 Test-bed Setup

In this section, we describe the test-bed and the used hardware. We fix a Cartesian coordinate system with origin at the special position HOME $(0, 0, h_0)$, with $h_0 = 1$m. At HOME, we place the GD's antenna at the top of a tripod of height h_0. Also, the MA is equipped with an antenna. When we set $h = 0$m, we refer to a rover mission where the MA and the GD are placed at h_0, while when we set $h > 0$m, we refer to a drone mission flown at an altitude $h_0 + h$.

In our test-bed, we employ different hardware components: a few DecaWave DWM1001 UWB antennas [77], a Raspberry Pi, and a 3DR Solo drone [78]. According to DecaWave, the transmission radius is 60m and hence, we set the manufacturer radius to $r_0 = 60$m. The main component that pilots the MA and sends commands to the GD is the Raspberry Pi. The Raspberry can be used for the experiments on the ground using a rover, or together with the drone for the aerial ones. In the former case, i.e., MA as a rover, we simulate the rover's behavior by just walking in the field at a regular walking speed (about 3km/h as measured by a smart-watch) keeping the Raspberry on the hands at $h_0 = 1$m above the ground. Moreover, since the rover has to send GPS positions, we rely on a cheap USB GPS module connected to the Raspberry. Finally, in the latter case, i.e., MA as a drone, we use the 3DR Solo drone which is able to fly up to 25min [78].

A mission is a static path hat consists of n segments S_i, $i = 0, \ldots, n-1$. Such a list of n segments is made by generating $n + 1$ random points in the deployment area. Each segment is delimited by two random points, and any two consecutive segments share one random point by setting the waypoints' coordinates (x_{W_i}, y_{W_i}, h) for each W_i, $i = 0, \ldots, n-1$ (see Fig. 4.3(b)). When all the waypoints are generated, the mission starts. Once the first waypoint of s reached,



(a) The drone and the ground device P.

(b) The MA's random path.

(c) VV and VH.

Figure 4.3: The test-bed setup.

the drone/rover starts to send message beacons according to its current position by converting GPS coordinates in local Cartesian (x, y) positions. This process continues until the MA reaches the last waypoint of · When the mission is accomplished, the MA comes back to HOME.

Regarding the GD, in our experiments, we set its antenna on the tripod placed at HOME, lying on two different planes, as sketched in Fig. 4.3(c) (left and right). According to the DecaWave's Datasheet document [79], there are three planes in the spherical space with respect to the antenna's center, i.e., xz, xy, and yz. Each plane experiences a different radiation pattern. In Fig. 4.3(c), we show the GD's antenna when it is in the xz plane (bottom left) and in the yz plane (bottom right). We denote as *vertically placed* an antenna that lies in the xz plane, and as *horizontally placed* an antenna that lies in the yz plane. The drone's antenna is always vertically placed in the xz plane (Fig. 4.3(c), top left and top right) keeping the UWB transceiver at the bottom for guaranteeing the most available free space. For simplicity, we indicate the first configuration (Fig. 4.3(c), left), where the two antennas lie on the same side, but in opposite direction, with vertical-vertical (VV); whereas we refer to the other configuration (Fig. 4.3(c), right) with vertical-horizontal (VH).

4.3 Antenna Analysis

In this section, we recap the UWB technology, report the DecaWave's technical datasheet information, and analyze the experimental data.

4.3.1 The Ultra Wide Band Technology

UWB is a promising radio technology that can use a very low energy level for short-range, high-bandwidth communications over a large portion of the radio spectrum. Nowadays, its primary purpose is in the field of location discovery and device ranging. Differently from both Wi-Fi and Bluetooth, UWB is natively more precise and accurate, uses less power and, as production of UWB chips blows up over time, holds the promise of a lower price point. Moreover, UWB offers relative immunity to multipath fading.

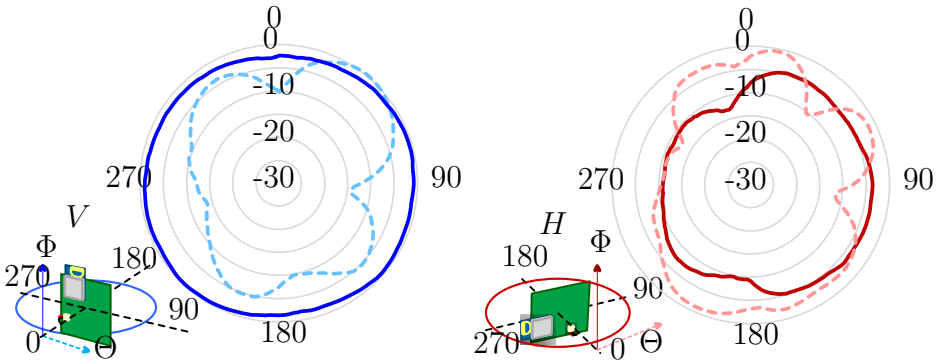
In this Chapter we rely on a kit of DWM1001 UWB antennas produced by DecaWave. According to DecaWave's datasheet document [77], those antennas provide 10cm accuracy for the measurements. Moreover, those chips have a 6.5GHz center frequency, and have a point-to-point range up to 60m in a line-of-sight

(LoS) scenario and up to 35m in a non line-of-sight (NLoS) scenario. Although the DWM1001 chip transmitting power is set to −41.3dBm/MHz, and the typical receiver sensitivity is −93dBm/500MHz [77], the received power is influenced by the antenna polarization.

4.3.2 Datasheet Antenna Information

Fig. 4.4 shows the antenna radiation patterns of the UWB antennas according to DecaWave's document (*Datasheet for the DWM1001C*, Tab. 12 [79]) for different configurations. The solid dark line of Fig. 4.4(a) shows that it is possible to obtain the same gain in all the directions in the xz-plane when an antenna vertically placed (i.e., on xz-plane) is observed by another antenna which shares the same *vertical* orientation (Φ *polarization*), i.e., they are concordant. We recreate this situation by implementing the antennas as VV. Thus, we expect that the VV configuration experiences the same gain at different angles, at least when the two antennas are at the same height.

Both the dashed lines of Fig. 4.4(a) and Fig. 4.4(b) refer to the VH configuration because they show the gain when an antenna is observed by another antenna *perpendicularly* oriented (Θ *polarization*), i.e., they are discordant. Both the dashed lines have nulls at certain angles that can limit the gain and can introduce "holes" and "bubbles" in the pattern. Thus, we expect that the VH configuration experiences different gains at different angles, and we also expect a relevant variability given that the dashed lines of Fig. 4.4(b) and Fig. 4.4(a) are different, although the relative position of the antennas seems to be the same. Analyzing these technical data, it seems that the gain is omnidirectional at least when the two antennas are

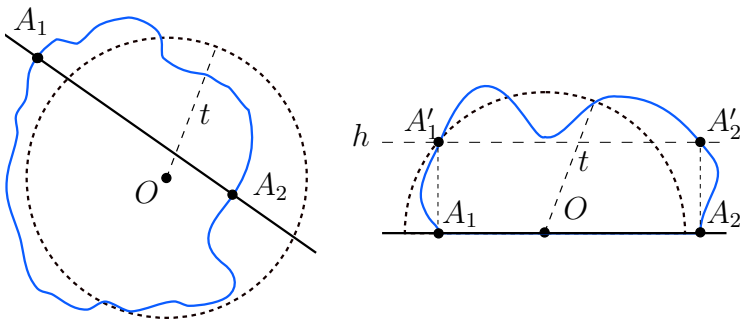


(a) xz-plane. (b) yz-plane.

Figure 4.4: The DWM1001 radio pattern: dBm vs angle.

Table 4.1: The DW1000 elevation gains (in dBm) at 6.2GHz.

		Θ (discordant)	Φ (concordant)
yz-plane	peak	0.30	2.92
	average	-6.99	-3.04
yz-plane	peak	0.26	1.39
	average	-5.74	-3.90



(a) xz-plane. (b) yz-plane.

Figure 4.5: The ideal (dashed) and actual (solid) antenna radiation profile in xz-plane and yz-plane.

placed as VV.

We have not found, for DWM1001, any data which correlates the gain and the polar angle. However, in a document of a former antenna model called DW1000 [77], DecaWave gives the gain values (reported in Tab. 4.1) for an antenna vertically placed (as in Fig. 4.4(a)) in an anechoic chamber. We report these values just to confirm the not negligible impact of the elevation[2]: the 3D radiation pattern is far from being a sphere, with the same gain in all the directions.

Then, we conjecture that, when the MA is a drone, the 3D antenna pattern is highly irregular and it can be sketched as a nibbled apple (Fig. 4.5). Therefore, when such an antenna shape is projected on the ground, holes and bubbles can be found. We speculate that the altitude is the main cause of the pattern irregularity. So, we conjecture that when the two antennas are both on the ground (rover as MA) in VV configuration, the gain is the same in almost all the directions.

[2]Note also that the 3D antenna pattern is completely defined if we know its behavior in 3 planes: xz, xy, and yz.

4.3.3 Experiments for Antenna Radius

In our experiments, we wish to characterize the 2D antenna pattern observing the range of values of its radius. The MA starts at HOME in $(0, 0)$ at different altitudes $h = \{0, 10, 20\}$m. As explained in Sec. 4.2, the MA traverses the deployment area with n segments that aim to cross the receiving shape of the GD. Moving along each random segment, the MA continuously broadcasts its current (x, y) position, and the GD registers the first and the last heard endpoints sent by the MA. Since we know in advance the HOME position of the GD, we can compute the actual 2D radius for each detected endpoint. Note that we observe the radius on the ground. It is important to recall that the beacons are sent at regular intervals of time, and so the observed radii have an intrinsic error of at most I_w. In this Chapter, we fix I_w to 0.40m since we have experimentally observed this value, which clearly depends on the speed of the MA. From the collected endpoints during the same experiment, we compute the mean μ and the standard deviation σ of the set of observed radii. Then, we apply the goodness-of-fit method in order to assess whether a given distribution is suitable to the built data-set[3]. We repeat the experiments for the two antenna configurations (VV and VH) and for different altitudes.

We start reporting in Tab. 4.2 the results of the first experiment with the rover (i.e., $h = 0$m) and VV configuration. According to DecaWave's datasheet (solid line in Fig. 4.4(a)), we expect an almost uniform radius in the experiments, at least when the two antennas are at the same height. We observed 38 different endpoints, with mean $\mu = 97.10$m and $\sigma = 39.74$m, and also with min = 32.14m and max = 162.40m. According to the Pearson χ^2-test, at $h = 0$m, the radii of the VV configuration have a uniform distribution. As we will see, this is the only configuration with uniform distribution of the radii. Therefore, we agree with DecaWave that this configuration is somehow special. However, the radius cannot be considered really constant.

Then, in Tab. 4.3 we report a second experiment at altitude $h = 10$m, with VH configuration. We observed 28 different endpoints out of 38, thus confirming several null angles. The observed radii have mean $\mu = 66.34$m, $\sigma = 22.81$m, min = 16.52m, and max = 121.66m. According to the Pearson χ^2-test, at $h = 10$m, the radii of the VH configuration have a normal distribution.

Finally, in Tab. 4.4 we summarize the statistic distributions that fit the ob-

[3]Fixed a distribution and a set of categories, we determine if there is a significant difference between the expected and observed frequencies in one or more categories by using the chi-squared test.

Table 4.2: $h = 0$m, VV

class	radii (in m)		frequencies		
#	from	to	observed	uniform (U)	normal (N)
1	32.14	58.34	8	0.08	3.14
2	58.34	84.54	7	0.01	0.13
3	84.54	110.74	8	0.08	0.34
4	110.74	136.94	5	0.69	1.06
5	136.94	163.14	10	1.06	8.15
	likelihood			0.75	0.01

Table 4.3: $h = 10$m, VH

class	radii (in m)		frequencies		
#	from	to	observed	uniform (U)	normal (N)
1	16.52	41.52	5	1.68	0.68
2	41.52	66.52	9	0.00	0.14
3	66.52	91.52	11	0.52	0.07
4	91.52	141.52	3	12.23	0.15
	likelihood			0.01	0.90

served experimental radii. For each distribution, i.e., Uniform (U) and Normal (N), we give the observed μ and σ, and the likelihood. Except for the rover in VV, all the experiments show that the radius most likely follows a normal distribution, but with a large σ. It is worthy to note that increasing h, the mean of the radii decreases. The mean decreases faster with VV while with VH it remains quite stable (see Tab. 4.4). The values of the radii are generally more concentrate with VH than VV.

We conclude that, oppositely to our conjecture, VH seems better than VV and the radii obtained with a drone seem more concentrated than those obtained with a rover. Marginally, let us point out that organizing a localization mission is easier with a drone than with a rover because the drone is faster and less attention has to be paid to the terrain. Although the results are different from what we expected, we continue our investigation in localization algorithms accuracy. Thus, we use the results reported in Tab. 4.4 to generate a large synthetic set of endpoints that fit the estimated parameters of the radii distributions for testing the different algorithms surveyed in this Chapter.

From now on, we refer to the radius reported in Tab. 4.4 as the *observed radius* $r = \mu$, while to the DecaWave declared radius as the *manufacturer radius* $r = r_0$.

Table 4.4: The radii (in m) distribution with its parameters $D(\mu, \sigma)$ and its likelihood p.

	VV	VH
$h = 0$m	$U(97.10, 39.74); 0.75$	$N(63.58, 33.01); 0.40$
$h = 10$m	$N(84.97, 31.70); 0.81$	$N(66.34, 22.81); 0.90$
$h = 20$m	$N(62.91, 34.06); 0.83$	$N(57.69, 24.91); 0.82$

4.4 Range-free Comparative Evaluation

In this section, we compare all the RF localization algorithms using first the set of synthetic endpoints, and then the set of real endpoints collected during the experiments.

Our goal is to analyze the localization error and the percentage of unsuccessful localizations of DRF, XIAO, LEE, and DRFE. From now on, with height $h = \{0, 10, 20\}$m and antenna configurations { VV, VH } we refer to a particular scenario. For each simulated scenario, we run 200 localizations generating at random three endpoints, B_1, B_2, and B_3, with the distribution and the parameters of the simulated scenario derived in Sec. 4.3.3 from the observed endpoints (see Tab. 4.4). For each triple, we invoke the four algorithms using either the observed radius $r = \mu$ used to generate the endpoints or the manufacturer radius $r = r_0$. In the former case, we test the performance of algorithms when they receive in input the actual radius, but still, the endpoints can be affected by the antenna irregularity (i.e., σ). In the latter case, we test the performance of algorithms when they receive in input a completely different radius from the actual one.

Reinterpreting the constraints to improve the accuracy given in [80], the three selected endpoints, i.e., B_1, B_2, and B_3 we use for localizing the GD should satisfy two constraints: the minimum distance $r_{\min} = 60$m and the minimum angle $\alpha_{\min} = 20$deg between them. The constraint $r_{\min}$ means that the distances $d(B_1, B_2)$, $d(B_2, B_3)$, and $d(B_3, B_1)$, must be at least as long as $r_{\min}$. The $\alpha_{\min}$ constraint means that the three angles $\alpha_1 = \angle B_3 B_1 B_2$, $\alpha_2 = \angle B_1 B_2 B_3$, and $\alpha_3 = \angle B_2 B_3 B_1$, must be at least as large as $\alpha_{\min}$. From a geometrical point of view, these two constraints guarantee that the three selected endpoints are sufficiently apart each other, thus avoiding the construction of degenerated triangles. Therefore, in the experiments, we discard any triple of endpoints that does not satisfy $r_{\min} = 60$m and $\alpha_{\min} = 20$deg. We repeat the endpoint extraction until we find three suitable endpoints.

We compare the RF algorithms under two metrics; the *localization error*, de-

fined as the Euclidean distance between the actual GD's position and the estimated one outputted by the algorithms, and the *percentage of unlocalized*. Concerning the first metric, we report the localization error resumed into a boxplot that highlights the median (horizontal line), the average value (solid circle), and the data between the first Q_1 and the third quartile Q_3 (box). Additionally, the extremes of the whiskers represent the $Q_1 - 1.5$ IQR and $Q_3 + 1.5$ IQR, respectively, where the interquartile range (IQR) is defined as IQR $= Q_3 - Q_1$. Lastly, about the second metric, an unsuccessful localization is an application of the algorithm which does not return any constrained area or in general any geometrical intersection where the GD can reside, i.e., the GD remains unlocalized. This mainly happens for the RF radius-based algorithms when the radius is under-estimated.

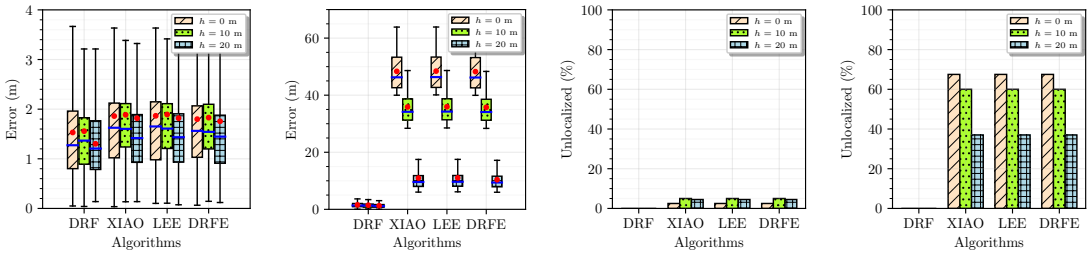
4.4.1 Algorithms Comparison Results

In this section, we evaluate the performance of the RF algorithms. We start considering a synthetic set of endpoints with an average radius equal to the observed radius μ, but very small standard deviation, emulating an ideal model. This experiment is to support the observation that a high accuracy is possible when the antenna is almost isotropic. Moreover, we discuss the performance of the RF algorithms on the synthetic set of endpoints generated according to the observed distributions in Tab. 4.4. These experiments, as those that use the manufacturer radius, show poor accuracy. In all the experiments, the impact of using a rover or a drone is considered. Finally, we report the comparison between the performance of DRF and DRFE.

Under the Ideal Model: Let us start considering the nearly ideal model in Fig. 4.6 in which the endpoints are generated with the observed radius μ given in Tab. 4.4 for the VV configuration, but selecting σ equal to 1. We evaluate the algorithms simulating an almost omnidirectional antenna. Since the dispersion is low, the radius can be considered almost constant. As expected, all the radius-based algorithms perform well when they use the observed radius $r = \mu$, which is the same radius used to generate the endpoints. In such a case, the localization error is on the order of a couple of meters or less for all the algorithms[4]. DRF is slightly better than the other algorithms at any altitude, while DRFE is worse than DRF. The number of unlocalized GDs is very small. However, the performance of the radius-based algorithms drastically drops down when the algorithms run using the

[4]Please note that the scale of y axis in Fig. 4.6(a) is zoomed with respect to the scale of y axis in Fig. 4.6(b)

manufacturer radius $r_0 = 60$m, while the endpoints have been generated with the observed radius. The percentage of unsuccessful localizations is extremely large.

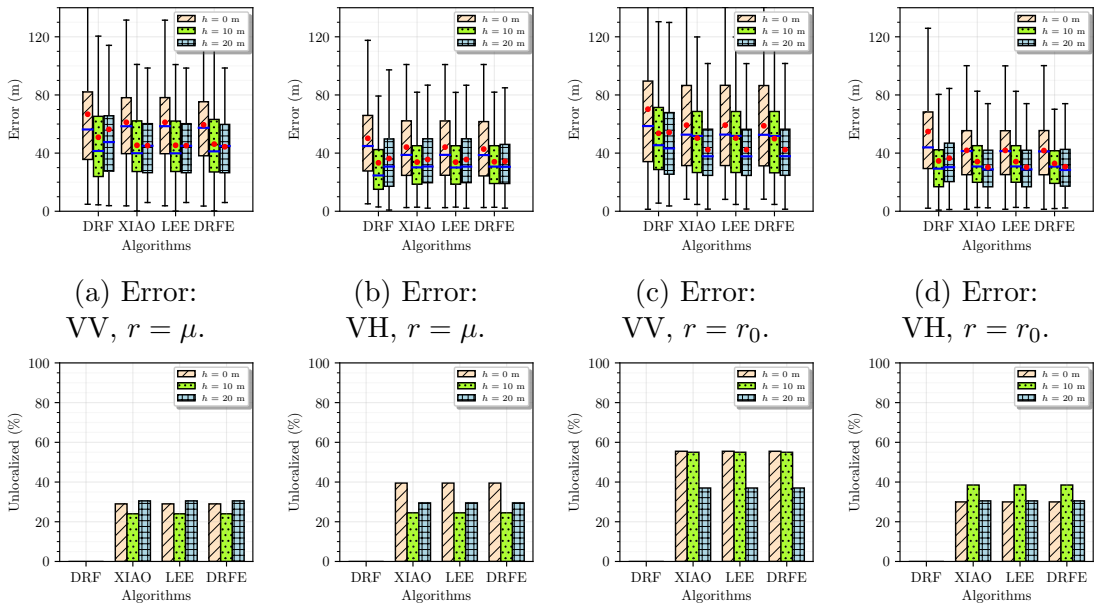


(a) Error: $\longrightarrow r = \mu$. (b) Error: $\longrightarrow r = r_0$. (c) Unlocalized: $\longrightarrow r = \mu$. (d) Unlocalized: $\longrightarrow r = r_0$.

Figure 4.6: The impact of the radius on the ideal model.

These results confirm our intuitions: to obtain an accurate localization, not only the antenna must be of good quality (i.e., with small σ), but also the observed radius μ must be exactly known by the algorithm. The results show that the error due to the use of the radius r_0 decreases when h increases because decreases the difference between the observed μ and the manufacturer r_0 radii. Alongside, note that since DRF is radius-free, the performance of DRF is not influenced by the radius selection, as shown in Fig. 4.6(b). DRF only requires an omnidirectional antenna.

Using the Synthetic Endpoints Set: Figs. 4.7(a), 4.7(b), 4.7(e), and 4.7(f) show the results when both the endpoints and the algorithms use the observed radius in Tab. 4.4. Figs. 4.7(c), 4.7(d), 4.7(g), and 4.7(h) instead show the results when the endpoints are generated using the observed radius and the algorithms run with the manufacturer radius. This is what happens in practice whenever the end-user implements the RF radius-based algorithms using the DecaWave datasheet radius on our test-bed. The errors of the algorithms are compared at different altitudes, antenna configurations, and radii. For all the algorithms, the average error is large. The worst error occurs with the VV configuration of the antennas and $h = 0$m: in such a scenario, the endpoints follow a uniform distribution. In general, the VH makes an error smaller than the VV probably because the radius that generated the endpoints for VH is less dispersed (i.e., σ is smaller) than that for VV. The three radius-based algorithms, XIAO, LEE, and DRFE, exhibit the same performance. They are inspired by slightly different ideas, but they actually act the same.



