Automatic Tracking of a Target Using a Small UAV

João Morais
joaopgfmorais@tecnico.ulisboa.pt

Instituto Superior Técnico - Universidade de Lisboa, Lisboa, Portugal

July 2019

Abstract

The drone sector is experiencing an aggressive growth, both in terms of users and applications. The maturation of technology has contributed to increasingly more affordable components, allowing virtually anyone to own a drone. In addition, new developments have allowed these platforms to perform more complex tasks, including aerial mapping, surveillance and inspection of hard to access places. Video or image capture is among the most common uses given to a drone. However, not all contexts allow the user to control the equipment efficiently and perform the task at hand. This project proposes a system that, using data regarding a mobile target's position, automatically tracks the target, with the attitude the user desires, for example moving always on the reference's right. To achieve this goal, a drone was projected, built and tested, a ground station was developed to control the drone, along with an application for Android smartphones to retrieve the positions of the target to follow. During the development of this project, whenever possible, open-source software from on-going projects, such as ArduPilot and MAVLink, was integrated. Aspects such as safety of operation, both during and post mission, were considered, including the creation of a network of previously visited paths.

Keywords: Drones (UAV), Android OS, Ground Control Station (GCS), ArduPilot, MAVLink, Networks

1. Introduction

Unmanned Aerial Vehicles (UAV), also named drones, are defined as aircraft with no on-board crew or passengers capable of performing a given mission and safely returning. The earliest report of drone usage dates back to 1849 during the Austrian siege of Venice in Italy, when the military used hot air balloons filled with explosives, rigged to a timer, with the objective of bombarding the city from above [1]. The war theatre was, in fact, the moving force behind the advancement of drones. The idea was always the same: to minimize the casualties of allied troops during risky manoeuvres. Despite their usage in training and reconnaissance, drones would continue to be seen as risky and unreliable assets on the battlefield until the 1980s, when drone warfare boomed, especially in the Middle East.

It would not be until the 21st Century, more precisely in 2006, that drones would start being used in non-military scenarios, specifically as a means of providing relief to disaster struck areas, monitoring and surveillance. It was also in this year that the Federal Aviation Agency (FAA) issued the first drone permit [2]. Since then, drones have seen an ever increasing diversification in terms of hardware, software, missions and user base. The prediction in 2017 was that the number of drones would exceed all other aviation groups combined by an order of magnitude within that year, on both the professional and consumer sectors [3]. The most common use given to UAVs is image or video recording. However, not all contexts allow the user to actively control the equipment and record the surroundings or the target. For example, an athlete running an orienteering competition cannot perform this tasks and guarantee his or her movement and safety at the same time. These are cases in which a drone with automatic reference tracking would prove useful. As such, an outdoor automatic tracking system for a small UAV was implemented, with the following objectives:

1. Project and construction of a drone, based on low-cost market options, with special consideration for energy and payload aspects;
2. Development of a system to obtain the user's geographical position, based on GNSS;
3. Development of Ground Control Station (GCS) software, implementing a tracking algorithm that allows the controller to specify the attitude of the drone during tracking. Safety of

operation is another topic to be explored;

4. Demonstration of proof of concept of the proposed solution.

2. Mission Specifications

Since the purpose of the system is to track a mobile target autonomously, possibly capturing footage of that target, the following mission specifications have been established:

- The drone needs to be capable of tracking a target that may or may not be in movement;
- The target is a person whose pace during movement is expected to oscillate between slow walking and steady running. As such, assume that the target's velocity is between 1 m/s (3.6 km/h) and 2.78 m/s (10 km/h);
- The drone should have a minimum autonomy of 10 minutes;
- Assume good meteorological conditions: wind up to 10 km/h and absence of rain;
- The drone should follow the target as closely as possible, depending on the attitude desired. As such, and to account for the latency of the connections and the delay in the response, the drone should be within 15 metres of horizontal range of the position it needs to occupy.

2.1. Projected System Architecture

In order to fulfil the objectives set and the missions specifications, the projected system is divided in three equally important subsystems. They are:

1. The UAV: it is responsible for tracking the target. Given the goal of having the best cost-effective solution, open-source software is of great importance. As such, ArduPilot, an open-source auto-pilot suite, is used on-board. Given the necessity for beyond line of sight operations, broadband 3G/4G cellular technology is used to communicate with the ground computer;
2. The smartphone: it is responsible for acquiring the target's position and forwarding it to the ground computer. This is achieved through an application developed for Android smartphones, programmed in Java. Communication with the ground computer is achieved through a TCP session;
3. The ground computer: it is responsible for running the GCS software, implementing the tracking algorithm, through processing of the position estimates received from the smartphone, and monitoring the drone. The GCS implementation involves a mix of existing GCS software, such as MAVProxy and Mission Planner, and code developed in Python. Communication with the drone is achieved by using the MAVLink message protocol.