(a) Error: VV, $r = \mu$. (b) Error: VH, $r = \mu$. (c) Error: VV, $r = r_0$. (d) Error: VH, $r = r_0$.

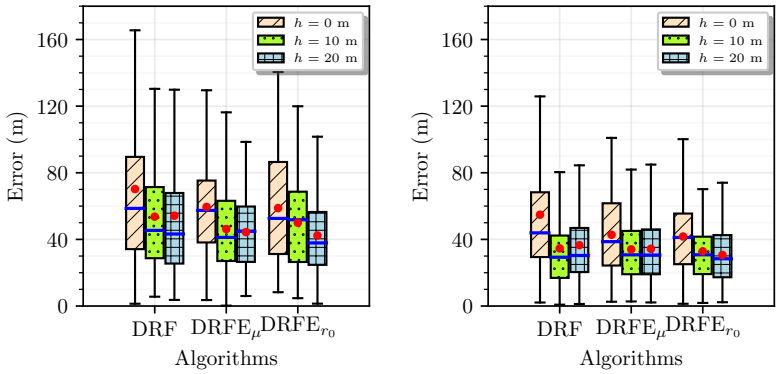
(e) Unlocalized: VV, $r = \mu$. (f) Unlocalized: VH, $r = \mu$. (g) Unlocalized: VV, $r = r_0$. (h) Unlocalized: VH, $r = r_0$.

Figure 4.7: Comparisons between all the algorithms in the studied scenarios.

The Drf algorithm experiences the worst average error, but the error is only slightly more than that of the radius-based algorithms. The whisker of the largest error of Drf is the longest whisker among all the algorithms probably because Drf finds a localization also in extreme cases when other algorithms return an unsuccessful localization. The error in Drf decreases when the altitude increases, moreover the error of the VV configuration is worse than that of the VH one.

The localization error is almost the same regardless of the adopted radius (observed or manufacturer). As witnessed by comparing the whiskers of the boxplots in Figs. 4.7(a) and 4.7(c) (resp., by Figs. 4.7(b) and 4.7(d)), the localization error of the radius-based algorithms, when they use the manufacturer radius r_0, is slightly worse than when they use the observed radius μ. Instead, the knowledge of μ reduces percentage of unlocalized GDs as illustrated in Figs. 4.7(e) and 4.7(g) for VV. The improvement in Figs. 4.7(f) and 4.7(h) is weaker for VH than for VV because for VH the difference between the observed radii $\mu = \{63.58, 66.34, 57.60\}$m and $r_0 = 60$m is smaller than VV.

In conclusion, the average localization error for VV and VH is 50m and 30m, respectively, regardless of if the algorithm knows the true radius used to generate the endpoints or not. The knowledge of the radius used to generate the endpoints



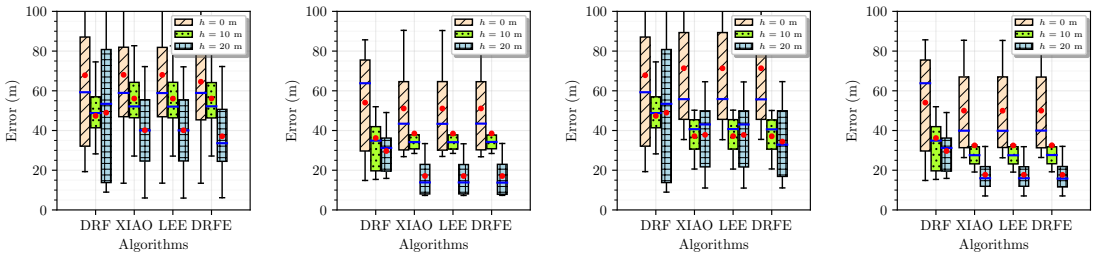
(a) Error: VV, DRF vs DRFE. (b) Error: VH, DRF vs DRFE.

Figure 4.8: DRF vs DRFE.

by the algorithm only matters when the radius dispersion is small. The take-away lesson here is that to apply a RF radius-based algorithm the antenna must be omnidirectional and its radius must be known by the algorithm.

Between DRF *and* DRFE: In Fig. 4.8, we compare the performance of DRF, DRFE$_\mu$ (DRFE with radius μ), and DRFE$_{r_0}$ (DRFE with radius r_0). In general, DRFE$_\mu$ performs better than DRFE$_{r_0}$, but DRFE$_\mu$ has always an error much greater than the error of the ideal model (see Fig. 4.6).

Using Real Endpoints: In Fig. 4.9, we show the results using the real endpoints collected during our test-bed. For the radius-based algorithms, in VV, the results of the localization error seem to follow the trend already seen on the synthetic generated set. In general, VV has a larger error compared to that of VH. In VH, the results are slightly better than that obtained on the synthetic generated set, although the number of valid triples is small. Indeed, the average error at



(a) Error: VV, μ. (b) Error: VH, μ. (c) Error: VV, r_0. (d) Error: VH, r_0.

Figure 4.9: Error using real data.

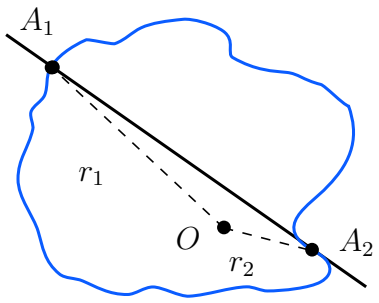


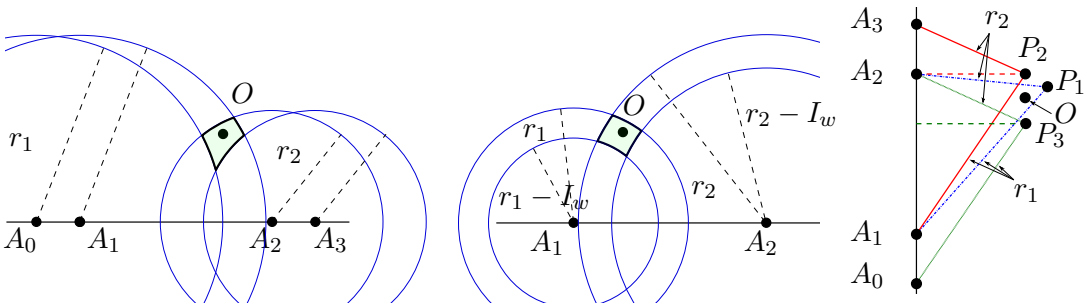
Figure 4.10: The two radii measure by the GD when the antenna pattern is irregular.

$h = 20$m is about 15m although σ is greater than 20m when the radius μ is used. The average error increases to 20m when the radius r_0 is used. This can be explained with the fact that there is a much stronger correlation between the endpoints that are not fully captured by the two constraints $r_{\min}$ and $\alpha_{\min}$ that we imposed for the selection of the synthetic endpoints. Finally, the error of DRF is worse with real endpoints than with synthetic endpoints, especially for the VV configuration.

So far, we learned the performance of all the RF algorithms strongly relies on the radiation antenna pattern. The simplest way to discover the radiation antenna pattern is to measure for each direction up to which distance the MA and the GD are one in the range of the other. So, in the next section, we significantly improve the localization accuracy exploiting the capability of the UWB antennas used in our test-bed of taking distance measurements. Even though there are bubbles and holes in the antenna pattern, it is always possible to discover them if the distance from the GD and the MA is taken (see Fig. 4.10).

4.5 Range-based Comparative Evaluation

Our interest in RF algorithms lay in the fact that they are simple to implement, do not require specialized hardware, and are scalable with respect to the number of GDs. Also, they are immune to problems that come from the measurement of the 3D distance. Indeed, distance measurements are affected by several errors that depend on the adopted technology and GPS. Technologies like WiFi or Bluetooth are much less accurate than UWB and may have measurement errors of the order of tens of meters, whereas GPS and barometer inaccuracies together with bad weather conditions can seriously impact the drone's position. Such combined 3D slant



(a) RB Xiao. (b) RB Lee. (c) RB DrfE.

Figure 4.11: The Xiao, Lee, and DrfE localization algorithms using two radii.

errors are then reflected on the ground, leading to 2D ground errors. Although range measurements come with their troubles, given the results of the previous section, we decided to include distance measurements in our algorithms. Since in our test-bed both the MA and GD are equipped with UWB antennas that are able to take distance measurements, we do not need to heavily modify our test-bed.

In the algorithm variants that we are going to propose, the actual measured distance between MA and GD is used to run the algorithms instead of using the observed or manufacturer radius. This adaption makes the RF algorithms actually RB algorithms, but we maintain the original algorithmic rules (e.g., intersection of annuli or circumferences). It is worth noting that we can apply modifications only to the RF algorithms that actually make use of the radius in their localization rules, and hence only for the radius-based ones, i.e., Xiao, Lee, and DrfE. Accordingly, the simplest radius-free algorithm Drf cannot be adapted to measurements.

Range-based version of Xiao: Fig. 4.11(a) shows how the Xiao algorithm is modified. When the GD hears for the first time the beacon in A_1, the GD measures the distance $r'_1 = d(A_1, O)$ between its position O and A_1. Whenever the MA is inside the receiving area of the GD, the GD still takes distance measurements neglecting the intermediate measurements. When the last beacon A_2 is heard, the GD saves the last distance $r'_2 = d(A_2, O)$ between its position O and A_2. As in the original version of Xiao, the GD computes the other two points A_0 and A_3 knowing the line that interconnects A_1 and A_2 and the value of I_w. Then, the algorithm proceeds as before, except that it draws two circumferences of radius $r_1 = r'_1 + \frac{I_w}{2}$ centered in A_0 and A_1, and two circumferences of radius $r_2 = r'_2 + \frac{I_w}{2}$ centered in A_2 and A_3. That is, the predefined observed or manufacturer radii are substituted by the distances from the endpoints and the GD. Note that the radius

r_1 and r_2 are used instead of $r'_1 = d(A_1, O)$ and $r'_2 = d(A_2, O)$ to avoid that O falls outside the intersection area in case of measurement error that underestimated the distances (see Fig. 4.11(a)).

Range-based version of LEE: As seen for XIAO, also for LEE, the two distances r'_1 and r'_2 between the endpoints and the GD are ranged. The algorithm proceeds by drawing two annuli centered at A_1 and A_2, with outer radius equal to $r_1 = r'_1 + \frac{I_w}{2}$ and $r_2 = r'_2 + \frac{I_w}{2}$, and inner radius equal to $r_1 - I_w$ and $r_2 - I_w$, respectively. The main difference with the previous version is that the radii are replaced by the distances between the GD and the endpoints. As in XIAO, we use the radii $r_1 = r'_1 + \frac{I_w}{2}$ and $r_2 = r'_2 + \frac{I_w}{2}$ instead of r'_1 or r'_2 to limit the risk that, due to measurement inaccuracy, O falls outside the intersection area. The width I_w of the annuli is preserved.

Range-based version of DRFE: We replace r with $r_1 = r'_1 = d(A_1, O)$ and $r_2 = r'_2 = d(A_2, O)$. The intersection of the two circles of radius r_1 and r_2 centered, respectively, at A_1 and A_2, returns P_1 which coincides with O (assuming no measurement errors). To repeat the construction of the RF DRFE, the points P_3 and P_2 are drawn in a way similarly to P_1. Precisely, P_3 is at the intersection between a circumference of radius r_1 centered in A_0 and a circumference of radius r_2 centered in A_2. P_2 is at the intersection between a circumference of radius r_1 centered in A_1 and a circumference of radius r_2 centered in A_3. As a result, differently from the original version where there were three isosceles triangles lying on the same line passing through A_0 and A_3, here there are three scalene triangles, i.e., $\triangle(A_1A_2P_1)$, $\triangle(A_1A_3P_2)$, and $\triangle(A_0A_2P_3)$. Eventually, the centroid resulting from the vertices P_1, P_2, and P_3 is selected as the estimation point, as shown in Fig. 4.11(c).

4.5.1 Error Analysis with Different Radii

In this section, we argue concerning the localization error that can be obtained once the original RF versions of the algorithms have been adapted to be actually RB. We focus on XIAO, but a very similar analysis applies to LEE. Some words will be finally spent for DRFE.

Without loss of generality, we assume that the MA moves along a straight line along the x-axis (see Fig. 4.11(a)). Let $O = (x_O, y_O)$ be the actual GD's location with respect to Cartesian coordinate system with origin in $A_0 = (0, 0)$, and let $P = (x_P, y_P)$ be the estimated GD's location by the algorithm. We fix A_1 and A_2

so as $r_1 = d(A_1, O) \leq d(A_2, O) = r_2$ and let $A_1 = (I_w, 0)$ and $A_2 = (kI_w, 0)$, where k is an integer number that represents the number of times the MA has sent the beacons from A_1 to A_2. Finally, let $A_3 = ((k+1)I_w, 0)$.

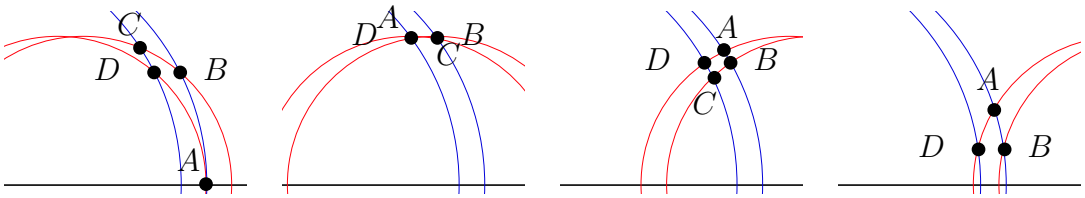
The RB XIAO relies on two different radii, i.e., r_1 applied to A_0 and A_1, and r_2 applied to A_2 and A_3, for constructing four circumferences, i.e., (a) $x^2 + y^2 > r_1^2$, centered in A_0, (b) $(x - I_w)^2 + y^2 \leq r_1^2$, centered in A_1, (c) $(x - kI_w)^2 + y^2 > r_2^2$, centered in A_2, and (d) $(x - (k+1)\, I_w)^2 + y^2 \leq r_2^2$, centered in A_3. We also denote as L_1 the *lune*[5] delimited by Eqs. (a)-(b), and as L_2 the lune delimited by Eqs. (c)-(d), also in accordance with the original XIAO algorithm.

Note that r_1 and r_2 are affected by the measurement errors [80] and for this reason we do not directly conclude that the estimated position P is at the intersection of Eqs. (b)-(c). Moreover, the quantity $kI_w = d(A_1, A_2)$ is influenced by the MA's speed, and hence it cannot be considered exact as well. Therefore, we prefer to select P inside the intersection among L_1 and L_2.

The correctness of the RB XIAO algorithm is completely different from that of the original XIAO. Although our implementation only memorizes the first and the last measured radius, i.e., r_1 and r_2, the RB XIAO algorithm performs measuring, at each i^{th} beacon $A_i = (iI_w, 0)$, the current distance $r_2(i) = d(A_i, O)$. Therefore, for each intermediate beacon A_i and corresponding measurement $r_2(i)$, with $1 \leq i \leq k$, RB XIAO knows that $O \in (L_1 \cap L_2(i))$, i.e., where $L_2(i)$ is the lune created by the two circumferences of radius $r_2(i)$ centered in A_i and A_{i+1}, respectively. Hence, although we do not implement this feature in our algorithm, after each measurement $r_2(i)$ RB XIAO could stop. As we will see, the best moment to stop would be when $r_2(i)$ is minimum. This is the main reason that makes RB XIAO robust to hole and bubbles: whenever it stops, the intersection area is limited. However, a discussion about the intersection of lunes while A_i varies along the x-axis is needed to complete the error analysis.

Assuming an omnidirectional antenna pattern, as MA moves along the x-axis, it crosses the GD's receiving area disk. The distances $d(A_i, O)$ decrease until MA reaches the closest position to O in $A_{k^*} = (k^* I_w, 0)$ (precisely, $x_O \leq k^* I_w < x_O + I_w$). Then, $d(A_{k^*}, O) \approx y_O$. After A_{k^*}, the distances $d(A_i, O)$ start to increase again up to $A_3 = ((k+1)I_w, 0)$, where the MA is no longer reachable from the GD, and whose last measured distance taken from $A_2 = (kI_w, 0)$ is $d(A_2, O) = r_2$. The observed distances while MA moves in A_i, with $i = 1, \ldots, k^*, \ldots, k$, form a unimodal sequence R with minimum in $k^* I_w$. As long as the radii in R decrease,

[5] In plane geometry, a lune is the concave-convex region bounded by two circular arcs.



(a) Tangent concordant. (b) Parallel to x-axis. (c) Tangent discordant. (d) When C is undefined.

Figure 4.12: The possible intersection areas depending on the lunes: the solid line is the x-axis.

i.e., $x_{A_i} < x_{A_{k^*}}$, the lunes L_1 and $L_2(i)$ have the same *curvature*[6] (see Fig. 4.12(a)), while when the radii in R increase, i.e., $x_{A_i} > x_{A_{k^*}}$, L_1 and $L_2(i)$ have opposite curvature (see Fig. 4.12(c)). In A_{k^*}, the tangent to the circumferences of the lune $L_2(k^*)$ is parallel to the x-axis (see Fig. 4.12(b)).

In a general scenario in presence of holes and bubbles, the MA stops at A_2 in one of the three cases depicted in Fig. 4.12: (i) the curvatures of A_1 and A_2 are concordant, i.e., $1 < k < k^*$ or $k^* < 1 < k$; (ii) the tangent at A_2 is parallel to x-axis, i.e., $1 < k = k^*$; (iii) the curvature A_1 and A_2 are discordant, i.e., $1 < k^* < k$. To evaluate the size of the intersection among L_1 and L_2 in these three cases, let observe that they cross each other making at most four intersections, i.e., A is the intersection among Eqs. (b)-(c), B among Eqs. (b)-(d), C among Eqs. (a)-(d), and D among Eqs. (a)-(c) (see Fig. 4.12), where:

$$A = \left(\frac{r_1^2 - r_2^2}{2\,(k-1)\,I_w} + \frac{k+1}{2} I_w, \sqrt{r_1^2 - \left(\frac{r_1^2 - r_2^2 + (k-1)^2\, I_w^2}{2\,(k-1)\,I_w} \right)^2} \right)$$

$$B = \left(\frac{r_1^2 - r_2^2}{2kI_w} + \frac{k+2}{2} I_w, \sqrt{r_1^2 - \left(\frac{r_1^2 - r_2^2 + k^2 I_w^2}{2kI_w} \right)^2} \right)$$

$$C = \left(\frac{r_1^2 - r_2^2}{2\,(k+1)\,I_w} + \frac{k+1}{2} I_w, \sqrt{r_1^2 - \left(\frac{r_1^2 - r_2^2 + (k+1)^2\, I_w^2}{2\,(k+1)\,I_w} \right)^2} \right)$$

$$D = \left(\frac{r_1^2 - r_2^2}{2kI_w} + \frac{k}{2} I_w, \sqrt{r_1^2 - \left(\frac{r_1^2 - r_2^2 + k^2 I_w^2}{2kI_w} \right)^2} \right)$$

[6]The curvature is the sign of the tangent to the curve.

Note that A is stable when A_2 moves because it is at the intersection of the two measurements of O. If there were no errors, $O \equiv A$. It is worth noting that $y_B = y_D$ and $x_B - x_D = I_w = d(D, B)$ regardless of k. Now we are in the position of clarifying the three previous cases. Recalling that $A_1 = (I_w, 0)$, depending on which is the last heard beacon $A_2 = (kI_w, 0)$, it may occurs:

Case (i): The lunes have the same curvature, as illustrated in Fig. 4.12(a). The area where P can be selected has width I_w and height $y_C - y_A$. In particular, if k is very small, $r_2 \to r_1$, and thus $y_C \to r_1$. If $y_A \to 0$, the error $y_C - y_A \approx r_1$. As k approaches k^*, $y_C - y_A$ quickly decreases. A similar error occurs when $k^* < 1 < k$.

Case (ii): By intersecting the coordinates $x_C = x_A$, and all the four points A, B, C, and D are very close. The area where P can be selected is very small.

Case (iii): Observe that x_C can be re-written as $x_C \approx (x_A + I_w)\ \frac{k-1}{k+1}$ for $k \geq 2$. Thus, $x_C < x_A + I_w$ for any k. Due to this and the fact that the curvatures of L_1 and L_2 are opposite, their intersection area is quite limited. Unfortunately, we could not find a simple formula for describing $|y_A - y_C|$ here, but it can be computed by approximating the curves with their tangent in A. Hence, the intersection of the two lunes can be inscribed in a rectangle with two sides parallel to the x-axis of length I_w and two vertical sides whose length depends on the angular coefficient of the tangents in A. We can only add that their lengths decreases below $2I_w$ when $\arctan \frac{y_A}{x_A} > 1$. So, the area where P can be selected only depends on I_w. When $\arctan \frac{y_A}{x_A} \leq 1$, the trivial bound is y_A, which cannot be very large however. So, the size of the area where P can be selected depends on I_w and y_A. When $r_1 \leq x_A < r_1 + I_w$, C is undefined (see Fig. 4.12(d)), Hence, the vertical side of the rectangle that contain P has length at most $y_A \leq \sqrt{r_1^2 - (r_1 - I_w)^2}$. This leads to the same error described in Xiao when the lunes have only 3 intersection points [70].

In presence of irregularities, RB Xiao, as in our implementation, stops at the last heard beacon. The error can be large only when A_1 and A_2 fall on the same side with respect to x_O, and thus the associated lunes have concordant curvatures. In order to limit the occurrences of Case (i), in our implementation we have forced B_1, B_2, and B_3, i.e., the three selected endpoints, to respect the $r_{\min}$ and $\alpha_{\min}$ constraints, putting them sufficiently apart, so as the two lunes will have opposite curvature. Moreover, in Cases (ii) and (iii) the error is quite limited and only depends on I_w. Without irregularities, RB Xiao always falls in Cases (ii) or (iii). The same happens for the original Xiao which stops when $r_1 = r_2$ and thus the curvatures are opposite.