The implemented system architecture is shown on figure 1.

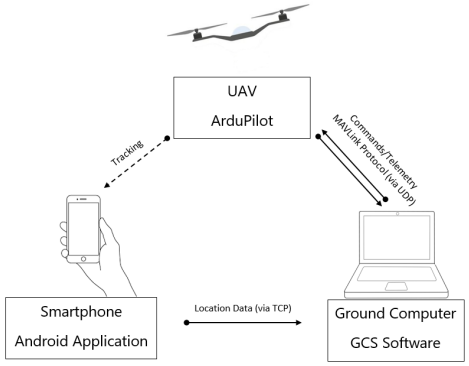

Figure 1: System architecture

3. Drone Construction

The first step is to project and build the drone to use in the duration of the project. Given the existence of many types of drones, it makes sense to analyse them first. The most common drones types are multi-rotors, fixed-wing and hybrids. While the first is characterized by the usage of motors and propellers that generate vertical thrust in order to hover, the second has a wing which allows the drone to fly, with motors and propellers mounted on the wing to allow horizontal movement. As such, multi-rotors can perform Vertical Take-Off and Landing (VTOL), are easier to manoeuvre and can fly on confined spaces but suffer from a reduced payload capacity and a higher energy consumption since flight is achieved through a constant vertical force, which translates into small autonomy. Fixed-wing drones require a clear space to take-off, are less manoeuvrable and more experience is needed when piloting. However, they are capable of much longer flight times, which makes them ideal for applications that require more time in the air or covering larger distances. A multi-rotor/fixed-wing hybrid, joins the best characteristics of the two, being able to fly in multi-rotor configuration when precision manoeuvring is required and in fixed-wing configuration to cover more ground. They are, though, highly complex to develop, since two different flight characteristics need to be seamlessly integrated [4].

In terms of required hardware, whatever the drone type may be, the requirements are similar [5]. A main-frame is necessary, where the payload and other components are mounted. Motors and propellers are either mounted on the tip of each main frame arm or on the trailing edge of the wing. Electronic Speed Controllers (ESC) are used to vary the speed and direction of rotation of the motors.

Since brushless motors are typically used, they also convert DC power into three-phase AC power. A flight controller is also installed on-board and is responsible for issuing commands, reading and interpreting data from on-board sensors and controlling the other hardware components during flight. For flight beyond line of sight to be possible, a link to the ground computer needs to be online at all times. As such, broadband 3G/4G cellular network technology is used. In order to use this type of link, an on-board computer is required, which will interface between the flight controller and the ground computer. In addition, for autonomous flight and waypoint navigation, a GNSS receiver is required. Finally, a mission effective battery should be used. This list can, naturally, be expanded with the required payloads to fulfil a specific mission.

3.1. Component Selection

The implemented process for component selection was adapted from the conceptual design process proposed in [6], which covers areas of analysis including take-off weight estimation, wing, fuselage and tail design, take-off and landing considerations, among others. However, not all of these elements are of interest for the design of a drone and, as such, the following design process was implemented:

1. Defining the mission profile;
2. Choosing the drone configuration;
3. Obtaining the total mass of the platform;
4. Obtaining the thrust required for flight;
5. Selecting the appropriate motor and propeller;
6. Choosing the ESCs;
7. Obtaining the required battery mass;

3.2. Autonomy Analysis

The choice the battery to include in a drone is heavily dependant on the mission intended and its implications in terms of autonomy. As such, a study was performed to attempt to find an upper bound for the maximum achievable autonomy with marker components, prioritizing the low cost ones.

The assumptions made are:

1. The drone has only one motor;
2. The motor is weightless;
3. All the generated thrust is directed to lifting battery mass;
4. There are no losses in the circuit or other battery consumption;
5. The hover scenario is considered, with a thrust-to-weight ratio of 1.

With these, a drone is reduced to the concept of a flying battery. It is expected the results will reflect solely the impact of different combinations of motor and propeller and simplify the selection of one of said pairs. Note that, under these assumptions, increasing the number of motors and the battery mass results in the same increase in battery consumption and autonomy remains the same.