To conclude, the above analysis also applies for Lee. In Lee, the GD belongs to

the intersection of two annuli of different radii but with the same width I_w, and the error depends on the curvature of the lunes created by the intersection of annuli. As regard to DRFE, P_1, P_2, and P_3 are the intersections of pair of circumferences, and here the lunes coincide with portions of circumferences. Differently from XIAO and LEE, the position uncertainty depends only on the measurement errors and on I_w, and it is bounded according to [80].

4.5.2 Range-based Versions Results

In this section, we evaluate the performance of the RB variants, and we compare these results with those of the RF original implementation.

In reality, in our previous experiments, since the goal was to investigate the performance of RF algorithms, we did not take any distance measurement even though our UWB antennas were able to range measurements with good accuracy. Indeed, we only knew in advance the GD's HOME position $(0, 0)$ and the MA's position with respect to HOME. Instead, the new RB variants rely on the 3D slant distances between the MA and the GD, which are then converted in 2D ground distances. However, it is important to recall that, due to the pandemic COVID-19, it is forbidden to take new distance measurements in the open field. According to those restrictions and based on our previous research experience, we decided to estimate the 2D ground distances (see Fig. 4.3(a)) between the MA and the GD just calculating the Euclidean distance from the endpoints to the HOME in $(0, 0)$ plus a random error (overestimation or underestimation) computed as proposed in [80][7]. In other words, we perturb any Euclidean distance r adding the error E_r:

$$E_r = \gamma_d + \frac{h}{r}\gamma_h + e_s \cdot \sqrt{1 + \frac{h^2}{r^2}} = 1.2 + \frac{h}{r}0.2 + e_s \cdot \sqrt{1 + \frac{h^2}{r^2}} \tag{4.1}$$

where h is the drone's altitude, $\gamma_d = 1.2$m is the drone's rolling error, $\gamma_h = 0.2$ is the drone's altitude error, and e_s is a random number in the range $[-\epsilon_s, +\epsilon_s]$, where $\epsilon_s = 0.1$m is the UWB instrumental accuracy [80]. From Eq. (4.1), when the ground distance $r = 30$m, the maximum ground error E_r is approximately 1.3m, 1.4m, and 1.5m, for h is 0m, 10m, and 20m, respectively. First, we compare the accuracy of the new RB variant algorithms toward that of their RF implementation. Then, we make remarks on the accuracy of the new RB variants of the three RB algorithms.

[7]Our decision is supported by our previous experience in converting slant from/to ground measurements [80, 81].

Table 4.5: Error (in m) and unlocalized (in %) between RF and RB algorithms in VV.

		RF				RB	
		$r = \mu$		$r = r_0$		$r = d(A_i, O)$	
Algorithm	h	error	unloc	error	unloc	error	unloc
Xiao	0m	68	30%	72	55%	4	0%
	10m	56	25%	38	54%	4	0%
	20m	40	31%	39	37%	3	0%
Lee	0m	67	30%	73	55%	3.8	0%
	10m	55	25%	38	53%	3.8	0%
	20m	40	32%	38	37%	2.7	0%
DrfE	0m	64	29%	74	55%	3.2	0%
	10m	55	26%	37	53%	3.2	0%
	20m	37	31%	33	36%	2.2	0%

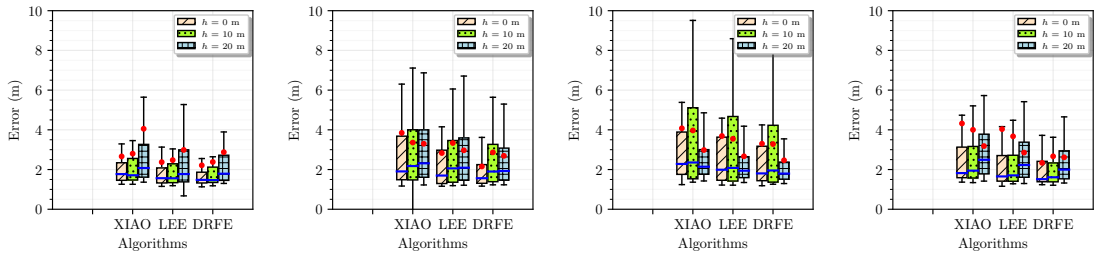
Range-free vs Range-based: In Tab. 4.5 we report the localization accuracy of Xiao, Lee, and DrfE showing the comparison between the original RF and adapted RB versions of them. Tab. 4.5 also reports the number of times (in %) that an algorithm does not localize the GD. The experimental results take into account the constraints $r_{\min} = 60$m and $\alpha_{\min} = 20$deg, and only VV. In particular, for the original RF versions we report the average error for both the radii, i.e., the observed radius $r = \mu$ and the manufacturer radius $r = r_0$. For the adapted RB versions we report the average error for the measured radius, denoted as $r = d(A_i, O)$, where A_i is an endpoint.

Clearly, the average error, using measurements, is one order of magnitude smaller. We have seen that the knowledge of the exact distance significantly improves the accuracy and practically clear the number of unlocalized devices. This also enforces our original statement that the knowledge of the radiation pattern is fundamental for the success of the localization. Bubbles and holes in the antenna radiation pattern are no longer a problem because we range the effective distances between the endpoints and the GD. So to avoid measurements, any manufacturer should give as much information as possible on the antenna radiation pattern.

Range-based Comparison: Here, we present performance-wise the RB versions of the Xiao, Lee, and DrfE algorithms. As previously said, Drf is not present because it is not radius-based. For this simulation evaluation, we consider different cases varying the constraints of minimum distance $r_{\min} = \{0, 60\}$m and minimum angle $\alpha_{\min} = \{20, 40\}$deg, as explained in Sec. 4.4.

In Fig. 4.13 and Fig. 4.14, we evaluate the performance of the RB algorithms. In particular, in Fig. 4.13 we compare the algorithms fixing the minimum angle $\alpha_{\min} = 20\text{deg}$, while in Fig. 4.14 such constraint is kept to $\alpha_{\min} = 40\text{deg}$. Overall, we can see that DrfE works better than the other two algorithms with an average error smaller across all the tests, and that Xiao works better than Lee with those settings. Also, a better localization accuracy can be obtained for the experiments carried on the ground at $h = 0\text{m}$, exception made for Fig. 4.13(c) in which, oddly, the error is smaller at $h = 20\text{m}$. More specifically, in both Fig. 4.13 and Fig. 4.14 is also evident that VH benefits slightly more from the $r_{\min}$ constraint of 60m, while it is the opposite for VV which behaves better showing smaller error with no $r_{\min}$ constraint. Finally, there is very little improvement while testing with larger $\alpha_{\min}$. It is extremely important to note that using measurements, all the selected endpoints are suitable for localizing the GD in O. This means that the number of unlocalized GDs is zero, but also this leads to the growth of the outliers number. This can be seen by the presence of long whiskers and a considerable difference between the average (i.e., solid circle) and median (i.e., horizontal line) values. However, even though such whiskers are pretty long, they are really short with respect to the ones seen in the previous experiments in Sec. 4.4, Fig. 4.9.

Concluding this Chapter, we compared the performance of four RF algorithms based on HnH on a real test-bed using the DecaWave DWM1001 UWB antennas as MA and GDs. We implemented and simulated the algorithms on a large data-set of endpoints collected in the field. We analyzed the antenna radiation pattern of the GD at different altitudes and configurations, i.e., VV and VH. We have shown how such algorithms actually perform assuming (i) first the datasheet radius of the antenna equal to that released by the manufacturer, (ii) then the



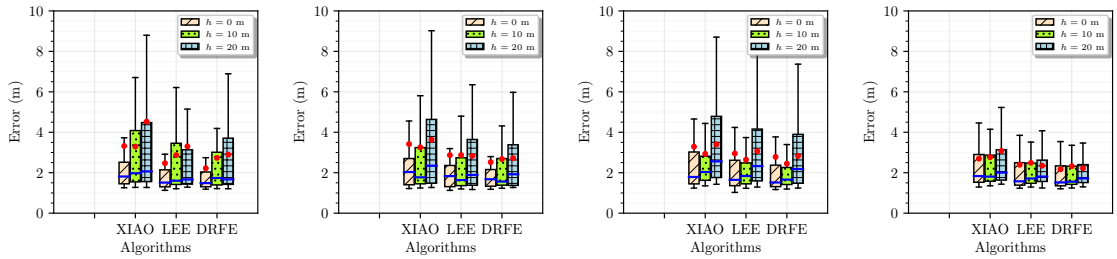
(a) Error: VV, $r_{\min} = 0\text{m}$.

(b) Error: VH, $r_{\min} = 0\text{m}$.

(c) Error: VV, $r_{\min} = 60\text{m}$.

(d) Error: VH, $r_{\min} = 60\text{m}$.

Figure 4.13: Error using distance measurements when $\alpha_{\min} = 20\text{deg}$ and $r_{\min}$ varies.



(a) Error: VV, $r_{\min} = 0$m.

(b) Error: VH, $r_{\min} = 0$m.

(c) Error: VV, $r_{\min} = 60$m.

(d) Error: VH, $r_{\min} = 60$m.

Figure 4.14: Error using distance measurements when $\alpha_{\min} = 40$deg and $r_{\min}$ varies.

experimental observed radius, and (iii) finally the actual radius obtained via distance measurements. The manufacturer datasheet radius poorly performs because usually it does not characterize the antenna radiation pattern very well. The observed radius can help only if it is almost constant in all the directions. When the antenna is irregular, only the knowledge of the distances between the MA and the GD can alleviate the localization error. We conclude that the RF algorithms are simple and elegant, but they can be very inaccurate and can leave a high percentage of unlocalized GDs if the antenna is not omnidirectional and measurements are not allowed. However, the exact knowledge of the irregular antenna can make accurate the RF algorithms.

Part III

Security and Safety for and from Drones

In Part I and II, drones' potential is discussed for several civilian applications, like delivery systems and search and rescue missions. In all these activities the most critical aspect is accurately flying through certain positions. Even a small error in terms of *UAV positioning* made by the pilot or due to the navigation satellite system when harvesting fruits or cleaning windows can nullify the use of UAVs, causing damage to the surroundings or even harming people. The problem is that the positioning can also be altered by external users for malicious purposes performing specific attacks, such as the Global Navigation Satellite System (GNSS) *spoofing* which focuses on hijacking UAVs, or even by *hacking* the private ad-hoc local network between the drone and its controller, in case of a UAV guided by a human pilot. Under such attacks, an authorized UAV cannot perform anymore the assigned tasks safely because the path it is following could be tampered by the attackers, therefore becoming an *unauthorized* and *compromised* UAV.

For instance, *unauthorized* drones may operate in restricted or crowded areas such as airports and stadiums, where they can collide with airplanes or land/crash on people. Moreover, most Commercial-off-the-Shelf (COTS) drones are equipped with cameras to enable First-Person View (FPV), which allows real-time video transmission from the drone's camera back to the controller (or any separate viewing device). Intuitively, this capability can easily lead to privacy violations even though it was meant for recreational purposes and monitoring tasks [82, 83].

Thus, compromised or unauthorized UAVs pose serious security and privacy concerns as reported for example in [84].

In response to such issues, the Federal Aviation Administration (FAA - which is the aviation's regulatory body for the United States Department of Transportation - USDoT) has released the final set of policies on the 15th of January, 2021, that governs the identification requirements for drones which is known as Remote ID [85]. These policies require any drone that operates in the National Air Space (NAS) to continuously broadcasts identifying information such as a unique identifier, a timestamp, drone velocity, latitude, longitude, and altitude of both the drone and its controller. However, it is difficult to enforce the compliance of such policies on the myriad of commercial drones that are pushed to the market every day, and therefore an identification solution of non-compliant rogue drones is still needed. Many drone detection solutions exists [86, 87], but mostly remain prohibitively expensive for the majority of citizens and some of the solutions target only specific drone models [86]. These issues motivated a surge of research efforts whose objective is to provide cost-effective solutions for detecting unauthorized drones entering restricted airspace or private areas. Toward this goal, many recent

contributions (*e.g.*, [88, 89]) attempt to detect drones by analyzing their network traffic patterns, and in chapter 6 a new and very effective system is presented.

The rest of this Part is organized as follows: in Chapter 5 the most recent attacks against UAVs are studied. We surveyed the literature, proposing our *taxonomy* based on the UAV's flight mode and then on the number of UAVs deployed. Basically, UAVs are easily subjected to different kinds of attacks, especially cheap and affordable models available on the market. Some of them use an unauthenticated connection with the controller, meaning that everyone can "watch" the data stream, stealing information or the control of the UAV itself. More advanced UAVs require more sophisticated attacks (i.e., spoofing), mostly against the navigation system in use. Fortunately, the more UAVs deployed, the more complex becomes the attack that an attacker must carry on. The conclusions from our study, exhibit the fact that to increase safety for UAVs, the most prominent way is to increase the number of UAVs, therefore creating a fleet of them, and adding some auxiliary technology able to help the UAV in case of attack, rendering the whole attack process almost impossible to achieve.

The last Chapter 6 presents a *WiFi-based detection system* for the detection of drones that enter a specific area such as airports, a stadium, or crowded places in general. We define those drones as unauthorized drones because authorized drones should have been declared in advance, therefore their presence should be known. Thus, unauthorized drones are either drones controlled by an attacker (as shown in Chapter 5, attackers can steal drones and use them for malicious purposes) or simply drones controlled by users unaware of the rules enforced in the detection area. Our system scans the WiFi connections in search of the data stream sent by drones, especially First Person View (FPV) data recorded by the drone's camera. The proposed detection system is a machine learning-based system, so we start by collecting data from different sources that we assume are transmitting simultaneously in the restricted area, such as: mobile phones while using the network in general, VoIP applications, IoT cameras transmissions, and of course drones data transmissions. VoIP and IoT cameras' data are especially important because their data stream is similar to an FPV stream, therefore we pose emphasis on the ability to discern correctly those data streams from drones data stream. Furthermore, we discovered a specific feature from which we computed our unique set of features, able to detect with high accuracy the drone traffic, even and especially, drones traffic not included in the dataset.

Chapter 5

UAVs Path Deviation Attacks: Survey and Research Challenges[1]

In this Chapter, we concentrate on the threats that can prevent single or multiple UAVs to accomplish their path and/or reach the destination. We primarily focus on commercial UAVs, available and affordable for public and small companies that have small sizes, limited battery lifetime, moderate payload capacity, and specific constraints in terms of computational power. Despite their limited cost, such drones can be equipped with several external sensors (cameras, antennas) that facilitate communication with the external world, and single-board computers that extend their computational capability.

Since we are interested in attacks that limit the flying ability of UAVs and their faculty of following predefined paths, we organize a *taxonomy* that groups the attacks based on three different flight modes, i.e., First Person View (FPV), Global Navigation Satellite System based (GNSS), and an enhanced GNSS, dubbed GNSS "plus" (GNSS+). Then, we also sub-group the works based on the multiplicity of UAVs (i.e., Single or Multiple).

Organization: The rest of the Chapter is organized as follows. Sec. 5.1 reviews recent surveys on security aspects of UAVs. Sec. 5.2 introduces our taxonomy of the literature about path deviation attacks. Sec. 5.3 presents the defenses according to the proposed attack taxonomy. Finally, Sec. 5.4 explores future research directions.

[1]This work has been published to IAUV 2020 - 2nd International Workshop on Internet of Autonomous Unmanned Vehicles ⟶ Publications n. [7]

5.1 Related Surveys in the Recent Past

Recently, due to the increasing interest in UAVs, several surveys on UAVs have been published in the literature that, however, do not cover our focus.

In [90], the authors provide a comprehensive study on UAV cellular communications. Authors ponder on UAVs that can be used as flying Base Stations (BSs) extending and enhancing the user's capabilities especially for crowded areas where there is a lack of coverage. They survey the issues, challenges, and opportunities of using UAVs for cellular communications with a little focus on safety and cyber physical security aspects.

The authors in [91] survey UAV networks from a Cyber Physical System (CPS) perspective dealing with security and privacy concerns on possible UAV networks. Also, in [92] security concerns are addressed in networked UAVs presenting Intrusion Detection Systems (IDSs).

In [93], the authors categorize the UAVs attacks according to three different attacks, i.e., *physical* on unattended and unsecured UAVs, *remote* exploiting back doors, bugs, or other UAV software vulnerabilities, and *target* aimed to a specific UAV's sensor, for example GPS or cameras. Physical attacks are conduced by an intruder who finds the UAV (or its ground control) left unattended and unsecured. Given physical access to the UAV, the attacker can inject malicious code on the UAV for further attacks and transform the attack in a remote one. Target attacks are for a specific UAV's sensor, such as GPS or cameras. The conclusion in that survey is that the first two categories of threats (physical and remote) are well investigated, while the last category (target) has been ignored.

Authors in [50] present a cyber attack/challenges taxonomy in UAV systems based on three challenges, i.e., *confidentiality* that can be broken by malware injection or hijacking, *integrity* that focuses on modifying information of UAV systems, and *availability* that can be performed with the goal of flooding the communications links.

Authors in [94] study the impact of Global Positioning System (GPS) spoofing attacks on different settings, like power distribution networks, ships, aircraft, trucks, train, and mobile phones. It does not consider UAVs, though.

They conclude that spoofing is a nascent threat not yet trivially doable, but with high potential to be severe if low-cost GPS spoofers will become common and the economical value of the applications that use GPS continues to grow.

This last survey is on GPS spoofing and it is the one closest to our focus.

5.2 The proposed taxonomy

In this section we introduce a taxonomy of the pertinent literature. In Sec. 5.2.1 and 5.2.2, we sketch the UAV's flight modes and introduce the multiplicities.

In Sec. 5.2.3, we describe the background on the attacks that deviate the UAV from its path.

5.2.1 UAV's Flight Modes

The first taxonomy level represent the UAV's flight modes used to group the attacks. We distinguish between three different UAV's flight modes:

- **FPV**: UAVs are directly moved by a pilot's either eyesight or through a screen monitor using the remote controller.
- **GNSS**: UAVs autonomously fly by retrieving the GNSS signals. They also rely on the Inertial Measurement Unit (IMU) which uses various sensors to control the flight.
- **GNSS+**: GNSS mode can be enhanced by adopting other technologies, such as using images, WiFi, or cellular networks, to improve the flight robustness. GNSS+ could be also GNSS enriched by a perception system.

5.2.2 UAV's Multiplicity

In the second taxonomy level, we consider the multiplicity i.e., the number of UAVs that can take part together in a task:

- **Single**: only a single UAV is deployed for a task.
- **Multiple**: many UAVs are deployed for a common task.

Note that we consider any number of UAVs in FPV as a set of single UAV because a pilot is required for each UAV. We use the multiplicity just to group the defenses.

5.2.3 UAV's Attacks

We now describe the most significant attacks to deviate the UAV's path based on the flight mode.

Attacks in FPV: In FPV flight mode, in-flight attacks can be substantially performed by hacking the local controller/UAV WiFi network. Nowadays, UAVs rely on WiFi links to which a remote controller, cameras, gimbals, and also other smart devices like smartphones or tablets, can connect. Many manufacturers design proprietary protocols for those wireless communications, which in turn can be highly more powerful than common wireless protocols in terms of performance. The weakness of proprietary protocols is, however, their scarce ability to quickly respond to security threats [95], letting the UAVs more susceptible to attacks of hackers.

Attacks in GNSS/GNSS+: The GNSS and GNSS+ flight modes are based on the multiple interoperable constellations of satellites like GPS, GLONASS, BeiDou, Galileo, and more [96, 97]. To better understand the threats behind this flight mode, we describe how the GPS works [98]. The GPS uses a certain number of satellite transmitters S_i located at known locations L_i^S on their orbits. Each transmitter is equipped with a synchronized clock and broadcasts at a certain time t a carefully chosen navigation signal $s_i(t)$, which includes the spreading code, the phase, the satellite ID, timestamps, and information on the satellites' deviation from the predicted trajectories. The signal propagates with light speed c. The GPS receiver V receives at time t the combined signal $g(L,t)$ of all the signals earlier broadcast by the satellites in range: $g(L,t) = \sum_i A_i s_i \left(t - \|L_i^S - L\|/c\right) + n(L,t)$, where A_i is the signal attenuation, $s_i(t - \|L_i^S - L\|/c)$ is the signal broadcast by S_i at time $t - \|L_i^S - L\|/c$, and $n(L,t)$ is the background noise. Note that, the signal s_i has been encoded at the time the signal started from the satellite. That is, the time t the signal was received by the receiver minus the time required to traverse the Euclidean distance $d_i = \|L_i^S - L\|$ between S_i and V at the light speed c. Due to the properties of the signal coding, the receiver can extract from $g(L,t)$ all the single information $s_i(t)$. From there, the time delay $\|L_i^S - L\|/c$ is retrieved, and the *ranges* d_i inferred. In reality, the receiver which has a time offset δ infers the *pseudo-ranges* $r_i = d_i + \delta c$ because the clock of the receiver is not fully synchronized with that of the satellites. With d_i rewritten as the distance $\sqrt{(x - x_i^S)^2 + (y - y_i^S)^2 + (z - z_i^S)^2}$ between the receiver position $L = (x, y, z) \in \mathbb{R}^3$ and the satellite position $L_i^S = (x_i^S, y_i^S, z_i^S)$, L and the offset $\Delta = c \cdot \delta$ can

be determined by solving a system with at least four pseudo-ranges. We say a receiver has a *lock* to a specific transmitter when it is already receiving data from that satellite. So, the receiver needs at least four locks to localize itself.

GPS receivers are vulnerable to different attacks which depend on how the attacker modifies the satellites signal. The attacker can easily deceive the receiver at any location L' if it has full control over the input of such antenna. On civilian GPS receivers, the attacker is free to choose its position, the delay, and the satellite position.

The attacker can claim a fake satellite location because civilian GPS signals are not authenticated. Given the right hardware, anyone can transmit his own GPS signals and their data content are not checked for plausibility or consistency. On military GPS receivers, instead, the attacker cannot change the data content of the GPS messages.

Common attacks on every GNSS are blocking, jamming, and spoofing. GNSS jamming and blocking are techniques which create strong interference when detecting the GNSS signal with the goal of preventing the lock of the receiver to a satellite [99]. GNSS spoofing focuses on deceiving the legitimate GNSS signal and hence to corrupt the pseudo-ranges. A GNSS spoofing attack requires the attacker to make the victim stop receiving signals from the legitimate satellites and start receiving the attacker's signals. GNSS spoofing can either be classified as *hard* or *soft* spoofing attack [100]. The hard attack consists in the transmission of a very powerful GNSS signal, that can be considered as jamming attack at first, forcing the receiver to drop the original signal in favor of the fake one. Once the victim has locked the new spoofed GNSS signal, the attacker can easily modify the GNSS signal as desired. More complex is the soft attack, due to the fact that the signal strength of the spoofed signal has to be coherent to the legitimate one. Once the victim locks into the new spoofed signal, the attacker can smoothly increase the strength of the spoofed signal, and finally move the target accordingly.