Autonomy, A, can be expressed as:

$$A\ [min] = 60 * \frac{E\ [Wh/kg] * T\ [kg]}{P\ [W]} \tag{1}$$

where E is the energy density of the batteries, T is thrust and P is the power required to produce that thrust.

To obtain results, energy density needs to be computed and pairs of thrust and power, a characteristic of the combinations of motor and propeller, need to be collected from the manufacturer datasheets and benchmarks. Energy density or specific energy, characterizes the nominal energy per unit of mass and is dependant on the battery chemistry and packaging [7]. Because of their higher energy and power density, along with their shape, Lithium-Polymer (LiPo) batteries are usually used in UAVs in detriment of other batteries. With the information collected, namely from 3-cells, 4-cells and 6-cells batteries, the graphic of figure 2 was made, which depicts batteries organized by their C-rate, defined as a measure of the rate at which a battery is discharged relative to its maximum capacity.

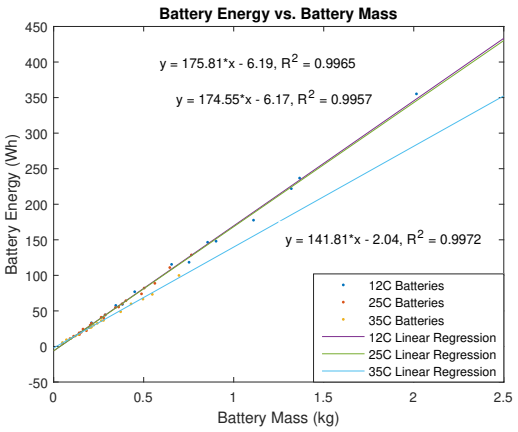

Figure 2: Energy versus Mass for the Batteries Studied

It is possible to see that different C-rates result in different values for energy density due to the higher mass of the batteries. Since the aim is finding the upper bound for autonomy, the value of energy density for 12C batteries will be used, 175.81 Wh/kg, as it is the highest.

The information collected regarding motors and propellers is found on table 1, where PD refers to Propdrive, TG to Turnigy and MS to Multistar. It is possible to observe that the Multistar 2216-800 motor with 11 Thin-Style (T-S) propellers and a 3-cell battery results in the highest value of autonomy,

Table 1: Motor Information

Name	Cells	Prop.	T (kg)	P (W)	A (min)
PD 2826-10[A]	3	9x4.7	0.632	176	37.9
PD 2826-11[B]	3	9x6	0.730	149	51.7
TG L2215J-9[C]	3	-	0.820	200	43.3
TG D2836/9[D]	3	9x6	0.850	243	36.9
TG D2830-11[E]	3	8x4	0.890	210	44.7
MS 2216-8[F]	3	11 T-S	0.908	128	74.8
PD 2826-10[G]	4	9x6	0.937	235	42.1
MS 2216-8[H]	4	11 T-S	1.006	217	48.9
PD 2826-11[I]	4	9x6	1.050	285	38.9
TG D2826-11[J]	4	-	1.050	265	41.8
MS 3508-5.8[K]	4	11x4.5	1.360	209	68.7
MS 3508-7[L]	4	11x4.7	1.370	277	52.2
MS 3508-7[M]	4	12x3.8	1.630	335	51.4

at 74.8 minutes. This is because, when compared to the other pairs, it possesses a much lower power consumption for the thrust it outputs. This can be seen better on the graphic of figure 3, where each of the pairs is identified by a letter from A to K, according to the table. The combination in question is identified by the letter F.

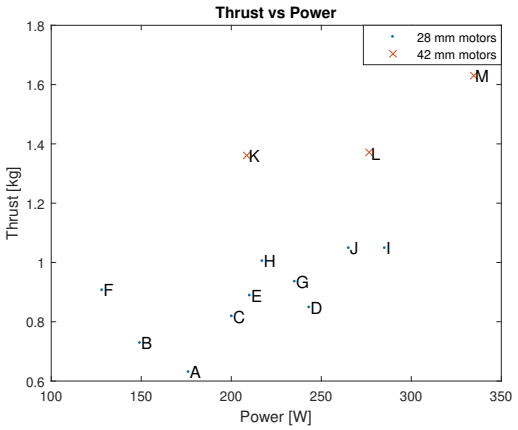

Figure 3: Thrust vs Power for the motors studied

3.3. Implementation

On figure 4, the final drone can be seen, fully assembled. It uses the combination from before.