Meaconing is the simplest form of GNSS spoofing based on interception and rebroadcast of GNSS signals. It is a kind of hard spoofing. The fake signals are rebroadcast after a delay, typically, with power higher than the original one, to confuse victims [94]. The Lift-of-Align (LoA) attack sets the spoofing signal perfectly aligned to the legitimate signal, and then gradually increases its power so as the receiver will select the fake signal. LoA is a kind of soft spoofing [101].

We are not aware of specific attacks for GNSS+. A specific attack for GNSS+ should involve both GNSS and the additional guidance system used. We instead found some defenses that exploit the redundancy in the guidance systems.

5.3 Defenses by Flight Mode and Multiplicity

In this section, we discuss the defenses, if any, reported in the literature for the attacks illustrated before. We organize the surveyed papers as reported in Fig. 5.1.

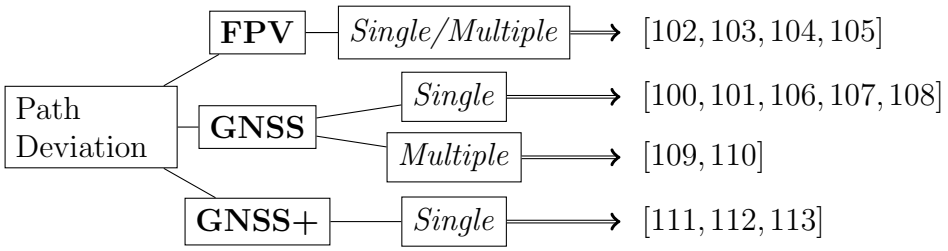


Figure 5.1: The proposed taxonomy of the inherent literature

5.3.1 FPV Mode

In FPV mode the UAV is guided by a pilot. The UAV can be controlled eyesight Line-of-Sight (LoS) or remotely controlled through a camera mounted on the UAV and a built-in screen monitor on the controller. The only available option for an attacker is directly taking control of the UAV by bypassing the connection between the UAV and the controller. We found evidence in literature that attacks to this mode are feasible, while almost no defenses were found, as listed below.

Single/Multiple UAVs

A work in [102] propose a system able to take control of criminal UAVs exploiting a radio frequency spoofing technique. The system emulates the original signals from the original controller. The downsize is that the UAV's model must be known beforehand to be able to emulate the signal correctly. The paper does not explain whether it is easy to extend the attack to other drone models.

Authors in [103] test the weakness of Parrot AR Drone 2.0. As other UAVs, this model runs a Linux kernel and uses an open WiFi that leads to different attacks. Acquiring full access to the UAV over the unencrypted WiFi network is possible and not hard. The UAV creates an open hotspot, and because no password is required, authors were able to connect to it and scan its ports. Telnet port grants

access to root shell with no password, and whereby to the entire UAV's operating system.

Authors of [104] show vulnerabilities able to alter the WiFi controller/UAV connection. For this task, they select two UAVs: again the Parrot AR Drone 2.0 and Cheerson CX-10W Drone. Those are toy-like UAVs sold everywhere, widely used, and ideal for various applications. Attacks like Denial of Service (DoS), deauthentication, man-in-the-middle, and packet spoofing are tested using very cheap hardware. Results show that all of those attacks were successful.

Similarly, authors in [105] investigate security vulnerabilities associated with two more complex UAVs, i.e., DJI Phantom 4 Pro and Parrot Bebop 2. While Phantom 4 Pro is mainly susceptible to interception and jamming of the controller radio signal, Bebop 2 uses an open WiFi connection making it weak against various wireless attacks (e.g., deauthentication attack, main-in-the-middle, etc). Therefore both can be easily hijacked. According to the authors, security improvement like WPA/WPA2 for encrypting the WiFi password may cause decrease in range and transmission speed.

In conclusion, the main issue disclosed in the FPV mode analysis is the lack of an encrypted WiFi connection. This means that not only the access to the WiFi is visible and open to everyone (because the password is not required) but also that all the network traffic between UAV and controller can be sniffed by any user employing simple techniques and using very affordable hardware. So far, defenses in this field have not attracted great interest perhaps because the FPV UAVs have been mainly used for recreational purposes. However, this flaw might be very dangerous. There are many applications whose importance is increasing like trellis maintenance, windows cleaning, and such, in which an hijacked FPV UAV can do catastrophic damages. Therefore, the first defense is to use encrypted WiFi connections as long as they can be supported in terms of energy and computational power by the UAV's controller [102, 105]. Investigating the scenario of exchanging info between FPV UAV and Internet of Things (IoT) devices on the ground, seems worthy too.

5.3.2 GNSS Mode

In this section, we consider the defenses for preventing spoofing.

Single UAV

In the literature, GNSS attacks mainly focus on different targets of UAVs. However, UAVs can incur on the same attacks because their GNSS receivers are standard. The only paper we are aware to conduct a GNSS spoofing attack to UAVs is [100]. The authors perform GNSS spoofing for a *safe hijacking attack* to move away a (terrorist) UAV from protected areas instead of incapacitating it. They consider Phantom DJIs, Parrot Bebop 2, and 3DR Solo UAVs. Simple hard spoofing is enough for hijacking DJIs and Parrot, while for 3DR, based on ArduPilot, soft spoofing is required. They successfully conduct the attack to all the three drones. As for the attacks, there are not ad-hoc defenses for single UAV. Proposed defenses for other targets can be used for UAVs as long as they do not demand specific hardware or high computational burden. As said, not all the defenses can be adapted. An interesting case of countermeasure that cannot be moved to UAVs is that in [98]. Such countermeasure is based on the use of multiple receivers installed on the same target with a certain spacing. The authors exemplify their countermeasure on a cargo ship in which there is enough space to put several receivers apart one from the others. Clearly, such defense is unfeasible on a single small UAV.

GNSS Signal Property Technique: We report here a set of papers that leverage the incoherence of the received counterfeit GNSS signals to defenses.

The authors in [106] introduce and evaluate the Time Synchronization Attack Rejection and Mitigation (TSARM) technique for time synchronization attacks over common GNSS receivers. This technique works in two steps: first, a dynamic model is introduced which analytically detects and models the GNSS attacks in the receiver's clock bias and drift, and second, the spoofed signatures, i.e., the estimated clock bias and drift, are accordingly modified based on the estimated attack. So, the attacked GNSS receiver can ignore the inconsistent signatures and proceed with its normal operation applying adequate time corrections on the application.

Authors in [107] exploit Signal Quality Monitoring (SQM) techniques for detecting spoofing attacks. Originally designed for multipath detection, SQM-based methods have good feasibility in a simpler context such as the duel between spoof-

ing and legitimate signals. The authors improve the SQM technique proposing a new system that works by analyzing the initial interaction between spoofing and legitimate signals when the SQM metric fluctuates significantly. Once both the fake and legitimate signals reach the victim, the SQM fluctuation is empathized to gain the lock. Such fluctuation causes an abnormality quantity of output values which allows taking adequate countermeasures.

These papers are based on sophisticated detection schemes which, however, may become quickly obsolete because powerful attackers may use advanced techniques (e.g., based on reinforcement algorithms, machine learning) for breaking the defenses.

IMU: The IMU-based defenses are countermeasures that exploit the redundancy of available knowledge at the UAV. IMU essentially grants another pool of information to which turn under dubious conditions. The measurements from the IMU are fed to the Inertial Navigation System (INS) that helps the UAV to stay on track. It works using Kalman Filters (KFs) which elaborate a series of observed measurements over time to estimate unknown variables. Usually, KFs estimate the position and velocity errors, gyroscope bias, random noise errors, and accelerometer bias error.

The authors in [108] propose a technique based on the KF innovation sequence (i.e., the output prediction error) for a robust spoofing detection method. The weakness of traditional KF methods is that they validate the correlation between the measurements and the output prediction errors only in specific snapshots. In this way, KFs are able to detect only rough GNSS spoofing attacks, i.e., when the measurement drift introduced by the spoofer is very large. Knowing this, an attacker can smoothly spoof a victim. The core idea behind the defense in that paper, is to use the aforementioned KFs on a longer time window testing all the possible accumulated errors along the time and not only those on a specific snapshot. The authors simulate GNSS spoofing attacks and prove the effectiveness of their detection method.

Similarly, in [101] authors study the effects on the KF of two different spoofing attacks: meaconing and LoA attacks. The authors propose a detection method of the fake signal based on IMU error compensations obtained by applying the KF. Results show that in the presence of meaconing and LoA attacks, the compensation of errors greatly increases, probably due to route instability (i.e., small forced movements) induced from the spoofing signal. This instability creates inconsistency between IMU readings and expected values calculated from the GNSS

signal, that leads to worthy compensations by KFs. In conclusion, the authors claim that their proposed system works well when detecting both the attacks.

Multiple UAVs

This section presents papers that propose countermeasures when multiple UAVs either preserve an arrangement during the flight or not.

In [109], authors investigate a scenario in which an attacker with multiple antennas (for multiple spoofed signal) attacks a fleet of UAVs preserving the fleet arrangement. First of all note that the attack can be only conducted using multiple antennas [98] because a single attacker can spoof any number of receivers in its transmission range, but they can be shifted only around the same coordinates L'. Thus, the attacks to a fleet must be conducted using multiple physical antennas. Nonetheless, as demonstrated in [98, 109], there are a bounded number of physical positions from which the attack can be launched. Generally, such attacker locations belong to a quadratic space, called hyperboloid, constrained by the pseudo-ranges between attacker and receivers and by the pseudo-ranges between claimed satellite and the two shifted positions. Not only, increasing the number of victims, the attacker locations belong to intersection of hyperboloids, i.e., the possible locations are no longer squared space, but just set of points. When the fleet has two UAVs, the possible locations are a set of hyperboloids. When the fleet has three UAVs, the possible locations are a set of intersection of hyperboloids. When there are four UAVs, the possible locations are a set of two points; and from five or more UAVs, the possible locations are a set of single points. Therefore, it is particularly difficult to launch such an attack to arranged multiple drones.

Authors in [110] propose defenses against multiple UAVs spoofing, not arranged. The defense is based on cooperative localization. Indeed, whenever a UAV doubts the authenticity of GNSS signal, it can invoke a cooperative localization with other three non-collinear UAVs in the fleet. However, cooperative localization mechanism could not be reliable because other UAVs could be under attack. To overcome this limitation, authors assume that only a single UAV at a time can be attacked.

From these results, we can state that the deployment of an arranged fleet of UAVs (that preserves the relative distances), instead of a single UAV, is enough to increase the defenses against GNSS spoofing attack. But also just flying multiple UAVs together, without any kind of arrangement, offers opportunities of thwarting the attack as illustrated in [110]. Multiplicity is a resource to be carefully exploited.

However, one should also evaluate the overhead in accomplishing a complex task with multiple UAVs that cooperate for the same goal.

5.3.3 GNSS+ Mode

Another simple but effective way to thwart spoofing attacks is to integrate a perception system in the UAV, as it happens in GNSS+. We are not aware of papers that deal with attacks that simultaneously threat the perception system and the UAV. So, GNSS+ is more robust than GNSS. The extra modules added to the UAVs are to be considered as a defense in themselves.

Single UAV

A commonly used additional technology on-board on UAVs is the combination of devices formed by gimbal and camera. Gimbal allows the fixed camera to move along two or three axes enhancing the flight experience.

Authors in [111] propose a vision-based approach using a camera. Basically, they combine together the velocity registered from the IMU and the calculated instant velocity from the Lucas-Kanade (LK) algorithm. The result is further compared with the GNSS velocity, and in case of discrepancies, a possible GNSS spoofing is detected. The experiments are carried on a DJI Phantom 4, and the authors claim to be able to detect an attack, on average, between 5 seconds.

A camera for improving defenses against attacks is used in [112]. Authors substantially repeat the same approach as [111], but focusing on the UAV's path instead of the UAV's velocity. From the camera images, using the Visual Odometry (VO) technique, the relative path is extracted and compared with the GNSS path. If the two paths do not fit each other, then the UAV is considered under a spoofing attack. To evaluate the similarities between the two measures, the authors propose different metrics that could detect spoofing attacks.

The system in [113] is based on Angle of Arrival (AoA) integrity-monitoring method. AoA method works on the assumption that all GNSS signals arrive from different positions and angles, while signals from a spoofer come all from the same direction. The proposed solution adopts the generalized likelihood ratio test and an optimal decision-making algorithm is built to detect the presence of attacks using a three-antenna array (with a maximum distance between the antennas of half signal wavelength). In a situation of possible attack in which the GNSS position cannot be trusted and thus considered unknown, results show that the

detection is possible if the power level of the spoofed signals exceeds the power level of legitimate signals.

From these results, we can state that to flank GNSS receivers with other systems that help to control the route, as GNSS+ does, improves the system robustness. Attack-oriented solutions that simultaneously compromise the systems (GNSS, perception) involved in GNSS+ should be investigated.

5.4 Research Challenges and Future Works

We conclude this work by reporting some lacks that attract our attention. First of all, almost no defenses have been proposed for UAVs flying in FPV mode. As the economical interest in first person drone activities increases, defenses should be designed.

Spoofing attacks still seem to be relatively easy for single UAVs based on civilian GNSS receivers. With the growth of UAV's applications and being easy to build GNSS spoofers, GNSS spoofing attacks can become more attractive and profitable.

Several scheme to detect spoofing that operate on the check of signal properties have been devised, but their effectiveness can be reduced by highly capable attackers that carefully generate GNSS counterfeit signals to avoid triggering these detection schemes. We believe that the most effective way to improve resilience against spoofing attacks for single UAV is to strengthen the GNSS+ mode using different flanked perception systems. Very interesting could be the integration of the 5G technology to UAVs, that will be soon available. 5G antennas will be an almost ready-to-use crowd-sourcing mechanism to cross-check the GNSS signal in the surroundings.

Ad-hoc defenses for special UAV's tasks should also be investigated. For example, in the case of a UAV patrolling an area (e.g., repeating the same route constantly), the success rate of an attack is higher due to the UAV predictability. A defense mechanism based on multiple vision solutions that exploits the history of past patrols could be ideal.

Also, we have not found any paper that considers security on delivery with UAVs. This is an area where applications daily bloom and attacks, like hijacking, could be extremely profitable. Several algorithms for delivery have been proposed that integrate trucks and drones: this coupled delivery system could offer opportunities also for ensuring the delivery process.

We have also learned that it is substantially much more difficult to conduct a spoofing attack to a fleet of UAVs that preserves an arrangement than to a single UAV. However, accurately preserving the arrangement of multiple UAVs during the flight is not always possible. So, it could be interesting to verify up to inaccuracy of the arrangement (i.e., relative distances among the drones), the attack is thwarted. Also attacks for multiple drones that fly together without using GNSS system, which are now a hot-topic, need attention.

Finally, attack and defense are two faces of the same coin; whenever new defensive mechanisms are introduced, new attacks can be carried on. Therefore, some future research topics could be attack-oriented to show attacks on GNSS+ mode or on special drone applications.

In conclusion, we reported the state-of-the-art of path deviation attacks against different UAV's flight modes (FVP, GNSS, and GNSS+). We learned that very little has been done to make secure the FPV flight mode. In GNSS mode, spoofing is the most worrisome attack that can hijack the UAV.

The strong defenses seem to come from redundant devices that assist the UAV in the flight.

An interesting future research direction is to consider ad-hoc solutions for thwarting attacks in emergent UAVs applications, such as delivering. UAV fleet that fly without GNSS is a new emergent trend not investigated at the moment.

Chapter 6

Unauthorized Drones Detection Using Video Streaming Characteristics[1]

In this Chapter, we propose a novel and cost-effective drone detection framework (using a WiFi adapter and a laptop/desktop) that solves the problem of detecting new drone types among other similar streaming devices based solely on the drone's FPV highly-dynamic video stream.

The majority of COTS drones employ WiFi chips that allow the drone to act as WiFi Access Point (AP). The user can install an app in a smartphone/tablet to connect to the AP and establish a bidirectional data stream to receive the live-streaming FPV video and to control the drone. Most prior work on drone detection uses the FPV video stream and analyzes traffic patterns from a statistical perspective. For instance, in [88] the authors presented a framework to detect drones' FPV streams based on channel use. That approach proved challenging as video bit rates can widely vary in response to the changing scenery that is captured by the drone's camera which triggers the video compression algorithm to add/remove video frames from the stream [88, 114]. The authors in [115] use the Received Signal Strength Indicator (RSSI) as a feature to discern moving vs. stationary devices. However, this approach may fail to differentiate drones from other moving radio sources. In [89], the authors presented a framework that use features extracted from WiFi frames exchanged between the drone and its controllers. Sim-

[1]Part of this work has been published to ICDCN 2021 - the 22nd International Conference on Distributed Computing and Networking, and the full work has been submitted to ACM Transactions on Cyber-Physical Systems journal ⟶ Publications n. [8, 9]

ilar to the previously mentioned paper, the approach is not tested with changing bit rates emitted over the FPV channel and may be unable to differentiate drones' FPV streams from other WiFi video streams such as IoT cameras [114]. Furthermore, the framework requires both the drone and its controller to be within the detection range. In [116], the authors trained a machine learning model to detect drones with high accuracy using a training set composed of drone and controller traffic. Despite using controller traffic in the dataset, the authors left the problem of the "recognition of new UAV types" and "modified video patterns" as open problems.

The framework accomplishes the detection task by leveraging specific packets we identified in the encrypted FPV stream sent by drones over the WiFi channel. We refer to these specific packets, which appear in video streams at periodic intervals for synchronization purposes, as "pivots".

The framework consists of two components: the Pivot Extraction Algorithm and the Classification Model (Figure 6.1). The Pivot Extraction Algorithm identifies the pivot packets for each encountered WiFi device based on FPV video traffic patterns, while the Classification Model extracts machine learning features based on the pivot packets and trains a drone classifier.

The framework produces high drone detection accuracy under challenging settings, including (i) the scene being captured by the drone's camera is highly dynamic which produces varying bit-rates, (ii) only the drone is within the packet capture range while its controller is out of capture range, (iii) the network traffic of the drone is completely encrypted, (iv) there are other IoT devices in the environment actively emitting video streams that exhibit an FPV packet pattern similar to those generated by drones, and (v) the classifier encounters and detects drones that were not used during the training phase of the classifier.

Contributions:

Under the aforementioned challenges, we summarize our contributions as follows:

- We introduce the concept of *Pivot*; a special packet found in video streams (which is used as an anchor for designing drone features) that is not affected by video parameters such as the bitrate. Furthermore, we introduce the notion of *Pivot Fingerprint*; a series of packets that most likely contains a pivot packet, and an approach for locating such fingerprints within a stream of packets.

- We design a novel feature selection strategy that is executed in two phases. The first phase is *model-independent* and based on the Jaccard Similarity Index which that selects drone features with high discriminative power to detect new unprofiled drones regardless of the machine learning model used or any unique drone behaviors that exist in a training set. The second phase is *model-dependent* and based on Recursive Feature Elimination (RFE) algorithm that evaluates the sensitivity of the remaining features toward detecting unprofiled drones and selects the best set of features for the underlying machine learning model used in the detection framework.

- We provide an implementation and evaluation of the detection framework based on a Random Forest algorithm using features created from pivot packets that can detect drones with accuracy up to 99% even when the drones coexist with other video streaming devices such as IoT cameras and VoIP applications. We also compare our pivot features with the state-of-the-art machine learning-based drone detection features proposed by [116].

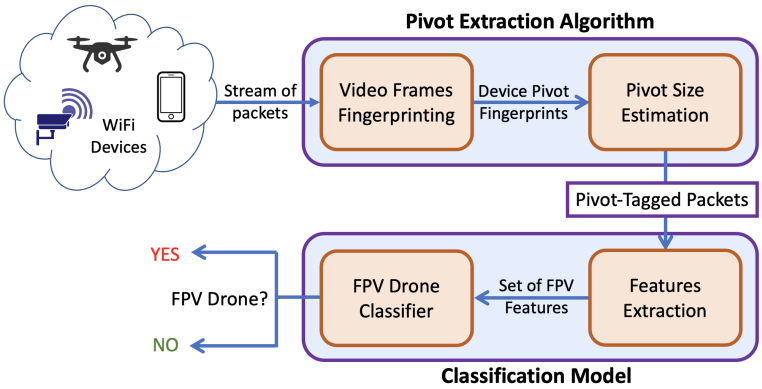


Figure 6.1: Components of the Drone Detection Framework

Organization: The rest of the Chapter is structured as follows: in Section 6.1 we highlight the related work on the field of drone detection. In Section 6.2, we discuss the attacker and defender model, then we provide a brief background on the characteristics of the uplink and downlink channels between the drone and its controller. In Section 6.3 we introduce our drone detection framework and its two components, then we discuss in details the characteristics of the pivot packets and the design of the first component; the Pivot Extraction Algorithm. We also assess the algorithm's behavior when encountered with benign video traffic such as IoT cameras. Section 6.4 shifts the discussion towards the framework's second components; the Classification Model, where we describe our novel pivot features creation and selection strategies and our FPV drone classifier. In Section 6.5, we outline the implementation design of the framework in terms of hardware and software, then we describe our data collection, processing, and sampling procedures. The evaluation and results analysis of the proposed framework is reported in Section 6.6.

6.1 Related Work

Drones detection is a research topic which is capturing considerable attention from the research community. Mainly, drone detection techniques fall under four different categories: (i) Radar-Based Detection [117]: based on active radars that send electromagnetic pulses toward the area to be monitored to detect the electromagnetic energy reflected by any flying objects. (ii) Vision-Based Detection: based on computer vision devices (such as regular or Thermal cameras) to detect flying objects [118, 119]. (iii) Acoustic Detection: that detects the high-pitched sound frequency generated by the rotating propellers of the drone [120, 121]. Lastly (iv) RF/WiFi Detection techniques that monitor the wireless communication links between the drone and its controller (the Ground Control Station, or GCS) [122] [88]. Since in this Chapter we proposes a new WiFi detection technique, we focus on related works on RF/WiFi detection of drones. We also summarize some previous results on detection of WiFi cameras due to the similarities between the streams transmitted by the drones and those transmitted by fixed WiFi cameras.