Figure 4: Final drone set-up

In terms of batteries, two types were purchased: one with 3-cells, 25C rated and with capacity of 5800 mAh, weighting 0.427 kg; and another with 3-cells, 35C rated and with 2200 mAh of capacity, weighting 0.191 kg. The drone without battery weights 1.145 kg. Applying equation 1 with the thrust variable assuming the thrust required to lift the battery mass and the power variable assuming the power required to produce enough thrust for the drone to hover yields:

$$A_{2.2Ah} = 60 * \frac{141.81 * \frac{0.191}{4}}{\frac{1.336}{4} * \frac{128}{0.908}} = 8.62\ min \qquad (2)$$

and

$$A_{5.8Ah} = 60 * \frac{174.55 * \frac{0.427}{4}}{\frac{1.572}{4} * \frac{128}{0.908}} = 20.18\ min \qquad (3)$$

Note that the energy density value assumes 141.81 Wh/kg for the 2200 mAh battery and 174.55 Wh/kg for the 5800 mAh battery, since they are of 35C and 25C rates, respectively. It is expected that the real autonomy values are lower, since there will be other consumptions and losses.

4. Android App for Position Acquisition

In order for tracking to be possible, the reference's geographical position needs to be made available to the drone. This can be achieved with the usage of Global Navigation Satellite System (GNSS) receivers. GNSS refers to a system which includes a constellation of satellites providing signals from space that transmit data to GNSS receivers, which is used to determine location. Each position estimate is computed by determining the distance between the receiver and the satellites and correcting it for errors such as atmospheric delays and clock offsets [8].

In order to simplify communication between different electronics, the National Maritime Electronics Association (NMEA) introduced the 0183 Interface Standard. This is a combined electrical and data specification for communicating between marine electronics used worldwide across many industry segments [9]. It is intended for one way serial data transmission from a single talker to one or more listeners. This standard includes a set of messages that are used to transmit specific data, such as position, velocity, timing and course [10]. For this project, NMEA GGA sentences, which transmit position data, are of interest.

The reference needs to be in possession of a GNSS receiver, in order to obtain location data. These can be purchased individually from sellers. However, data needs to be forwarded to the ground computer to be processed, demanding that a broadband 3G/4G cellular network link is established. The receiver would have to be interfaced with a computer, for example a Raspberry Pi, to be able to send data. There is another solution to this problem: smartphones. The advantages are two-fold: firstly, they already possess all the required hardware; secondly,

almost everyone owns own, which means no acquisition of equipment is necessary. Given these considerations, it makes sense to develop an app for smartphones to obtain position data and send it to the ground computer. Given the larger user base and its open-source nature, the app is targeted for Android systems.

The Android system accesses the location services, obtaining NMEA messages which are sent to the ground computer via the established connection. This connection is a TCP session, which guarantees that the packets are delivered in order and retransmitted if necessary. Given the blocking nature of data sending and receiving, the TCP client runs on its own thread. Another feature that is present is the implementation of an ID and login system, which requires the user to input a unique username upon first usage of the app. If the username is already in use, the user will be prompted to input a new one.

5. Ground Control Station Software

Ground Control Station (GCS) software refers to a software application, typically run on a ground computer, that communicates with the drone via wireless telemetry [11]. It displays real-time flight data and instruments, such as performance and position. Besides providing monitoring, a GCS can also be used to control a drone during flight, by uploading new mission commands and setting parameters. With GCS software, the ground computer interacts with the auto-pilot aboard the drone via commands, which are interpreted by the auto-pilot and acted upon. At the same time, the GCS receives data from the drone which can be used to monitor position, flight mode, speed, attitude, among others. There are existing GCS software available for download and installation and it is possible to use more than one together or develop codes to run in parallel with the existing GCS.

In order to use GCS software, certain requirements need to be met. Firstly, a connection needs to be established to the flight controller on-board the drone. As seen before, this is not attainable without an on-board computer to interface with the flight controller. Secondly, auto-pilot software needs to be installed on the drone. One such software is ArduPilot. It is an open-source auto-pilot suite that aims to enable the creation and use of autonomous, unmanned vehicle systems [12]. ArduPilot does not manufacture hardware but its auto-pilot firmwares work on many different boards to control several types of vehicle. For multi-rotors, that comes under the form of ArduCopter, with several flight modes. In addition, it also presents autonomous flight capabilities through the execution of scripted missions. Finally, comes communication. In order to communicate beyond line of sight, a broadband 3G/4G cellular network connection is required. Besides this, a protocol that both the GCS and the flight controller understand needs to be implemented. The MAVLink protocol is one such example and it is used by ArduPilot to communicate with the GCS.

5.1. The MAVLink Protocol

MAVLink is a serial protocol used to send data and commands between vehicles and ground stations and between on-board components [13]. The ground computer encodes commands with this message set that, upon arrival, are interpreted by the companion hardware and the actions executed. At the same time, data is encoded on other messages and sent from the drone to the ground computer, which decodes it. MAVLink messages can be issued over any serial connection and does not depend on the underlying technology. That means it is compatible with radio transmitters, WiFi and cellular networks. One of the downsides of this communication protocol is the fact that certain messages are not guaranteed to be delivered. As such, all the systems involved need to periodically check the status of the drone.

Given its importance for the tracking algorithm, the mission protocol implemented in MAVLink should be explored [14]. It allows a GCS to manage mission (flight plan), geofence and safe point information on a drone. It covers operations such as upload, download and clear missions, as well as messages to exchange mission items and commands. The protocol follows a client and server architecture. Operations and commands are initiated by the GCS, the client, and acknowledged and executed by the auto-pilot, the server. The mission protocol supports re-request of messages, allowing reliable transfer of information over a lossy link.

5.2. Tracking Algorithm

In order for the connection to the smartphone to be opened, a server needs to be implemented on the ground computer. This is achieved through TCP protocol. Since multiple clients will be connecting, and to avoid blocking, each client will be placed on its own thread. The data that is needed regards latitude, longitude, altitude and the geoid. This is found on NMEA GGA type messages. As such, only NMEA GGA messages are accepted to the parsing mechanism. In addition, since the attitude of the drone in relation to the target can be specified, calculations may have to be made to obtain the corrected coordinates. At this stage, the position data is finally ready to be used and is sent to the thread that manages the drone, called the GCS thread. This thread is connected to MAVProxy, which is used for networking.

On drone take-off command, issued through a

MAVLink COMMAND_LONG message, the first waypoint to follow is sent to the drone via the mission protocol described above. To start the mission, the flight mode is changed to Auto mode, thus starting the following procedure. Whenever the drone reaches the waypoint, a new waypoint needs to be sent. Given that ArduPilot's implementation of the MAVLink protocol does not allow for flight plan changes with the Auto flight mode, the flight mode is changed to Guided mode. Whenever the new flight plan is uploaded, Auto mode is engaged once more. Finally, whenever the tracking procedure is finalized, the drone is ordered to return to the Launch Point (LP).

5.3. Obstacle Avoidance Post Mission

Returning to the Launch Point is not as direct as it appears. In theory, it is as simple as sending the coordinates of the Launch Point to the drone in the format of a waypoint. However, since the drone might have no knowledge of what lies in the line between the two points, it may crash against obstacles. Another solution would be to keep the path taken until the point in time when Return Home is triggered. Still, while a safe option, it is not optimal as there may be another safe, previously explored and shorter path to the Launch Point. Among the data sent to the GCS are the coordinates of the drone's position, which can be used to create a network of safe paths that have been previously visited by the drone. This network is made of tracks - lines that connect two or more successive positions occupied by the drone - and nodes - the first and last points that comprise a track and the result of an intersection between tracks

When processing MAVLink position messages, every consecutive point is, in reality, processed as a line segment connecting the previous received point, $p-1$, and the current one, p. This is done in order to detect intersections with existing tracks in the network.

If there are no intersections, p is added to the new track list. Then, it is verified if it is possible to decimate or down-sample the new track. In this case, decimation happens if the last three points to enter the new track list, including p, are considered collinear, that is, it is possible to unite them all with a single, straight line. If they are, the point in the middle can be removed without impacting the track shape while, at the same time, reducing the number of waypoints that will have to be sent to the drone if this track is to be used to return to the Launch Point. Finally, the track is carried over to the next iteration as the new track list.

If there are intersections, the existing tracks are broken down into smaller tracks, with the intersection points becoming nodes of those smaller tracks. These new nodes are also the nodes of the tracks that form from uniting every two consecutive intersection points. The first intersection point is added to the new track list that existed in the beginning of this iteration (until $p-1$), and it is checked if the last three points can be decimated, as described above. Whatever the result, that track, decimated or not, is added to the network. Finally, the track from the last intersection point to p is carried over to the next iteration as the new track list.