Detection of Drone Traffic: The authors of [88] propose to detect drones by their FPV streams. The bit rate of a FPV stream emitted by a drone is compared with the bit rate of a well known and previously recorded FPV data streams. However, the more the scene changes, the higher the bit rate is, and a drone recording a highly dynamic scene might not always match a specific set of FPV bit rate. The authors also argue that since the drones move, the RSSI of their FPV channels is different from those of other WiFi video streaming services. However, they did not take into account video streaming devices that might be moving such as VoIP applications on smartphones.

In [89], drones are detected by monitoring their FPV stream sent over WiFi and applying Machine Learning models. They extracted features from WiFi frames exchanged between the drone and its controllers. As in [88], the authors do not consider either the problem of detecting drones FPV streams with changing bit rates or that of differentiating between drones FPV streams and other WiFi video streams. The same authors, in [123], took the previous work a step further by improving the robustness of the algorithm. Precisely, they aim to detect a drone flying in stealth mode (i.e., a drone that is not transmitting video).

In [124], a framework is proposed to fingerprint drone WiFi communications for the identification of specific drone models. Firstly, drones are discerned by other devices via their speed (exploiting the signal strength information used to calculate the acceleration). Once a WiFi session is associated with a drone, it is further classified using different features (i.e., pattern of probe messages and information in a Frame Header). However, this detection is based on the assumption that other devices cannot move at the same speed as the drone. In some scenarios, a device could be inside a vehicle (i.e., a phone inside a car) that moves at comparable or higher speeds than of a drone.

In [125,126], standard classification algorithms are used on eavesdropped traffic exchanged between a drone and its remote controller to analyze extracted features such as packets' inter-arrival times and sizes. The main aim is to detect a drone and its status, i.e., flying vs. resting.

In [116], Alipour et al. use classical features as those used in [125, 126] for differentiating FPV drones from other devices, and hence for detecting drones. The authors show that their framework detects drones using features extracted from packet samples of at least 50 packets exchanged between the drone and controller. The authors left the problems of "recognition of new UAV types" and "modified video patterns" as open problems. The aim of this Chapter it to improve the results in [116] and to close the problem of recognizing new UAV types.

Detection of WiFi Cameras: In [114], the authors detect hidden wireless cameras using smartphones. Their system, DeWiCam, automatically analyzes wireless traffic to recognize camera transmissions, specifically relying on physical and MAC layer features. Importantly, camera traffic streams have relatively stable volume and packet size pattern. Besides, wireless cameras can work continuously without interruptions in the stream.

In [127], the main objective is to identify hidden WiFi cameras by altering the ambient light captured by the cameras to induce variations in the camera's packet flow (caused by the video compression algorithm) which can be identified by statistical techniques. Both papers exploit the compression mechanism that cameras use for video transmissions, discussed below in Section 6.2. In this Chapter, we are also interested in investigating characteristics of video transmissions, with the different objective of detecting drones in challenging scenarios.

6.2 Drone's Network Model - Background & Assumptions

In this Section, we first discuss the *attacker and defender* model and its underlying assumptions. Then we provide a brief background on the characteristics of the communication channels between the drone and its controller. Specifically, Section 6.2.2 describes the control channel (uplink) and its components, while Section 6.2.3 discusses the FPV channel (downlink) and how a video compression algorithm affects the behavior of a network traffic pattern.

6.2.1 Attacker and Defender Model

In the defender model (Figure 6.2), we consider a restricted airspace protected by a monitoring station. The station's objective is to detect unauthorized drone access and is comprised of a general purpose computer such as a laptop or a desktop, and a WiFi adapter that captures all WiFi packets that are sent within its vicinity. For each stream of packets, the station analyzes the stream and labels it as either 'drone' or 'non-drone' based on drone FPV signatures. We also assume the existence of other video streaming devices that can generate false-alarms within the station's receiving range such as IoT cameras.

In the attacker model, we consider an unauthorized FPV drone attempting to access the restricted airspace. When the drone operates, a bi-directional connec-

tion is established between the drone and its controller. The downlink channel is a video stream sent by the drone to the controller, while the uplink channel carries control commands sent from the controller to the drone. We assume that both channels are encrypted and the drone's controller is out of the monitoring station's detection range.

6.2.2 Control Channel:

The control channel carries flight commands from the controller to the drone. Overall, the control channel comprises of two different network flows, periodic control packets and heartbeats packets. In case of controllers that receive a video stream over TCP, a flow of acknowledgments (ACKs) will be sent out back to the drone as well. Note that even if ACKs are encrypted, they can be easily identified by the packet size (20 bytes TCP header without options + 20 bytes IP header without payload). Some control applications embed heartbeat messages within the stream of periodic control packets. Both streams (heartbeats and control packets) have rather unique sizes and inter-arrival times. Therefore, a simple detection system can easily differentiate between the controller's traffic and other background traffic. In our framework, we reflect a more realistic scenario where we assume that controllers are not always present within the range of the detection system and their traffic cannot be captured and classified (Figure 6.2).

6.2.3 FPV Channel & Video Encoding:

The FPV channel transports a compressed video from the drone to the controller. A video is a stream or a sequence of still pictures. In order to decrease the portion of channel capacity used to transport the video over wireless channels,

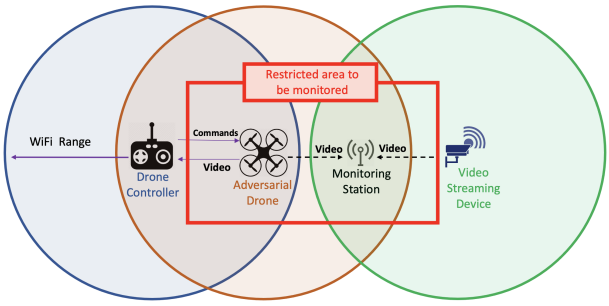


Figure 6.2: Attacker and Defender Model

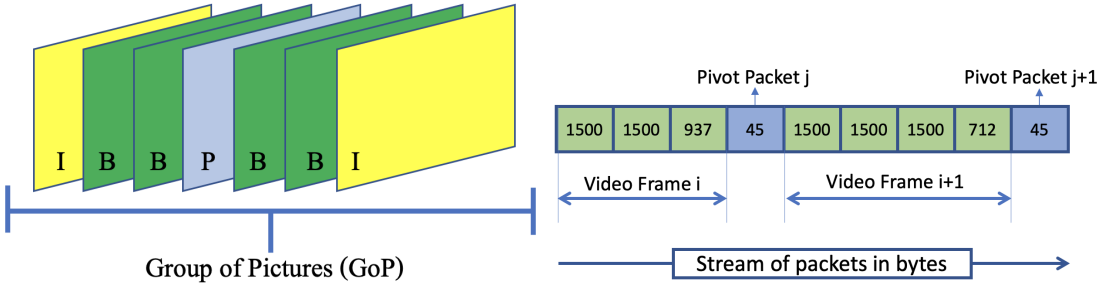
almost all video encoders include a video compression algorithm.

Typically, compression algorithms exploit the - possibly high - correlation along the spatial and temporal dimensions within Group of Pictures (GoP). In brief, while JPEG compression is used in the spatial domain, the core idea to harness temporal correlation is to encode differences compared to reference frames within each GoP. We, then, have three frame types: Intra-Coded Frames (I-Frames), Predicted Frames (P-Frames), and Bi-directional Frames (B-Frames). I-Frames are Intra-coded frames that exploit spatial redundancy (correlation among the pixels in the frame) to achieve compression at individual frame-level. P-Frames and B-Frames are Inter-coded frames that exploit temporal redundancy prediction. The difference between P- and B-Frames is that P-Frames only encode pixels that are changed compared to the last reference frame, while B-Frame considers both previous and future reference frames. These frames are grouped into a single GoP, which starts and ends with an I-Frame (Figure 6.3(a)). Typically, video streaming applications rely on well-known video codecs such as H.264 to compress raw images captured by some camera and turn them into compressed frames for faster transmission. In general, B-Frames (which includes futures changes in the frame) are used in prerecorded videos (such as YouTube) but are not used in real-time streaming applications (such as FPV video streaming) to minimize the video delay, on the expense of lowering the video quality or increasing the occupied bandwidth.

Intuitively, scene characteristics and the motion of the drone heavily influence the compression rate achieved by the scheme described above, as well as the temporal structure of the data stream. In other words, if the captured scene is dynamic, then, the encoding scheme will either (i) generate I-Frames more frequently, which in turn results in shorter GoPs or larger P-/B-Frames, or (ii) generate larger differential frames. In both cases, the resulting compression gain is reduced.

The keynote of a video stream transmitted over a WiFi network is that the size of these frames is larger than a WiFi MTU (Maximum Transmission Unit) which is 2304 bytes. Network interfaces translate all packets sent and received from the operating system into Ethernet, which has an MTU of 1500 bytes. Therefore, each individual video frame that is larger than the MTU is fragmented into a series of packets of a size equal to the maximum Ethernet MTU size, except for the last packet, which contains the residual of the frame.

FPV video streams also include other messages such as synchronization information for maintaining the state of the channel. Typically, these messages are small and fit a single WiFi packet and are sent periodically at predicted intervals.



(a) GoP starts and ends with an I-Frame (b) A 45-bytes pivot packet buffered between two video frames packets

Figure 6.3: A single video frame is sent over multiple packets while pivots are contained in a single packet

Packets from the compressed video stream are emitted more frequently compared to sync packets, which then, have a *higher probability* to be buffered right after the last less-than-MTU sized frame packet (Figure 6.3(b)).

The resulting pattern contains rather unique packet sequences. The fixed-size of the sync packets, make them suitable to act as **pivots** which can be used to build FPV drone features for a detection model. The characteristics of pivot packets will be discussed in Section 6.3.1.

6.2.4 Discussion:

From Section 6.2.2 we can infer that the packet pattern generated over the control channel is quite unique: short periodic packets. Although this approach can lead to robust detectors [89, 123, 125, 128], the source of those packets is the controller, which might be out of the range of the monitoring station (Figure 6.2). On the other hand, as we have discussed in Section 6.2.3, the FPV channel is susceptible to the frame variations in the captured scene and therefore approaches focused on the FPV channel analyze statistical properties of this data stream. Specifically, they look at the timing and size of I- and P-Frames [88,114]. However, drones generate data streams that are statistically similar to those of other IoT devices emitting video streams. Moreover, those characteristics widely vary as environmental and trajectory parameters change (e.g., compression gain). We contend, thus, that many existing approaches would fail to provide robust detection in many relevant conditions. Therefore, to overcome the previous inadequacies, the framework we propose is based on traffic emitted over the FPV channel and its objective is to detect the drone even when the controller is out of the framework

detection range and uses features robust to a wide spectrum of scenarios and parameters. Most importantly, the framework should detect new drone types without the need of having a prior profile of the encountered drone.

6.3 Drone Detection Framework

In this Section we describe our *Drone Detection Framework* (Figure 6.1), which is based on the concept of a *pivot*; a special packet found in the drone's FPV traffic which will be utilized as an anchor for building drone features. The Drone Detection Framework is the composition of two main components: a *Pivot Extraction Algorithm* (Section 6.3.3) and a *Classification Model* (Section 6.4). In short, the Pivot Extraction Algorithm takes a sequence of packets as an input, tracks the packets that belong to video frames, extracts pivot fingerprints, and estimates the size of the pivot packet for each encountered WiFi device. The engine of the Classification Model uses the pivot size to construct a set of FPV features and feeds it to a binary classifier which outputs "YES" if the packets belong to an FPV drone and "NO" otherwise.

This Section mainly focuses on the first component of the framework, Pivot Extraction Algorithm, while the second component will be discussed later in Section 6.4. In this Section, subsection 6.3.1 will discuss what does a pivot packet look like via network traffic analysis. Subsection 6.3.2 describes how pivot packets appear with respect to other normal packets. In subsection 6.3.3 we show how to locate and extract pivot packets from a network trace using our Pivot Extraction Algorithm via tracking video frames that form pivot fingerprints, while subsection 6.3.4 illustrates how the algorithm differentiates between FPV video streams and other benign streams such as IoT cameras.

6.3.1 Pivot Definition via FPV Traffic Analysis

To define pivot packets, we analyze data collected from three FPV drones from three different brands: EACHINE E58 WiFi FPV Quadcopter, Spacekey DC 014 FPV WiFi Drone, and Ryze Tello Quadcopter Drone. In the following, the three drones are referred to (according to their painted color) as Black (BLK), Red (RED), and White (WHT) drone, respectively (Figure 6.4(a)).

Each drone has a built-in camera with a First-Person-View (FPV) capability. Upon powering up, the drone creates an Access Point (AP) and the user connects his/her smartphone/tablet to that AP. Each drone has its own app that can be

(a) FPV Drones (b) IoT Cameras

Figure 6.4: The FPV drones and IoT cameras used in the pivot analysis

downloaded from Google Play or Apple's App Store. When the user connects to the drone's AP, a video stream can be initiated and viewed on the smartphone through the drone's designated app.

For every one of the three drones, we capture decrypted network traffic (for easier initial analysis) of the drone's FPV using an external WiFi adapter set into Monitor Mode from five meters away for 30 seconds. We consider two different video states: the first state corresponds to the drone's camera capturing a stable scene (STB), while the second state corresponds to a highly unstable (shaking) drone flight (SHK) where the camera rapidly points at different directions. The latter state induces fast changes in the captured video stream which would potentially affects the frame-rate of the video compression algorithm and ultimately the video bit-rate as explained earlier in Section 6.2.3.

Analyzing the BLK drone's traffic, we observe that before transmitting a series of full-sized packets (which we assume is a video frame), the drone transmits a 46-bytes packet. This packet includes an ASCII-readable command called *lewei_cmd*, which appears to be responsible for transferring media files from the drone to its associated app on the controlling smartphone [129]. The app associated with the BLK drone also sends some *lewei_cmd* packets to the drone, but only a few of them (once every second). The RED drone, instead, sends 4-bytes packets (with identical payloads) to its RED-app right before sending a video frame. The RED-app sends the same 4-bytes packet to the drone, but only four times in the entire 30 seconds network trace. Finally, the WHT drone sends an identical 35-bytes packet (except for the last 2 bytes in the payload) to its WHT-app between some

video frames. These 35-bytes packets are sent at uniform intervals (every 0.1 of a second), which is independent of the varying FPV frames transmission times. For all of the three drones, traffic patterns were observed for both SHK and STB video states.

From these observations we conclude that each drone periodically sends special packets that are repetitive and identical in length, and are not part of a video frame payload. In our drone detection framework, we will leverage these packets and we name them the *pivot* packets that will be used to create features enabling drone detection.

6.3.2 Patterns & Behavior of Pivot Packets

We now study the occurrence and patterns of pivot packets in relation to the two different video states: SHK and STB. In both states of each drone, we compute the average bit-rate and the pivot appearance rate (pivot per second). The results are reported in Table 6.1. It can be observed that when the video state shifts from STB to SHK, the FPV bit-rate of the BLK drone on average increases by 55 kbps (25%), while the number of pivots observed per second increases by 2 (25%). In the RED drone, instead, the FPV bit-rate increases by 4 kbps (2%), while the number of pivots observed per second increases by 2 (13%). In the WHT drone, although the FPV bit-rate increases by 42 kbps (13%), the number of observed pivots remains almost constant.

Although the collected data from the BLK drone seems to indicate a dependence of the number of pivots on the FPV bit-rate (both increased by almost the same percentage), the patterns observed in the RED and WHT drones streams instead supports the hypothesis that the pivots are not necessarily tied to the bit-rate (especially for the WHT drone since the inter-arrivals of the pivot packets are constant) and the number of pivot packets are almost the same for both the video states, that is, pivots are independent of the bit-rate increase. The deviation in the number of pivots in the other two drones might be due to (i) the nonuniform inter-arrivals of pivots and (ii) the inherent packet loss in the wireless environment for packets captured via WiFi Monitor Mode, which can affect the small number of transmitted pivots. To this end, we can safely assume that the pivots are not necessarily tied to the bit-rate (i.e., number of video frames) of the FPV stream and, as such, pivots can be used to create robust features for drone detection regardless of the video's motion state.

Table 6.1: Pivot frequency & bit-rate change

Drone	FPV Bit-Rate	Pivot/Second	Δ FPV Bit-Rate	Δ Pivot/Second
BLK-SHK	269.567 kbps	9.080	55.437 kbps	≈ 2
BLK-STB	214.130 kbps	7.308		
RED-SHK	215.342 kbps	18.379	4.128 kbps	≈ 2
RED-STB	211.214 kbps	16.200		
WHT-SHK	360.968 kbps	7.701	42.097 kbps	≈ 0
WHT-STB	318.871 kbps	7.760		

6.3.3 Pivot Extraction Algorithm

The Pivot Extraction Algorithm is the first component of the Drone Detection Framework, and its objective is to extract the size of the pivot packet for a given device. Assuming encrypted FPV streams, a fundamental problem is the identification the pivot packets. Herein, we take an approach based on their size. As different drones may have different formats for the pivot packets, the challenge is then to estimate the size of pivots from a network trace for each device we encounter during monitoring.

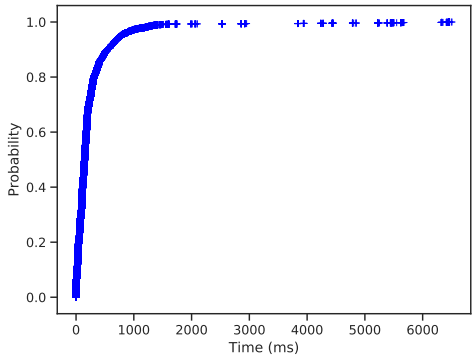
As discussed in Section 6.2.3, video frames are packetized by the Network Interface Card (NIC) and each packet's payload is capped at the Maximum Transmission Unit (MTU) of the transmitting NIC. Typically, the WiFi protocol has an MTU of 2304 bytes whereas Ethernet has an MTU of 1500 bytes. Since operating systems translate all sent and received packets to and from the WiFi card into Ethernet, all packets have to adhere to the Ethernet's MTU which is 1500 bytes. Therefore, video frames that are larger than 1500 bytes are packetized into a series of 1500 bytes packets where the last packet contains the residual of the video frame and its size is typically less than 1500 bytes.

Besides packetized video frames, the NIC also receives pivot packets from the drone application which, we recall, are synchronization packets sent periodically with large inter-arrivals between them (30 ms to 100 ms). On the other hand, each processed video frame is packetized and buffered at the NIC's transmission queue where each packet is sent out as soon as the wireless channel is cleared (inter-arrivals of packets belong to the same video frame ≈ 0.01 ms). Therefore, there is a high probability that pivot packets are buffered between two packetized video frames. In other words, since each packetized video frame ends with a less-than-MTU packet and begins with an MTU packet, a pivot is most likely to appear after a less-than-MTU packet and before an MTU packet (recall Figure 6.3(b)). From

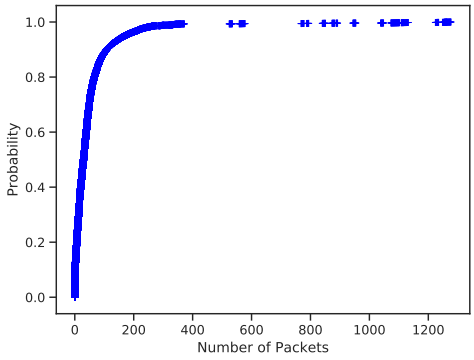
this observation, we established that a group of less-than-MTU packets between two MTU packets might include a pivot packet where the last packet is most likely the pivot. It is also observed that a pivot packet might appear after two less-than-MTU packets or slip individually between two MTUs.

During the pivot analysis, we also observed that a pivot packet has another attribute; it is sent more frequently compared to all other packet types. Based on this observation, we computed the occurrence frequency of all packet sizes within drones' traces. We observed that the highest occurring packet size is the MTU size for all of the drones we considered. Then, we observed that the second-highest occurring packet size for all three drones (BLK, RED, and WHT) is the size of the pivot packet in both motion states (STB and SHK). Knowing these characteristics, we set a new objective to determine how much time and how many packets are required to collect and have enough occurrence frequency in order to find the correct pivot packet size. To approach this objective, we calculate the distribution of the time and number of packets needed to correctly identify the size of the second highest occurring packet for every drone trace file. First, we extract the pivot size manually beforehand as the ground truth for each drone by distributing the occurrence frequency of packet sizes of the entire network trace of each drone. Then, we designate each packet in the trace as the starting point for accumulating all subsequent packets until the pivot size matches the ground truth of the trace file. Now for each starting point, we have the time and number of packets needed to correctly identify the pivot size. Finally, we calculate the Cumulative Distribution Function (CDF) over the detection time (Figure 6.5(a)) and number of packets (Figure 6.5(b)) for all drone traces. From the CDF, we can observe that we need at least 820 milliseconds and at least 170 packets to acquire the correct pivot size with probability of 0.95. Therefore, we set the sample size to be at least 170 packets with a time window of at least 820 millisecond.

Based on these observations, we develop a **Pivot Extraction Algorithm** that is executed in two phases, *Video Frames Fingerprinting* and *Pivot Size Estimation* (see Figure 6.1). During the Video Frames Fingerprinting phase, the algorithm moves a sliding window across a network traffic stream that can fit at least 170 packets sent within least a 820 millisecond window and records any packets sequence that (i) starts and ends with a packet of size exactly MTU, (ii) has less than six consecutive packets, (iii) and the size of all packets (except the first and last packet) is less-than-MTU. Each recorded sequence of packets are denoted as a *Pivot Fingerprint* which might include a pivot packet (Figure 6.6). In the *Pivot Size Estimation* phase, the recorded Pivot Fingerprints are evaluated against three



(a) Cumulative Distribution Function (CDF) of packet inter-arrivals



(b) Cumulative Distribution Function (CDF) of packet sizes

Figure 6.5: Cumulative Distribution Function of time and packets needed to acquire the pivot packet

types of fingerprints: type 1 contains three packet, type 2 contains four packets, and type 3 contains five packets (Figure 6.6). For each fingerprint, the before-the-last packet in the fingerprint is marked as a candidate pivot. Then on a separate process, the occurrence frequency of all packet sizes in the sliding window is computed. Finally, the candidate pivot size that matches the second-highest occurring packet size is declared as the size of the pivot packet.

It remains now to explain how to efficiently estimate the MTU packet size for a given sample of packets. The algorithm declares the size of the first packet in the sample as the MTU size then proceeds to find the next MTU packet. Whenever a packet size that is higher than MTU is detected, the Pivot Extraction Algorithm declares the new packet as the MTU and resumes the pivot estimation process.