Whenever the Return Home Protocol is engaged, the network that exists up to that point is searched for the shortest path to the Launch Point.

5.4. The Return Home Protocol

As stated before, whenever the battery falls below the threshold level, the GCS will issue a command to the drone to return to the Launch Point. This functionality is similar to the Return to Launch (RTL) and Smart RTL modes already present in ArduPilot. The RTL mode, when activated, will fly in a straight line from the current position to the Launch Point [15]. The Smart RTL mode differs from RTL as it uses the flight data from that session, being defined as all the data generated between take-off and landing, to find the shortest path to the Launch Point from the points already flown over during the session [16]. It provides a safer path to return home than the original RTL mode. Still, since it solely relies on the data from the current session, a shortest and previously explored path might be available that is not known to the system, as information of prior sessions is not used.

Based on this analysis and using the concepts introduced before, a new protocol to return home was developed, which attempts to make up for the short comings of the existing modes. It utilizes the safe path network that is generated and updated every time there is a flight session and existing Search Algorithms to find the shortest path to the Launch Point among all the paths that have previously been visited. It has been named the Return Home Protocol. Whenever this protocol is engaged, the safe path network that exists up to that moment is searched using the A* Algorithm. This algorithm is an example of an informed search strategy and a best-first search [17]. Since a priority queue is used, as soon as the goal state is achieved, the algorithm has found the lowest cost path, in this case shortest distance, to the goal state and the search stops. The points that make up the tracks the drone needs to follow are sent to the platform as waypoints.

6. Results

6.1. Drone Testing

After the drone was fully assembled, all the internal systems were calibrated. After this calibration was done, the flight modes were verified. Manual

flight was then tested as well and used with Altitude Hold (Alt-Hold) mode to verify the maximum achievable autonomy with both batteries. This was done under favourable meteorological conditions, in order to impact the final value as little as possible. The results are summarized on table 2.

Table 2: Autonomy Comparison

Cap. (Ah)	$A_{Predicted}$ (min)	A_{Real} (min)	Error (%)
2.2	8.62	7.36	14.6
5.8	20.18	19.67	2.5

As seen, the values obtained are bellow the predicted autonomies, which goes in line with the expected higher energy consumption due to other components. They are, however, good results and match the predictions of the model.

Some of the drone's characteristics are summarized on table 3.

Table 3: Drone Characteristics

Maximum Weight (kg)	1.572
Available Thrust (kg)	3.632
Maximum Autonomy (min)	19.67
Throttle @ Take-Off (%)	41.0
Climb Rate @ Take-Off (m/s)	1.5
Maximum Speed (m/s)	6.6

6.2. Android App Testing

The app developed was tested on different smartphones, in order to verify if it worked across multiple models. The first test regarded obtaining position data at different speeds around a well-known shape. This was achieved by adopting two different walking paces, one faster than the other, and a running pace. All smartphones output correct position information with all speeds. However, the running pace provided the more accurate, likely due to the higher velocity improving the position estimation.

The second test involved placing the smartphones on a car and moving around the roads collecting position data. Position acquisition was successful and part of the data collected is shown on figure 5.

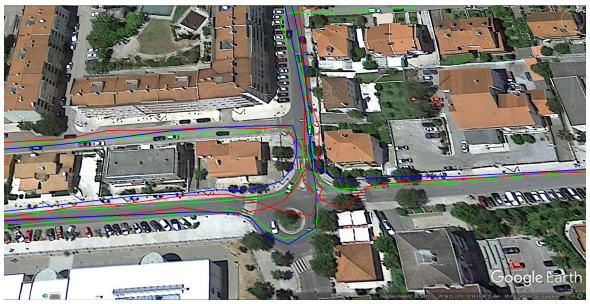

Figure 5: Track Details (Red - Huawei Y6, Green - Samsung Galaxy J6, Blue - Huawei P10 Lite)

The phenomena mentioned before can also be observed, where, on approximation to a roundabout, the car slowed down, which worsened the estimation, especially for the Huawei Y6. Still, the app worked as intended during all tests.

6.3. GCS Program Testing

The GCS program developed was tested under a simulation environment, using ArduPilot's implementation of Software In The Loop (SITL), and on the field. Simulation was performed using the real data collected from the previous app test. Three different tests were ran, each with the drone occupying a different position around the target through attitude specification: flight above the target (azimuthal angle equal to 0 degrees), on the target's left (azimuthal angle equal to -90 degrees) and on the target's right (azimuthal angle of 90 degrees). In addition, the horizontal distance between the target and the drone was fixed at 10 metres. On figure 6, the trajectories described during testing can be seen.

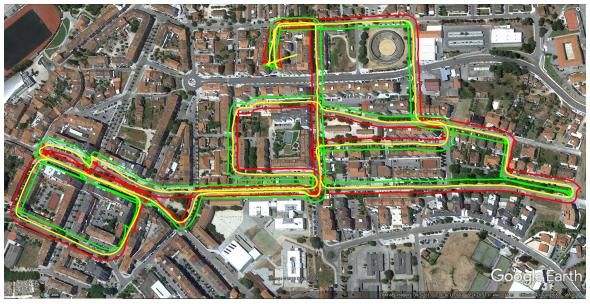

Figure 6: Trajectory comparison: Yellow (az = 0 deg, el = 45 deg, h = 10 m) vs Red (az = -90 deg, el = 45 deg, h = 10 m) vs Green (az = 90 deg, el = 45 deg, h = 10 m)

In order to analyse if the drone managed to closely follow the reference, the graphic of figure 7 was plotted, depicting the distance to the most recent reference position available.

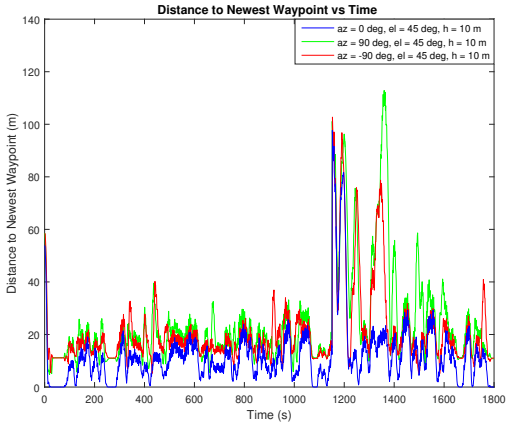

Figure 7: Distance to Newest Waypoint Through Time

The first observation is that the drone manages to closely follow the reference. However, and in line

with what is observed on curves on figure 6, whenever there is a turn, which implies lower velocity, the drone tends to fall behind. This is caused by a lack of updates on the direction of the reference, a necessary value for the calculation of the position to occupy. In fact, the drone occupies the wrong position for a while before the direction is updated, which causes backtracking and slowing down. It is made worse by the value of the horizontal distance. On average, the drone still manages to stay within 15 metres of the reference. Another set of three simulations was run with a smaller horizontal distance of 5 metres. Figure 8 shows the distance to the newest reference position throughout time.

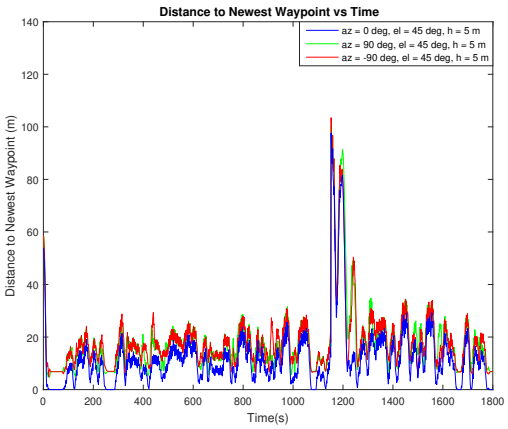

Figure 8: Distance to Newest Waypoint Through Time

As expected, a smaller distance between the position of the reference and the position the drone needs to occupy during flight reduces the deviation caused by the lower update rate of the reference's direction, which means the drone will have to backtrack less, curves will be performed more smoothly, faster and more efficiently from an energy point of view, as less corrections will need to be done. On average, the drone, again, manages to stay within 15 metres of the reference.

After these simulations were ran, the drone was tested on the field, specifically at Valdonas Aerodrome (Lat = 39.59 N, Lon = 8.37 W). Navigation in a confined area poses bigger challenges given the way ArduCopter handles waypoint navigation. In addition, the drone is capable of a top speed of 23.76 km/h, which makes it capable of tracking a running target. As such, tracking of a slow target was tested. For this test, the helicopter landing area was taken advantage of, with the reference walking in the H shape. The trajectories of this test are shown on figure 9. On figure 10, the distance to the newest waypoint throughout time is represented.