Our Pivot Extraction Algorithm has correctly identified up to 99% of pivot packets within a packet trace in $O(n)$ time where n is the number of packets in

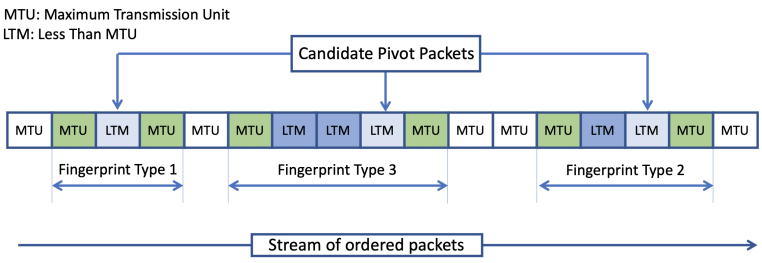
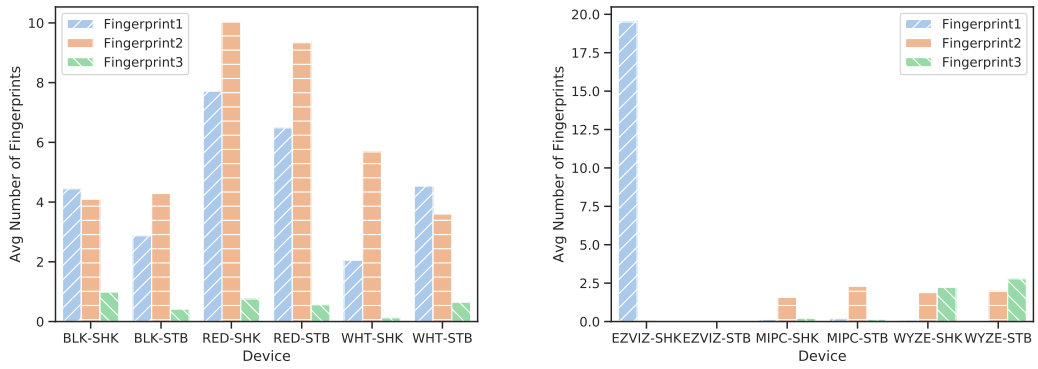


Figure 6.6: Candidate pivot packets.



(a) Drones Fingerprints.

(b) Cameras Fingerprints.

Figure 6.7: Average number of Fingerprint types per second for Drones (6.7(a)) and Cameras (6.7(b)).

the sample. Thus, the Pivot Extraction Algorithm can - with high probability - identify the pivot size.

6.3.4 Pivot Patterns in Other Real-time Video Streaming Devices

One can readily observe that the Video Frames Fingerprinting process is influenced by the video stream emitted by a device and therefore the ability of the Pivot Fingerprints in differentiating between FPV drones and other video streaming devices is questionable. We, then, analyze video streams emitted by IoT cameras in order to compare them with those emitted by FPV drones in terms of pivot patterns.

Real-time video streaming devices, such as IoT cameras, generate streams of packets that have a strong similarity with FPV drones' video streams. Analogously to FPV drones, video streams emitted by IoT cameras are unidirectional. That is, an IoT camera forwards a video stream to its designated app while the app maintains the video connectivity with the camera via heartbeats and keepalive messages. Conversely the video streams generated by VoIP applications (e.g. Skype) are bidirectional. Therefore, we incline toward using IoT cameras in the analysis of our pivot extraction approach since they are the most similar applications to FPV drones. However, we still included VoIP traces for the final evaluation in Section 6.5.1. In order to assess whether or not our pivot approach can distinguish IoT cameras from FPV drones, we collect video traces produced by three different IoT

Table 6.2: Pivot Features

Pivot Features	Description
is_large_MTU	1 if MTU is greater than 1400 bytes and 0 otherwise
fingerprint_1	Number of fingerprint type 1 in the sample
fingerprint_2	Number of fingerprint type 2 in the sample
window_time	Total packets inter-arrivals of the sample
total_packets	Total number of packets in the sample
total_size	Sum of packet sizes in the sample
pivot_size/MTU_size	The ratio of the pivot size to the MTU size
MTU_size/total_length	The ratio of MTU size to the sample size
pivot_size/total_length	The ratio of pivot size to the sample size
MTU_count	Total number of MTU packets in the sample
pivot_count	Total number of pivot packets in the sample
MTU_count/total_packets	Ratio of MTU count to packet count in the sample
pivot_count/total_packets	Ratio of pivot count to packet count in the sample

camera brands: Wyze Cam, EZVIZ C1C, and Lefun MIPC (Figure 6.4(b)). Just like the drones, we collect traces for each camera in a stable (STB) and dynamic motion (SHK) state, resulting in a total of six camera traces. We then calculate how many pivots are recorded per second under each pivot type, i.e., in which type of fingerprint a pivot is found.

In Figure 6.7, for each device (drone and camera), we compute the average number of pivot fingerprint type found per second. From the results, we can see that fingerprint type 3 rarely appears in drones video. On the other hand, each drone has at least four type 2 fingerprints per second. In Wyze Cam and Lefun MIPC cameras, we could detect some fingerprints of type 2 and 3 ($\approx$ 2.5 pivots), but almost no fingerprints of type 1 are detected in their packet traces. Instead, the EZVIZ camera in SHK state has on average about 20 fingerprints of type 1, but almost no fingerprints of type 2 and 3 in its packet trace. From the results, we conclude that other network devices that have traffic patterns similar to drones would have different pivot patterns. Thus, a pivot-based approach can discriminate the two classes of applications.

Table 6.3: Statistical measurements for computing Statistical Features

Measurements	Description
Standard Deviation	$\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - mean(x))^2}$
Variance	$\sigma = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - mean(x))^2$
Root Mean Square	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2}$
Mean Square	$\frac{1}{N}\sum_{i=1}^{N}(x_i)^2$
Pearson Skewness	$3(mean(x) - median(x))/\sigma$
Kurtosys	$\frac{1}{N}\sum_{i=1}^{N}((x_i - mean(x))/\sigma)^4$
Skewness	$\frac{1}{N}\sum_{i=1}^{N}((x_i - mean(x))/\sigma)^3$
Minimum	$(Min(x_i)\|_{i=1,...,N})$
Maximum	$(Max(x_i)\|_{i=1,...,N})$
Mean	$\frac{1}{N}\sum_{i=1}^{N}(x_i)$
Median	$\lceil\frac{N+1}{2}\rceil\|(Sort_{i=1,...,N})$
Mean Absolute Deviation	$\frac{1}{N}\sum_{i=1}^{N}(\|x_i - mean(x)\|)$
Median Absolute Deviation	$median(\|x_i - median(x)\|)$

6.4 Classification Model & Features Design

In this Section, we shift our discussion to the second component of the framework, the *Classification Model* (recall Figure 6.1). The classification model we adopt for the FPV Drone Classifier of our framework is the Random Forest (RF) algorithm. This is motivated by its low complexity and good performance in many settings. Moreover, RF is widely used in the literature to address similar problems.

We combine the RF algorithm with Grid Search; a technique where different hyperparameters are tested for a fine-tuning process. The hyperparameters that we used are: the maximum depth of the tree, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, and the number of trees in the forest. The basic idea is that the RF is repeated multiple times, with different hyperparameters, and the combinations of those hyperparameters that gives the best results in the validation set inside the Grid Search, are used for the final test using the test set.

For the rest of this Section, we discuss the process of utilizing pivot packets as an anchor for creating machine learning features for detecting unprofiled drones. Recall that one of the main contributions presented in this Chapter is detecting drones that the FPV Drone Classifier has never trained on. In Section 6.4.1, we provide the list of features used by the classifier. Then in Section 6.4.2, we intro-

duce two novel feature selection strategy that pick the best features for unprofiled drone detection in two phases. The first phase is based on Jaccard Similarity Index and is independent of the underlying classification model that is used in the framework, while the second phase is based on the Recursive Feature Elimination (RFE) technique and it involves the classification model during the selection process, meaning that its performance depends on the type of the machine learning algorithm used.

6.4.1 Feature Creation

As discussed in Section 6.3, pivot packets have the potential to differentiate between FPV drones and other network devices, including video streaming devices that exhibit similar traffic patterns to FPV drones such as IoT cameras. This differentiating potential creates an opportunity for constructing machine learning features that have the discriminative power to classify network devices into either drones or non-drones based only on the traffic observed from a WiFi network. In particular, the features are expected to correctly classify (with high probability) drone devices among all other non-drone devices even if the machine learning model does not have a prior profile of the drone being classified. The set of features are organized into the following three groups:

Pivot Features

Pivot features are features composed from the output of the Pivot Extraction Algorithm. Recall that the algorithm outputs a sequence of packets that contains at least 170 packets which are sent within at least 820ms time window. Within this sequence, the algorithm also tags the pivot fingerprints; the location of pivot packets with respect to the MTU and less-than-MTU video frame packets that envelop the estimated pivots (see Figure 6.6).

The set of pivot features and their definitions are reported in Table 6.2 and is comprised of 13 features. A brief description of two pivot features is provided next:

MTU Size Feature: During the drone's pivot fingerprint analysis in Section 6.3.1, we observed that FPV drones tend to fill their MTU packets to the maximum (1500 bytes in RED and BLK) or near maximum (1454 bytes in WHT) payload allowable by Ethernet protocol. From this observation, we created is_MTU_Large feature which is set to 1 if the Pivot Extraction Algorithm declares a device's MTU

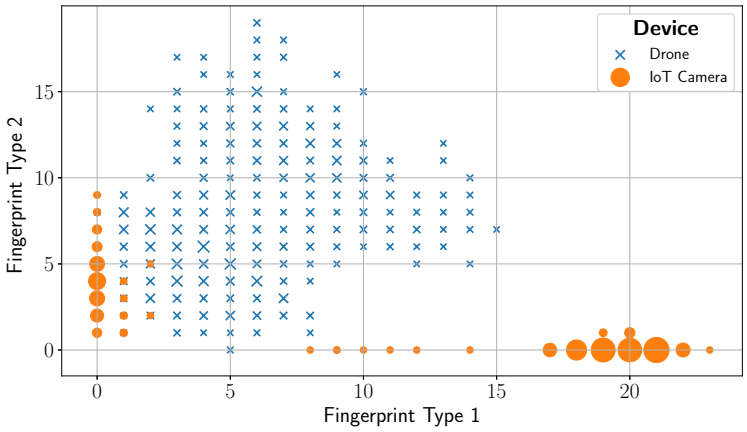


Figure 6.8: Scatter plot of fingerprint type 1 & 2 for drones and cameras

is more than some threshold - heuristically set to 1400 bytes - and 0 otherwise.

Pivot Fingerprint Features: We plotted the number of pivot fingerprint type 1 and 2 in a scatter plot for FPV drones and IoT cameras in Figure 6.8. From the plot, the difference in fingerprint appearance behavior between drones and cameras can be observed and therefore it has the potential of having the number of fingerprints as a feature for each device. Note that the scatter plot is derived from Figure 6.7. Because of the insignificant number of appearance of fingerprint type 3, it has not been considered as a feature.

Statistical Features

The State-of-the-Art in machine learning-based drone detection framework that exploits wireless network traffic for drone detection [116] uses 12 statistical measurements for feature generation. Just like our framework, the drone detection framework proposed by Alipur et. al. [116] uses only packet sizes and inter-arrivals for feature generation. These features will be denoted as *statistical features* and their statistical measurements are reported in Table 6.3 with the addition of the 'Variance' feature which was not used by Alipur et. al. In our framework, we extend our feature set and include these statistical features to our own set as well. Including such features to our feature set is a crucial step toward performing a comparative analysis between our pivot features and the statistical features. The analysis will be conducted in Section 6.6.1 and it will provide an insight on the performance of our pivot features compared to the latest drone detection features

proposed in the literature.

Recall that our framework assumes an encrypted network traffic and the Pivot Extraction Algorithm only extracts the size of WiFi packets and their inter-arrivals. Therefore each statistical measurement is computed twice, once over the *packet sizes* and another over their *inter-arrivals* which results in a total of 26 statistical features (13 statistical features $\times$ 2 packet measurements).

Statistical Pivot Features

The statistical pivot features combine the statistical measurements with our pivot measurements to create *Statistical Pivot Features*. From every sequence of packets received from the Pivot Extraction Algorithm, we extract three additional measurements, the inter-arrival time between pivot packets (pivot time-distance), the number of normal packets between pivot packets (pivot packet-distance), and the total number of bytes between pivot packets (pivot length-distance). For each one of the three values, we compute the statistical measurements on each group which resulted in 39 statistical pivot features (3 different pivot measurements $\times$ 13 statistical measurements). Combined with the rest of the two previously discussed feature subsets, the total feature set considered for our framework is 78 features.

6.4.2 Feature Selection

In a classification problem, conventional supervised feature selection techniques such as Recursive Feature Elimination (RFE) and Permutation Feature Importance aim at selecting features that better distinguish a class label for every class that exist in a dataset. In the context of unprofiled drone detection however, the model with the selected features would overfit the data used in the training phase and consequently would fail at detecting drones whose data was not included in the training set. This problem occurs because different samples/packets that are drawn from the same source would appear in the training set and the testing set which would create an undesired strong correlation between certain features and a specific device. In other words, the problem occurs because the selected features are too specific for the data used to train the model, creating a sort of mapping between the data and the class labels. Therefore, because an unknown drone class label is not inside the dataset, the model would fail its detection.

For example, the intrinsic feature selection of the Random Forest algorithm might select the pivot size as the main feature and therefore, declare any WiFi device that has a pivot size of 46 bytes (BLK), 4 bytes (RED), or 35 bytes (WHT)

as *drone* and any other pivot sizes as *non-drone* devices. This is an unwanted behavior because the model associates the unique pivot dimension with a class label, ignoring the other features. This problem will persist regardless of the dataset size; the number of packets collected from drones since all drone-labeled samples are representing (in our case) three distinct drones and their three pivot sizes are uniquely consistent with every drone sample.

Of course this would not be a problem if we implement our model under the assumption that every drone type in the market must be profiled prior to the detection process. However, this is not a realistic assumption for creating a practical drone detection framework. To the best of our knowledge, no prior efforts that adopt a machine learning-based approach for drone detection has addressed this issue.

To this end, we next propose our novel feature selection technique that overcomes the overfitting problem via a two-phase process. The first phase computes a Feature Similarity Score based on Jaccard Similarity Index (model-independent feature selection), while the second phase applies a modified RFE for unprofiled drones (model-dependent feature selection). The objective of our feature selection technique is to select features which describe a generalized drone behavior rather than features that describe a behavior of a specific networked device. In other words, the selected features are expected to detect unprofiled drones; drones that the framework has never seen before and the classifier has never trained on.

The main motivation of applying Jaccard Similarity Index (JSI) testing before using an RFE-based feature selection, is that the computation of JSI scores is much faster than running an RFE-based algorithm which helps us weed out the weak features before passing them to the RFE process. For a comparison, it took our computer (a laptop with Intel i7) around 20 seconds to calculate the JSI scores for 78 features while it took the same computer around four hours to compute the RFE scores of the remaining features selected by JSI (which is 37 as we will see next).

Phase I - Feature Similarity Score

As we have seen in Section 6.4.2, some features might describe a drone as a unique device rather than an FPV drone. For this reason, we need to identify the features that capture a generalized drone behavior and not a specific drone type. Our first step toward accomplishing this task is computing a Feature Similarity Score; a score that describes how a certain feature can capture the behavior of

all the three drones we have in our experiment. Note that this feature selection approach is independent from the classification model used in the framework. In other words, Feature Similarity Score pre-processes the features without involving the adopted machine learning algorithm and therefore Feature Similarity Score results are always consistent regardless of the used classification algorithm.

We first start by plotting the distribution of each feature as a histogram across all drones and measure the similarity of their distributions. For example, Figure 6.9(a) represents the distributions of *pivot_packet_distance_mean_root_square* feature for all three drones while Figure 6.9(b) represents the distribution for *pivot_packet_distance_total_size* feature for the same drones. From the Figure we can see that the first feature represents each drone individually while the second feature shows some shared characteristics among the drones. The main intuition is that if a feature shares similar distributions among different drones then this feature will most likely generalize a drone behavior while other features with distinctive distributions can help identify the specific device it represents. An extremely distinctive distribution such as the pivot size could overfit the data.

There are different techniques in the literature for testing the similarity between data distributions such as Pearson's chi-squared test and Kolmogorov–Smirnov test. We chose the Jaccard Similarity Index method because it is easier to interpret and provides greater flexibility in setting the appropriate similarity threshold that fits our desired application.

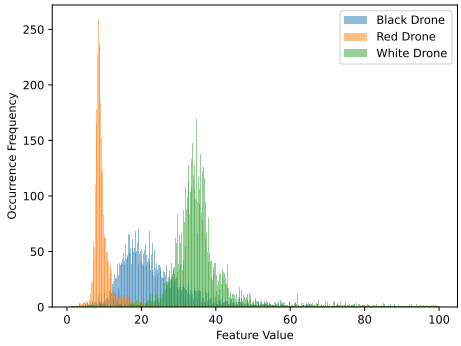
Pre-processing: Before testing the similarity of a feature between drones, we need to prepare the data points of that feature to be plotted as a histogram. First we remove the anomalies of each feature; the data points that are scattered (very few) and are either extremely large or extremely small compared to the majority of the data points. Since we don't have a prohibitively large number of features, we removed the anomalies for each feature manually by setting the histogram range for each feature to be within the range that capture most of the data points while leaving out the very few data points that are extremely out of range. Then we perform 'data binning'; a process of dividing the range into same-length intervals, each interval (a histogram bar) is referred to as a 'bin'. A data point that falls within the range of a certain bin will increase the value of that bin by one. If anomalies are not removed, it would collapse the bins together and severely change the shape of the histogram's distribution since the histogram range will be proportionally large for the majority of the data points. After removing the anomalies, we set the number of bins to 100 for all features as a unified bin interval.

Jaccard Similarity Index Measurement: After creating the histograms, we measure the size of the intersection area between the three drones' distributions. Then, we measure the occupation ratio of the intersection area to the total distribution area using a Jaccard index. For each bin position B_i, we record the minimum bin value among the three drones bins. The intersection area is then computed as $\sum_{i=1}^{n} min(B_{i_{1...m}})$ where n is the number of bins and m is the number of histograms. Note that since we have three histograms (one for each drone), then $m = 3$. Similarly, the total area of the three histogram is $\sum_{i=1}^{n} max(B_{i_{1...m}})$. Finally, the Jaccard Similarity Index JSI is computed as:

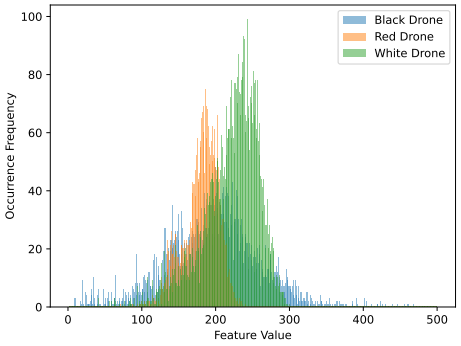
$$JSI = \frac{\sum_{i=1}^{n} min(B_{i_{1...m}})}{\sum_{i=1}^{n} max(B_{i_{1...m}})} \tag{6.1}$$

The resulted JSI gives us the ratio of how much of the intersected region occupies from the total sum of histograms areas.

Feature Selection Based On Jaccard Similarity Index: We computed the Jaccard Similarity Index for all of the 78 features based on equation 6.1. The results are sorted in ascended order and reported in Figure 6.10. Note that the Jaccard indices are ratios (≤ 1) and in the Figure they are multiplied by 100 for clarity. From the Figure, we notice that there are high variations between the top eight features where the index difference between features ft_i and ft_{i+1} ($\Delta JSI = ft_i - ft_{i+1}$) can reach up to 13 points between two consecutive features for the top eight indices. Then the ΔJSI started to stabilize to be ≈ 1 until it reaches to the 30^{th} feature which has $JSI = 12.71$ and $\Delta JSI = 2.64$. In other words, the number of useful features started to drop by $\Delta JSI = 2.64$ points after $JSI < 12$ and therefore we select the acceptance threshold to be 12, that is, the intersection region must occupy at least 12% of the total histogram areas. With this threshold, we reduce our features set from 78 features to 30 features. Additionally, we add another threshold that accepts a feature if it has $JSI > 70$ of intersection area between any two drones. The intuition of this additional threshold is that, a certain feature might be a strong candidate for detecting FPV drones but one of the three drones we used during testing might have a unique and outlying distribution towards that particular feature and therefore would inject an anomaly in the JSI testing which would potentially lower the JSI score of rather a strong feature. Therefore, we re-computed JSI score for each feature thrice, once between each two drones.



(a) Distribution of *pivot-packet-distance-mean-root-square* feature.



(b) Distribution of *pivot-packet-distance-total-size* feature.

Figure 6.9: Feature (a) uniquely describes each drone individually while feature (b) generalizes an FPV drone behavior.

In other words, for each feature we computed JSI for drones combinations BLK-WHT, WHT-RED, and BLK-RED. Then, we set a heuristically high threshold in which we set to $JSI > 70$ and selected the features that has $JSI > 70$ in at least one of the three drones combinations. The new threshold gave us additional 7 more features which resulted in a total of 37 features.

Phase II - Recursive Feature Elimination for Unprofiled Drones

The features selected in Section 6.4.2 were selected independently from the underlying classification model used in the detection framework. Now we need to test the sensitivity of the remaining features and further select the features that are most sensitive toward our Random Forest classifier in detecting new unseen drones. For this task, we implement a modified version of the Recursive Feature Elimination (RFE) technique that involves our FPV Drone Classifier which influences the RFE process to select features that have the discriminative power to detect unprofiled drones (Algorithm 1).

Since RFE involves the classification model in the selection process, the model would eventually influence the RFE to select the features that would cause overfitting for the reasons discussed in the beginning of Section 6.4.2. To overcome this issue, we modify the basic RFE technique and inject a Cross-Validation test in the step that invokes the classifier (line 9 to 13 in Algorithm 1). Since we have three drones, we apply a 3-folds cross-validation strategy, where we divide the dataset into three distinct sets or folds (line 6). However, the samples are not distributed

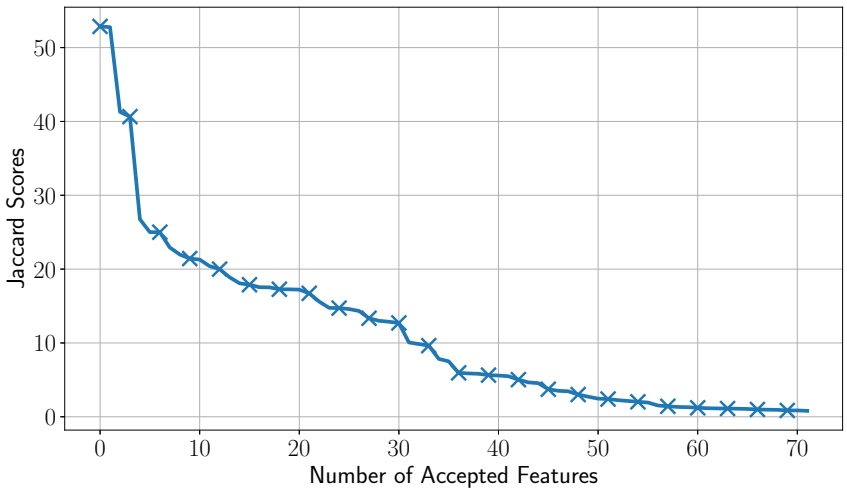


Figure 6.10: Jaccard Scores × 100. Each score defines the threshold of accepted features.

equally among the folds as in a conventional cross-validation. Instead, each fold contains samples from a single drone plus non-drone samples selected randomly. Then the RFE train on two folds and validate on the third fold. This process is repeated once for each drone/fold as the validation set. In other word, the RFE train on two drones and validate on new unseen drone then shuffle between the drones and train and validate again. The accuracy for each fold is accumulated out of 300% (# of drones × 100 at line 13) then mean of the accuracy is computed (line 14) after the cross-validation is performed on all drones. At the end of each iteration (line 15), the feature that yields the maximum accuracy when removed (i.e., weakest feature) is eliminated. Next, new and smaller set of features and their respective scores are then recorded (line 16) and the final feature set is selected at the end of the algorithm (line 17) based on the highest detection accuracy yielded by the best feature set. Our modified RFE process has further decreased the number of features from 37 to 12. It is worth noting that our features that are built from pivot packets are among the remaining features, and out of 28 statistical features, only three of them are kept including our 'Variance' feature in the final set. In summary, the selected features are: 3 from the Pivot Features group (is_MTU_Large, fingerprint_1, and fingerprint_2), 6 from the Pivot Statistical Features group (2 Pivot Time Distance features, 1 Pivot Packet Distance feature, and 3 Pivot Length Distance features), and 3 from the Statistical Features group (2 from Alipur et. al. [116] and our 'Variance' feature). We refer to the final set of

selected features as **Pivot-Based Features**.

Algorithm 2: Modified Recursive Feature Elimination

```
iteration_results.initialize();
while feature_set.count ¿ 1 do
    features_scores.initialize();
    for ft in feature_set do
        feat_set_no_ft = feature_set - {ft};
        drone_subsets = k_fold_cross_validation(dataset);
        ft_scores.initialize();
        for drone_set in drone_subsets do
            validation_set = drone_set;
            train_set = drone_subsets - drone_set;
            model = RandomForest(train_set, feat_set_no_ft);
            accuracy = model(validation_set);
            ft_scores.add(accuracy);
        features_scores.add(ft, ft_scores.mean());
    feature_set.remove_weakest_feature(features_scores);
    iteration_results.record(feature_set);
final_feature_set = iteration_results.best_feature_set();
```

6.5 Implementation

In this Section, we provide an overview of our experimental setup, the devices and applications that are used to generate network traffic, and the data collection strategy used to generate our datasets.

6.5.1 Data Collection

This subsection provides a quick overview on the hardware and software setup used to collect the network traces to generate the framework datasets, as well as the devices and applications that generated such traces.

Hardware & Software Setup

To collect the traffic emitted by WiFi devices, we used an Acer Aspire F5-573G-759N laptop with Intel Core i7 and Alfa AWUS036ACH WiFi adapter. We

used Kali Linux 64-Bit version 2020.1, which was installed as a virtual machine using Oracle VirtualBox version 6.0.14. The WiFi adapter was set to Monitor Mode and tuned to the operating frequency of the drone's AP being monitored. We used the Android app "WiFi Analyzer" [130] to find the correct operating frequency of the AP. The network traces were collected using Wireshark [131] and parsed using Scapy [132].

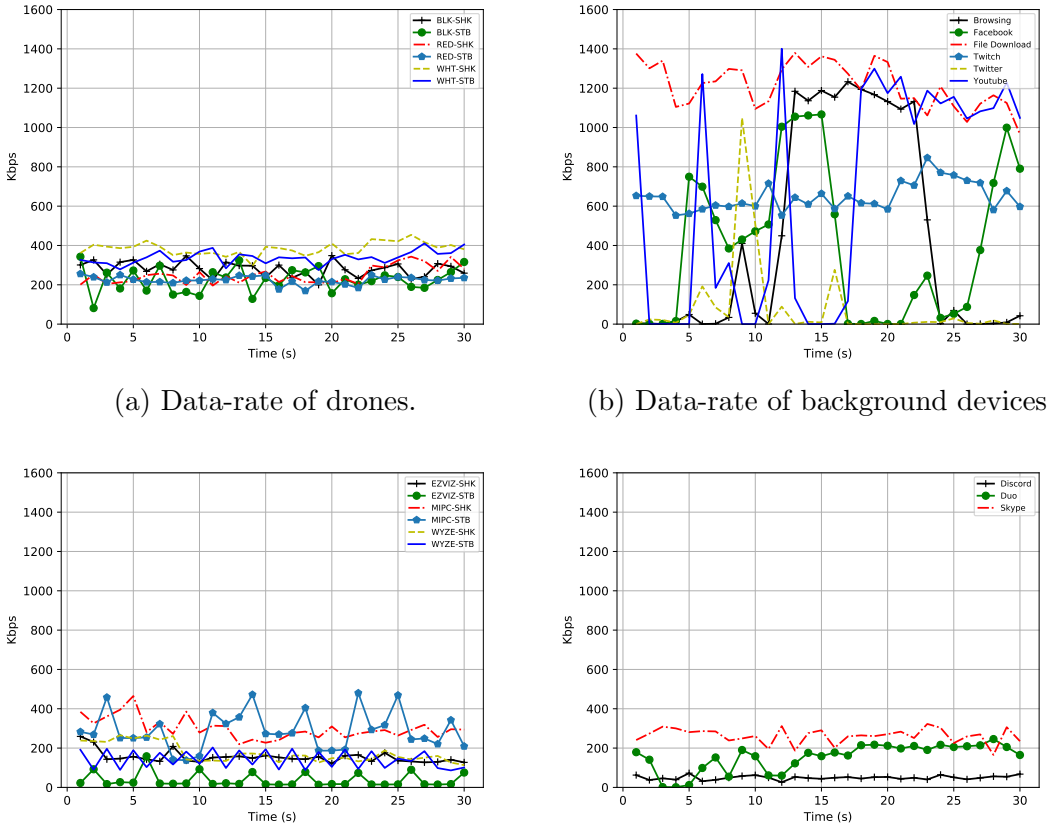
Network Traffic Generation

The network traffic traces are generated from various devices and applications in different conditions and scenarios. All traces are collected using the WiFi Monitor Mode (rather than on-device traffic capture) to include packet loss and other distortion effects that characterize real-world environments. The network traces are collected from the following devices and network services:

VoIP Apps: To test the performance of our framework, we ensure that our datasets include samples extracted from network devices that have traffic patterns similar to FPV drones'. In addition to the IoT camera traces collected for the pivot analysis in Section 6.3.4, we also collect traces of video traffic originated from Skype, Google Duo, and Discord VoIP apps while making video calls for five minutes. In Figure 6.11, we can clearly see that bit-rates of FPV drones, IoT cameras, and VoIP applications have similar traffic patterns.

Non-Drone Background Devices: To ensure the completeness of our dataset, we collect six different traces of network traffic originated from network services for three minutes each. These traces were collected during 1) file downloading, 2) non real-time video streaming (YouTube), 3) live internet video streaming (Twitch), 4) Internet browsing, and social media browsing: 5) Twitter and 6) Facebook. The bit-rate of each application is plotted in Figure 6.11(b).

Drones & IoT Cameras: The traces from drones and IoT cameras that are collected for the pivot analysis in Section 6.3.2 and 6.3.4 respectively, are collected under two different extreme cases; a completely stable video scene (STB) and a highly dynamic video scene (SHK). To include network traffic generated from a more realistic video pattern, we place all of the three drones and all of the three cameras in front of a screen playing a five minutes documentary video that contains slow, moderate, and fast-moving scenes. From each device, we capture the emitted video traffic which resulted in an additional six network traffic traces.



(a) Data-rate of drones.

(b) Data-rate of background devices.

(c) Data-rate of cameras.

(d) Data-rate of VoIP.

Figure 6.11: Data-Rate for Drones, Background Devices, Cameras and VoIP applications over 30 seconds.

Flying Drones: To further ensure that our dataset includes realistic samples, we recorded the FPV video of our drones during flight. First, we performed an experiment to determine the maximum distance from the monitoring station at which a drone can be detected. We set up a laptop as a monitoring station in the center of a university football field. Then we set up seven waypoints in a straight line starting 10 meters from the monitoring station. The distance between each two consecutive waypoints is 10 meters. We then flew each drone starting from the monitoring station in a straight line to the first waypoint while collecting the packets sent by the drone. Then, we flew the drone from the first waypoint to the second, up to the seventh while collecting the transmitted packets at each step. The drones we used are considered "micro-drones", and move at a relatively low

speed with a rather small tilting of the drone body during motion.

We notice that the RSSI severely drops after 40 meters at waypoint 4 and the received power level at the monitoring station's antenna is below −80 dBm. This is expected, as these drones are low-power devices with limited battery capacity. The packet loss between waypoint 3 and waypoint 4 was excessively high (around 70%). Therefore, we conclude that in the considered hardware configuration, the effective detection distance for the micro-drones is 30 meters.

From our monitoring station in the center of the football field, we flew each drone for three minutes inside a detection area of radius equal to 30 meters while the drone occasionally exits and re-enters the area to simulate a more realistic scenario of invasive drones. For each drone, we collected the emitted video traffic which resulted in three new network traffic traces.

6.5.2 Data Preprocessing

For each captured WiFi packet P_i transmitted by device D_j, we extract from the packet header the transmitter's MAC address, fragment number, sequence number, header flags, payload size $S_{i,j}$, and packet's arrival time $I_{i,j}$. We also extract the RSSI from the radiotap header that is added by the WiFi card upon receiving the packet. In case of fragmented packets, we use the sequence number, fragment number, and the "more fragments" flag in the header to put fragmented WiFi packets back into a complete packet. This approach is needed as WiFi adapters set into Monitor Mode do not reconstruct fragmented packets, and pass them individually to the operating system. We use the sequence number to identify retransmitted packets that were already captured. Note that we could not rely on the "retry" flag to capture duplicates because is not guaranteed that the original packet transmission has already been captured. Also a simple check of consecutive duplicate sequence numbers would not work either, since some retransmissions might arrive out of order, and thus the last captured packet is not always followed by its retransmitted duplicate. As an alternative solution for identifying duplicates, we simply keep track of the last eight captured packets for each D_j and whenever a new packet is received, we check if the packet is already in this window.

For each captured stream of n WiFi packets, we reconstruct the fragmented, drop the duplicates, then group them as a list of packets based on the transmitter's MAC address such as $D_j = \{P_{1,j}, ..., P_{n,j}\}$. For each packet i, we only save 1) the packet size S_i and 2) its arrival time I_i. Therefore, each device's packet list can be interpreted as $D_j = \{(S_{1,j}, I_{1,j}), ..., (S_{n,j}, I_{n,j})\}$ where $P_{i,j} = (S_{i,j}, I_{i,j})$. Then, we

add each device list D_j to our Packet Dataset $DS_p = \{D_1, ..., D_m\}$.

6.5.3 Data Sampling

After preparing DS_p, the packets are grouped into batches called samples (SP). Each SP consists of at least 170 packets that are sent within at least 820ms interval (The reason for sampling packets in batches with a minimum of 170 packets and a minimum of 820ms in inter-arrival window is discussed in Section 6.3.3). This process transforms DS_p into a Samples Dataset DS_{sp}. To illustrate, to generate the first sample $SP_{1,j}$ for a given device $D_j \in DS_p$, a sliding-window strategy is used; the sliding-window is filled with 170 $P_{i,j}$ packets for device D_j. Then if $I_{170,j} - I_{1,j} < 820ms$, the sliding-window is filled with additional n packets until $I_{170+n,j} - I_{1,j} \geq 820ms$. Finally, the current packets in the sliding-window is recorded as $SP_{1,j}$ and added under device D_j. For the next sample, the sliding-window shifts by 30 packets to create the second sample $SP_{2,j} = \{P_{31,j}, ..., P_{201,j}\}$ and add additional n packets until $I_{201+n,j} - I_{31,j} \geq 820ms$ is satisfied. This sampling process continues until all packets for D_j are batched into m samples such that $D_j = \{SP_{1,j}, ..., SP_{m,j}\}$ where typically $n > m$ since each two consecutive samples (with at least 170 packets) have 30 interleaving packets offset. The reason for this offset choice is because during the pivot analysis, we observed that two pivot packets are separated by a maximum of 30 normal non-pivot packets. In other words, 30-packets is the maximum offset that includes all possible pivots combinations among consecutive samples.

6.5.4 Features Extraction

The last step in creating our datasets is extracting features from the samples. Since the samples are already processed, features are easily computed from each sample $SP_{i,j}$ producing a machine learning record $R_{i,j}$ that contains a set of k features Ft where $R_{i,j} = \{Ft_{i,j_1}, ..., Ft_{i,j_k}\}$. The feature design is already discussed in detail in Section 6.4.1 and reported in tables 6.2 and 6.3 while the feature selection strategy is discussed in Section 6.4.2. Since the framework follows a supervised machine learning paradigm, each record is labeled 1 if it belongs to a drone, and 0 otherwise.

6.6 Evaluation

In this Section, we evaluate the detection accuracy of our proposed framework using the features created and selected in Section 6.4 and the dataset generated in Section 6.5. During the feature selection process, we used only 80% of the dataset and kept the remaining 20% for the final evaluation. Recall that one of the main motivations of the proposed framework is detecting unprofiled drones that the framework has never seen before. To test the framework's ability in detecting unprofiled drones, we use a 3-folds cross-validation technique on both the testing set and training set. In particular, we designate the 80% of the dataset that is used during the feature selection process as the training set and divided it into three subsets, each subset includes samples from a single drone plus randomly selected non-drone samples. The remaining 20% is also divided into the same manner, a single test subset for each drone. Finally, we collected additional samples from a fourth drone, a 3DR Solo drone (Figure 6.12). The Solo drone was never used in any shape or form in neither the analysis of the pivot behavior nor the feature selection process. The samples of the new drone were collected in the same manner as in Section 6.5.1.

The evaluation of the framework is divided into two experiments, each experiment is executed under four different settings. In the first experiment, we applied the features that were selected by our feature selection strategy proposed in Section 6.4.2 which is comprised of our pivot-based features. In the second experiment, we applied the statistical features only (Table 6.3) which was proposed by [116]. In each experiment, the framework is 1) trained on RED and BLK and tested on WHT, 2) trained on WHT and RED and tested on BLK, 3) trained on BLK and WHT and tested on RED, and 4) trained on RED, BLK, and WHT and tested on Solo. The results are reported in Table 6.4.

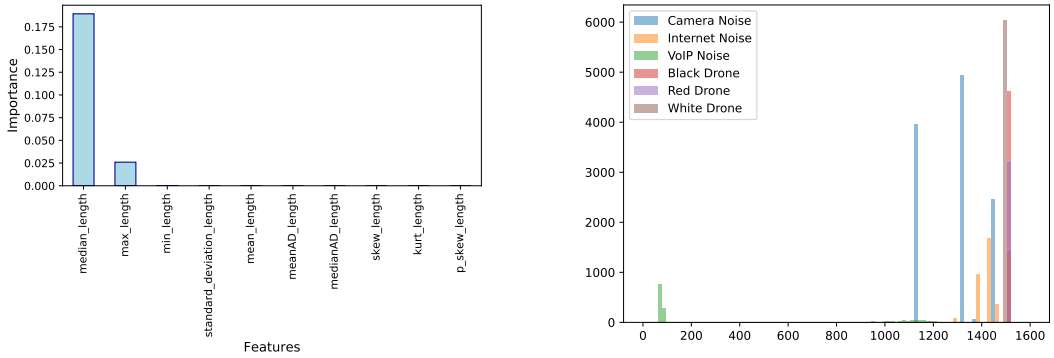
Table 6.4: Comparison of achieved accuracy.

	Detection Accuracy			
	Red Drone	Black Drone	White Drone	Solo Drone
Statistical Features	93%	69%	55%	94%
Pivot-Based Features	99%	94%	98%	98%

Figure 6.12: 3DR Solo Drone that was used as part of the final test set

6.6.1 Results Analysis

The results reported in Table 6.4 demonstrate that pivot-based features can detect unprofiled drones with accuracy of at least 94%. On the other hand, statistical features - such as those used in state-of-the-art machine learning-based drone detection - struggle at detecting unprofiled drones, especially when detecting WHT drone. To understand the drawbacks of the statistical features, we apply a permutation feature importance technique to score the importance index for the statistical features only (Figure 6.13(a)). From the Figure, we observe that the one and only important feature is the median; the middle value of a sorted list of values. As discussed in Section 6.2.3, a video stream is comprised mostly of MTU packets and therefore the median of a series of 170 packets is almost always the MTU size; maximum packet size in the sample. The second most important feature is the packet with the maximum payload (i.e., MTU) which essentially reflects the same value of the median feature. Then we plot the distribution of the MTU for each device in Figure 6.13(b). As we can observe, there is almost no variance in the distribution of the MTU. In other words, every device has a unique MTU size that is tied to its network interface card (NIC) and is represented as a single bar in Figure 6.13(b). This explains why the classification model privileges this single feature in the detection decision. In fact, the model can associate MTUs of 1500 bytes (RED and BLK drones) and 1482 bytes (WHT drone) to drone devices and any other MTU sizes to a non-drone device. This also explains why the detection accuracy drops to 55% when WHT drone is removed from the training set (since RED and BLK have the same MTU size).



(a) Permutation Feature Importance for Statistical Features.

(b) Distribution of the MTU feature. Note that "Camera Noise" is comprised of three distinct IoT cameras.

Figure 6.13: Fixed-valued features such as MTU force the classifier to heavily rely on a single feature

On the other hand, our feature selection strategy ensures the removal of features that have a single or very few distinct values across samples drawn from the same device such as MTU or pivot size which have the potential to uniquely identify a device rather than a group of devices that share the same characteristics (i.e., FPV drones). Furthermore, the selected features are mostly influenced by the behavior of pivot packets that indeed capture a generalized drone behavior rather than a single unique device behavior.

In conclusion, we proposed a COTS Drone Detection Framework that can detect invasive FPV drones flying into a restricted area without the need of having a prior profile of the invasive drone or requiring the drone's controller to be within the detection range. The framework rely synchronization packets that are interleaved with the video stream emitted by drones. These packets are denoted as "pivots" and unlike prior work, pivot packets are not affected by the varying bit-rate of the drone's video stream which can be used as a guideline to construct features for a machine learning-based system built from a Random Forest classifier. We also proposed a *Pivot Extraction Algorithm* algorithm that quickly searches a sample of packets for pivots in linear time. Furthermore, we proposed two-phase feature selection strategy that selects drone features which have the discriminative power to detect unprofiled drone. The first phase is a model-independent feature selection process which is based on Jaccard Similarity Index, while the second phase is

model-dependent and based on the RFE selection process.

Our experiments demonstrated that pivot-based features can detect unprofiled FPV drones even when other video streaming devices such as IoT cameras are within the detection environment. The detection accuracy using our pivot-based features is at least 94% using at least 170 captured packets that are transmitted within at least 820ms, while the detection accuracy of using the state-of-the-art drone features [116] is at least 55% using the same number of packets and detection window.

Bibliography

[1] Giulio Rigoni, Bhumika, Cristina M Pinotti, Debasis Das, and Sajal K Das. Uav-based dataset: a case study. *Submitted to: the 95th Vehicular Technology Conference (VTC2022-Spring).*

[2] Francesco Betti Sorbelli, Cristina M Pinotti, and Giulio Rigoni. On the evaluation of a drone-based delivery system on a mixed euclidean-manhattan grid. *Submitted to: Transactions on Intelligent Transportation Systems, Special Issue on Intelligent Supply Chain in Modern Challenges.*

[3] Lorenzo Palazzetti, Cristina M Pinotti, and Giulio Rigoni. A run in the wind: Favorable winds make the difference in drone delivery. In *Proceeding of the 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 109–116. IEEE, 2021.

[4] Francesco Betti Sorbelli, Federico Corò, Lorenzo Palazzetti, Cristina M Pinotti, and Giulio Rigoni. How the wind can be leveraged for saving energy in a truck-drone delivery system. *Submitted to: Transactions on Intelligent Transportation Systems, Special Issue on Intelligent Supply Chain in Modern Challenges.*

[5] Francesco Betti Sorbelli, Cristina M Pinotti, and Giulio Rigoni. Range-free localization algorithms with mobile anchors at different altitudes: A comparative study. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*, pages 1–10, 2020.

[6] Francesco Betti Sorbelli, Sajal K Das, Cristina M Pinotti, and Giulio Rigoni. A comprehensive investigation on range-free localization algorithms with mobile anchors at different altitudes. *Pervasive and Mobile Computing*, 73:101383, 2021.

[7] Francesco Betti Sorbelli, Mauro Conti, Cristina M Pinotti, and Giulio Rigoni. Uavs path deviation attacks: Survey and research challenges. In *Proceedings of the 2020 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–6. IEEE, 2020.

[8] Anas Alsoliman, Giulio Rigoni, Marco Levorato, Cristina Pinotti, Nils Ole Tippenhauer, and Mauro Conti. Cots drone detection using video streaming characteristics. In *Proceedings of the International Conference on Distributed Computing and Networking 2021*, pages 166–175, 2021.

[9] Anas Alsoliman, Giulio Rigoni, Marco Callegaro, Marco Levorato, Cristina Pinotti, and Mauro Conti. Intrusion detection framework for invasive fpv drones using video streaming characteristics. *Submitted to: ACM Transactions on Cyber-Physical Systems*.

[10] David Schneider. The delivery drones are coming. *IEEE Spectrum*, 57(1):28–29, 2020.

[11] On demand drone delivery. `https://flytrex.com/`, 2020. [Online; accessed 12-01-2021].

[12] Research-driven solutions to critical challenges in the uas industry. `https://vtnews.vt.edu/articles/2020/11/ictas-maap-UPP2.html`, 2020. [Online; accessed 07-01-2021].

[13] The Information. How many packages are moved by fedex and ups everyday? `shorturl.at/gpEPW`, 2021. [Online; accessed 13-11-2021].

[14] Sri Anggraeni, Aulia Maulidina, Mauseni Wantika Dewi, et al. The deployment of drones in sending drugs and patient blood samples covid-19. *Indonesian Journal of Science and Technology*, pages 18–25, 2020.

[15] Yanchao Liu. An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. *Computers & Operations Research*, 111:1–20, 2019.

[16] Ann E McKellar et al. Dual visible-thermal camera approach facilitates drone surveys of colonial marshbirds. *Remote Sensing in Ecology and Conservation*, 7(2):214–226, 2021.

[17] Isla Duporge, Marcus P Spiegel, Eleanor R Thomson, et al. Determination of optimal flight altitude to minimise acoustic drone disturbance to wildlife using species audiograms. *Methods in Ecology and Evolution*, 12(11):2196–2207, 2021.

[18] Joshuah K Stolaroff, Constantine Samaras, Emma R O'Neill, Alia Lubers, Alexandra S Mitchell, and Daniel Ceperley. Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery. *Nature communications*, 9(1):1–13, 2018.

[19] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458, 2018.

[20] Niels A.H. Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. *ERIM Report Series Reference No. ERS-2015-011-LIS*, 2018.

[21] Francesco Betti Sorbelli, Federico Corò, Sajal K Das, and Cristina M Pinotti. Energy-constrained delivery of goods with drones under varying wind conditions. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[22] Juan C Correa, Wilmer Garzón, Phillip Brooker, Gopal Sakarkar, Steven A Carranza, Leidy Yunado, and Alejandro Rincón. Evaluation of collaborative consumption of food delivery services through web mining techniques. *Journal of Retailing and Consumer Services*, 46:45–50, 2019.

[23] Jacco M Hoekstra and Joost Ellerbroek. Bluesky atc simulator project: an open data and open source approach. In *Proceedings of the 7th International Conference on Research in Air Transportation*, volume 131, page 132. FAA/Eurocontrol USA/Europe, 2016.

[24] Luca Bartoli, Francesco Betti Sorbelli, Federico Corò, Cristina M. Pinotti, and Anil Shende. Exact and approximate drone warehouse for a mixed landscape delivery system. In *2019 IEEE International Conference on Smart Computing, Washington, DC, USA, 12-14 June 2019*, 2019.

[25] Chase C Murray and Amanda G Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation: Emerging Technologies*, 54:86–109, 2015.

[26] Stefan Poikonen, Bruce Golden, and Edward A Wasil. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346, 2019.

[27] Jay R Brown, Maxim A Bushuev, and Alfred L Guiffrida. Distance metrics matter: analysing optimisation algorithms for the last mile problem. *International Journal of Logistics Systems and Management*, 38(2):151–174, 2021.

[28] Ning Ai, Junjun Zheng, Xiaochen Chen, and Kazuya Kawamura. Neighborhood-specific traffic impact analysis of restaurant meal delivery trips: Planning implications and case studies in chicago. *Journal of Urban Planning and Development*, 147(2):05021013, 2021.

[29] Francesco Betti Sorbelli, Federico Coro, Cristina M Pinotti, and Anil Shende. Automated picking system employing a drone. In *2019 15th Intl. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, pages 633–640. IEEE, 2019.

[30] Arindam Khanda, Federico Coro, Francesco Betti Sorbelli, Cristina M Pinotti, and Sajal K Das. Efficient route selection for drone-based delivery under time-varying dynamics. In *18th International Conference on Mobile Ad-Hoc and Smart Systems (MASS). IEEE*, 2021.

[31] Francesco Betti Sorbelli, Federico Corò, Sajal K Das, Lorenzo Palazzetti, and Cristina M Pinotti. Greedy algorithms for scheduling package delivery with multiple drones. In *Proceedings of the 23rd International Conference on Distributed Computing and Networking*, 2022.

[32] Yuyu Li, Wei Yang, and Bo Huang. Impact of uav delivery on sustainability and costs under traffic restrictions. *Mathematical Problems in Engineering*, 2020, 2020.

[33] George O Wesolowsky. The weber problem: History and perspectives. *Computers & Operations Research*, 1993.

[34] Chanderjit Bajaj. Proving geometric algorithm non-solvability: An application of factoring polynomials. *Journal of Symbolic Computation*, 2(1):99–102, 1986.

[35] Murray H Protter and Charles Bradfield Morrey. *College calculus with analytic geometry*. Addison-Wesley, 1977.

[36] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[37] Michael Frigge, David C Hoaglin, and Boris Iglewicz. Some implementations of the boxplot. *The American Statistician*, 43(1):50–54, 1989.

[38] Júlia Cária de Freitas and Puca Huachi Vaz Penna. A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27(1):267–290, 2020.

[39] Gloria Cerasela Crişan and Elena Nechita. On a cooperative truck-and-drone delivery system. *Procedia Computer Science*, 159:38–47, 2019.

[40] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Stefan Schwerdfeger. Drone delivery from trucks: Drone scheduling for given truck routes. *Networks*, 72(4):506–527, 2018.

[41] Roberto Roberti and Mario Ruthmair. Exact methods for the traveling salesman problem with drone. *Transportation Science*, 2021.

[42] F. Betti Sorbelli, F. Corò, S. K. Das, and C. M. Pinotti. Energy-constrained delivery of goods with drones under varying wind conditions. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2020.

[43] John M. Wallace and Peter V. Hobbs. *Atmospheric science: An introductory Survey.* Amsterdam: Elsevier Academic Press., 2006.

[44] Wayne Johnson. *Helicopter theory.* Courier Corporation, 2012.

[45] Matthew Ayamga, Selorm Akaba, and Albert Apotele Nyaaba. Multifaceted applicability of drones: A review. *Technological Forecasting and Social Change*, 167:120677, 2021.

[46] Robin Kellermann, Tobias Biehle, and Liliann Fischer. Drones for parcel and passenger transportation: A literature review. *Transportation Research Interdisciplinary Perspectives*, 4:100088, 2020.

[47] Javier Burgués and Santiago Marco. Environmental chemical sensing using small drones: A review. *Science of The Total Environment*, page 141172, 2020.

[48] Abbas Yazdinejad, Reza M Parizi, Ali Dehghantanha, et al. Enabling drones in the internet of things with decentralized blockchain-based security. *IEEE Internet of Things Journal*, 8(8):6406–6415, 2020.

[49] Brent Skorup and Connor Haaland. How drones can help fight the coronavirus. *Mercatus Center Research Paper Series, Special Edition*, 2020.

[50] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, and alias. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.

[51] Xianyu Chang, Chaoqun Yang, Junfeng Wu, Xiufang Shi, and Zhiguo Shi. A surveillance system for drone localization and tracking using acoustic arrays. In *2018 IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 573–577. IEEE, 2018.

[52] Saleh O Al-Jazzar et al. Aoa-based drone localization using wireless sensor-doublets. *Physical Communication*, 42:101160, 2020.

[53] Michael A Goodrich, Bryan S Morse, Damon Gerhardt, Joseph L Cooper, Morgan Quigley, Julie A Adams, and Curtis Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

[54] Balmukund Mishra, Deepak Garg, Pratik Narang, and Vipul Mishra. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156:1–10, 2020.

[55] Samira Hayat, Evşen Yanmaz, Christian Bettstetter, and Timothy X Brown. Multi-objective drone path planning for search and rescue with quality-of-service requirements. *Autonomous Robots*, 44(7):1183–1198, 2020.

[56] Chengyi Qu, Rounak Singh, Alicia Esquivel Morel, Francesco Betti Sorbelli, Prasad Calyam, and Sajal K Das. Obstacle-aware and energy-efficient multi-drone coordination and networking for disaster response. In *2021 17th International Conference on Network and Service Management (CNSM)*, pages 1–9. IEEE, 2021.

[57] UM Rao Mogili and BBVL Deepak. Review on application of drone systems in precision agriculture. *Procedia computer science*, 133:502–509, 2018.

[58] Wayne D Rosamond, Anna M Johnson, Brittany M Bogle, et al. Drone delivery of an automated external defibrillator. *New England Journal of Medicine*, 383(12):1186–1188, 2020.

[59] Sheldon Cheskes, Shelley L McLeod, Michael Nolan, et al. Improving access to automated external defibrillators in rural and remote settings: a drone delivery feasibility study. *Journal of the American Heart Association*, 9(14):e016687, 2020.

[60] Hailong Huang, Andrey V Savkin, and Chao Huang. Reliable path planning for drone delivery using a stochastic time-dependent public transportation network. *IEEE Trans. on Intelligent Transportation Systems*, 2020.

[61] Taner Cokyasar, Wenquan Dong, Mingzhou Jin, and İsmail Ömer Verbas. Designing a drone delivery network with automated battery swapping machines. *Computers & Operations Research*, 129:105177, 2021.

[62] Guangjie Han, Jinfang Jiang, Chenyu Zhang, Trung Q Duong, Mohsen Guizani, and George K Karag. A survey on mobile anchor node assisted localization in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 18(3):2220–2243, 2016.

[63] Han Xiao, Hao Zhang, Zengfeng Wang, and T Aaron Gulliver. An rssi based dv-hop algorithm for wireless sensor networks. In *2017 IEEE PACRIM*, pages 1–6. IEEE, 2017.

[64] L. Zhao, X. Wen, and Dan Li. Amorphous localization algorithm based on bp artificial neural network. In *International Conference on Frontiers of Internet of Things 2014*, pages 178–183, 2014.

[65] Jun Wang, Paulo Urriza, Yuxing Han, and Danijela Cabric. Weighted centroid localization algorithm: theoretical analysis and distributed implementation. *IEEE Transactions on wireless communications*, 10(10):3403–3413, 2011.

[66] Ji zeng Wang and Hongxu Jin. Improvement on apit localization algorithms for wireless sensor networks. In *2009 Int. Conf. on Networks Security, Wireless Comm. and Trusted Computing*, volume 1, pages 719–723. IEEE, 2009.

[67] Dimitrios Koutsonikolas, Saumitra M Das, and Y Charlie Hu. Path planning of mobile landmarks for localization in wireless sensor networks. *Computer Communications*, 30(13):2577–2592, 2007.

[68] Rui Huang and Gergely V Zaruba. Static path planning for mobile beacons to localize sensor networks. In *Pervasive Computing and Communications Workshops, 2007*, pages 323–330. IEEE, 2007.

[69] Jinfang Jiang, Guangjie Han, Huihui Xu, Lei Shu, and Mohsen Guizani. Lmat: Localization with a mobile anchor node based on trilateration in wireless sensor networks. In *IEEE GLOBECOM*, pages 1–6. IEEE, 2011.

[70] Bin Xiao, Hekang Chen, and Shuigeng Zhou. Distributed localization using a moving beacon in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(5):587–600, 2008.

[71] Sangho Lee, Eunchan Kim, Chungsan Kim, and Kiseon Kim. Localization with a mobile beacon based on geometric constraints in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 8(12):5801–5805, 2009.

[72] Francesco Betti Sorbelli, Cristina M. Pinotti, and Vlady R. Range-free localization algorithm using a customary drone: Towards a realistic scenario. *Pervasive and Mobile Computing*, 54:1–15, 2019.

[73] Francesco Betti Sorbelli and Cristina M Pinotti. Ground localization with a drone and uwb antennas: Experiments on the field. In *2019 IEEE 20th International Symposium on WoWMoM*, pages 1–7. IEEE, 2019.

[74] Jianlin Chen, Devin Raye, Wahab Ali Gulzar Khawaja, Priyanka Sinha, and Ismail Güvenç. Impact of 3d uwb antenna radiation pattern on air-to-ground drone connectivity. *2018 IEEE 88th Vehicular Technology Conference*, pages 1–5, 2018.

[75] Wahab Ali Gulzar Khawaja, Ozgur Ozdemir, Fatih Erden, Ismail Guvenc, and David Matolak. Ultra-wideband air-to-ground propagation channel characterization in an open area. *arXiv preprint arXiv:1906.04013*, 2019.

[76] Priyanka Sinha, Yavuz Yapici, and Ismail Guvenc. Impact of 3d antenna radiation patterns on tdoa-based wireless localization of uavs. In *IEEE Con-*

ference on Computer Communications Workshops, pages 614–619. IEEE, 2019.

[77] DecaWave. *Product Documentation*, 2020. `https://www.decawave.com/product-documentation/`.

[78] 3D Robotics. *Solo Specs: Just the facts*, 2020. `https://www.mydronelab.com/reviews/3dr-solo.html`.

[79] DecaWave. *Datasheet for the DWM1001C*, 2021. `https://www.decawave.com/dwm1001/datasheet/`.

[80] Francesco Betti Sorbelli, Sajal K Das, Cristina M Pinotti, and Simone Silvestri. On the accuracy of localizing terrestrial objects using drones. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[81] Francesco Betti Sorbelli, Sajal K. Das, Cristina M. Pinotti, and Simone Silvestri. Range based algorithms for precise localization of terrestrial objects using a drone. *Pervasive and Mobile Computing*, 48:20–42, 2018.

[82] Angelo Trotta, Fabio D Andreagiovanni, Marco Di Felice, Enrico Natalizio, and Kaushik Roy Chowdhury. When uavs ride a bus: Towards energy-efficient city-scale video surveillance. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1043–1051, 2018.

[83] Aakash Khochare, Yogesh Simmhan, Francesco Betti Sorbelli, and Sajal K. Das. Heuristic algorithms for co-scheduling of edge analytics and routes for uav fleet missions. In *IEEE INFOCOM*, pages 1–10, 2021.

[84] Jean-Paul Yaacoub, Hassan Noura, Ola Salman, and Ali Chehab. Security analysis of drones systems: Attacks, limitations, and recommendations. *Internet of Things*, 11:100218, 2020.

[85] U.S. Department of Transportation Federal Aviation Administration. Uas remote identification overview, 2021-03-09.

[86] DJI. Dji aeroscope, 2021-05-21.

[87] Robin Radar Systems. Robin radar systems, 2021-05-21.

[88] Nassi et al. Drones' cryptanalysis-smashing cryptography with a flicker. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1397–1414. IEEE, 2019.

[89] Bisio et al. Unauthorized amateur uav detection based on wifi statistical fingerprint analysis. *IEEE Communications Magazine*, 56(4):106–111, 2018.

[90] A. Fotouhi, H. Qiang, and alias. Survey on uav cellular communications: Practical aspects, standardization advancements, regulation, and security challenges. *IEEE Communications Surveys & Tutorials*, 21(4):3417–3442, 2019.

[91] H. Wang, H. Zhao, J. Zhang, D. Ma, J. Li, and J. Wei. Survey on unmanned aerial vehicle networks: A cyber physical system perspective. *IEEE Communications Surveys & Tutorials*, 2019.

[92] Gaurav Choudhary, Vishal Sharma, Ilsun You, Kangbin Yim, Ray Chen, and Jin-Hee Cho. Intrusion detection systems for networked unmanned aerial vehicles: a survey. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 560–565. IEEE, 2018.

[93] CG Leela Krishna and Robin R Murphy. A review on cybersecurity vulnerabilities for unmanned aerial vehicles. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 194–199. IEEE, 2017.

[94] Desmond Schmidt, Kenneth Radke, Seyit Camtepe, Ernest Foo, and Michał Ren. A survey and analysis of the gnss spoofing threat and countermeasures. *ACM Computing Surveys (CSUR)*, 48(4):1–31, 2016.

[95] Alan Boulanger. Open-source versus proprietary software: Is one more reliable and secure than the other? *IBM Systems Journal*, 44(2):239–248, 2005.

[96] Guochang Xu and Yan Xu. *GPS: theory, algorithms and applications*. Springer, 2016.

[97] Niranjana Vagle, Ali Broumandan, and Gérard Lachapelle. Multiantenna gnss and inertial sensors/odometer coupling for robust vehicular navigation. *IEEE Internet of Things Journal*, 5(6):4816–4828, 2018.

[98] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 75–86, 2011.

[99] Jon S Warner and Roger G Johnston. Gps spoofing countermeasures. *Homeland Security Journal*, 25(2):19–27, 2003.

[100] Juhwan Noh, Yujin Kwon, Yunmok Son, Hocheol Shin, Dohyun Kim, Jaeyeong Choi, and Yongdae Kim. Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing. *ACM Transactions on Privacy and Security (TOPS)*, 22(2):1–26, 2019.

[101] Rui Xu, Mengyu Ding, Ya Qi, Shuai Yue, and Jianye Liu. Performance analysis of gnss/ins loosely coupled integration systems under spoofing attacks. *Sensors*, 18(12):4108, 2018.

[102] Mauricio Donatti, Felipe Frazatto, Leandro Manera, Telmo Teramoto, and Eduardo Neger. Radio frequency spoofing system to take over lawbreaking drones. In *2016 IEEE MTT-S Latin America Microwave Conference (LAMC)*, pages 1–3. IEEE, 2016.

[103] Johann-Sebastian Pleban, Ricardo Band, and Reiner Creutzburg. Hacking and securing the ar. drone 2.0 quadcopter: Investigations for improving the security of a toy. In *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*, volume 9030, page 90300L, 2014.

[104] Ottilia Westerlund and Rameez Asif. Drone hacking with raspberry-pi 3 and wifi pineapple: Security and privacy threats for the internet-of-things. In *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, pages 1–10. IEEE, 2019.

[105] Vishal Dey, Vikramkumar Pudi, Anupam Chattopadhyay, and Yuval Elovici. Security vulnerabilities of unmanned aerial vehicles and countermeasures: An experimental study. In *2018 31st International Conference on VLSI Design (VLSID)*, pages 398–403. IEEE, 2018.

[106] Ali Khalajmehrabadi, Nikolaos Gatsis, David Akopian, and Ahmad F Taha. Real-time rejection and mitigation of time synchronization attacks on the global positioning system. *IEEE Transactions on Industrial Electronics*, 65(8):6425–6435, 2018.

[107] Chao Sun, Joon Wayn Cheong, Andrew G Dempster, Laure Demicheli, Ediz Cetin, Hongbo Zhao, and Wenquan Feng. Moving variance-based signal quality monitoring method for spoofing detection. *GPS Solutions*, 22(3):83, 2018.

[108] Yang Liu, Sihai Li, Qiangwen Fu, Zhenbo Liu, and Qi Zhou. Analysis of kalman filter innovation-based gnss spoofing detection method for ins/gnss integrated navigation system. *IEEE Sensors Journal*, 19(13):5167–5178, 2019.

[109] Zhang Renyu, Seow Chee Kiat, Wen Kai, and Zhang Heng. Spoofing attack of drone. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 1239–1246. IEEE, 2018.

[110] Abdel Rahman Eldosouky, Aidin Ferdowsi, and Walid Saad. Drones in distress: A game-theoretic countermeasure for protecting uavs against gps spoofing. *IEEE Internet of Things Journal*, 2019.

[111] Yinrong Qiao, Yuxing Zhang, and Xiao Du. A vision-based gps-spoofing detection method for small uavs. In *13th International Conference on Computational Intelligence and Security (CIS)*, pages 312–316. IEEE, 2017.

[112] Masood Varshosaz, Alireza Afary, Barat Mojaradi, Mohammad Saadatseresht, and Ebadat Ghanbari Parmehr. Spoofing detection of civilian uavs using visual odometry. *International Journal of Geo-Information*, 9(1):6, 2020.

[113] Antonina P Melikhova and Igor A Tsikin. Optimum array processing with unknown attitude parameters for gnss anti-spoofing integrity monitoring. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–4. IEEE, 2018.

[114] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. Dewicam: Detecting hidden wireless cameras via smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 1–13, 2018.

[115] Simon Birnbach, Richard Baker, and Ivan Martinovic. Wi-fly?: Detecting privacy invasion attacks by consumer drones. 2017.

[116] Alipour-Fanid et al. Machine learning-based delay-aware uav detection and operation mode identification over encrypted wi-fi traffic. *IEEE Transactions on Information Forensics and Security*, 15:2346–2360, 2019.

[117] SpotterRF. Drone detection system, 2020-03-13.

[118] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Flying objects detection from a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4128–4136, 2015.

[119] Sai Ram Ganti and Yoohwan Kim. Implementation of detection and tracking mechanism for small uas. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1254–1260. IEEE, 2016.

[120] Ellen E Case, Anne M Zelnio, and Brian D Rigling. Low-cost acoustic array for small uav detection and tracking. In *2008 IEEE National Aerospace and Electronics Conference*, pages 110–113. IEEE, 2008.

[121] Busset et al. Detection and tracking of drones using advanced acoustic cameras. In *Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*, volume 9647, page 96470F. International Society for Optics and Photonics, 2015.

[122] Phuc Nguyen et al. Matthan: Drone presence detection by identifying physical signatures in the drone's rf communication. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 211–224, 2017.

[123] Bisio et al. Improving wifi statistical fingerprint-based detection techniques against uav stealth attacks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[124] Li et al. Drone profiling through wireless fingerprinting. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 858–863. IEEE, 2017.

[125] Sciancalepore et al. Detecting drones status via encrypted traffic analysis. In *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*, pages 67–72, 2019.

[126] Sciancalepore et al. Picking a needle in a haystack: Detecting drones via network traffic analysis. *arXiv preprint arXiv:1901.03535*, 2019.

[127] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 243–255, 2018.

[128] Martins Ezuma, Fatih Erden, Chethan Kumar Anjinappa, Ozgur Ozdemir, and Ismail Guvenc. Micro-uav detection and classification from rf fingerprints using machine learning techniques. In *2019 IEEE Aerospace Conference*, pages 1–13. IEEE, 2019.

[129] Junia Valente and Alvaro A Cardenas. Understanding security threats in consumer drones through the lens of the discovery quadcopter family. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, pages 31–36, 2017.

[130] Wifi analyzer, 2020-03-13.

[131] Wireshark, 2020-03-13.

[132] Scapy, 2020-03-13.