As it can be seen, the drone manages to stay within reach of the target, with the peaks corresponding to the target moving away from the

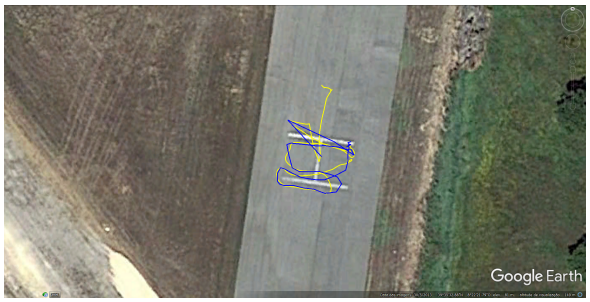

Figure 9: Reference Trajectory (Blue) vs Drone Trajectory (Yellow)

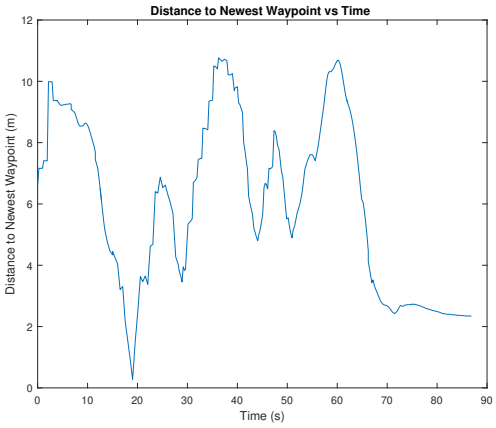

Figure 10: Distance to Newest Waypoint Through Time

drone while the drone is still chasing a sequence of waypoints in curves. As expected, the drone's speed drops, meaning that it will take more time to achieve a point when the waypoint is considered as completed. Despite these observations, with a mean distance to the target's latest position of 6.05 metres and a maximum distance of 10.78 metres, the results are satisfactory.

6.4. Safe Path Mapping

Given the complexity of the algorithm implemented to generate the path network and the necessity to long term test the solution, this part of the work was tested only in simulations. A new track was gathered from a smartphone, which is shown on figure 11.

Figure 11: Undecimated Track, opened on Google Earth

The first test was to the decimation algorithm and the threshold values used to eliminate unnecessary points, in particular the value for which two line segments are considered to be collinear. The results from this test are shown on table 4, which characterizes the point reduction for each value in terms of number, the total track length and the mean Dynamic Time Warping (DTW) distance. DTW is an algorithm used to analyse time series, in particular for measuring similarity between two temporal sequences, which may vary in speed [18].

Table 4: Represented Tracks Information

Thld (deg)	Points	$Length_{Track}$ (m)	$Distance_{DTW}$ (m)
0.00	2396	6656	0.28
0.50	1107	6656	0.33
1.00	740	6655	0.35
1.50	577	6655	0.40
2.00	475	6655	0.44
2.50	403	6654	0.51
3.00	357	6653	0.59
5.00	235	6649	0.97
10.00	127	6634	4.68

As expected, the higher the threshold value, the more points are decimated. However, by doing this, the track loses shape and details can be lost. On some cases, such as when the drone drifts off due to wind, this is a positive aspect; but on others, such as obstacle evasion, a larger threshold will cause these details to be lost. As such, the choice of what threshold value to use is a compromise between eliminating more points and losing track shape. This is also dependent on the original track shape, which is difficult to predict. The DTW results show a good degree of similarity for all the thresholds except 10.00 degrees. For the subsequent tests, a threshold of 2.50 degrees will be used.

Finally, the creation and update of the safe path network is tested. To do this, the two paths of figure 12 are ran. The starting state is empty, that is, the databases are empty. The first path to be added is the green path, from Point 1 to Point 2. Since the database is empty, no intersections are detected, the track is added to the track database and its first and last points are added to the node database as nodes N0 and N1. The second path, in red, from Point 3 to Point 4 is ran. There are clearly intersection points: 4 points of single intersection, 1 arrival node and 1 departure node, which is N1. That means 7 new nodes need to be added to the database, the extremities of the second track and all the other intersection points. The expanded network includes 8 nodes and 11 tracks and is shown on figure 13.

Using the RTL mode to travel from N7 to N8, the drone could crash against the obstacles present on the sawtooth. If, instead, the Smart RTL mode is used, the drone would fly the same path back, as there is no knowledge of the previously explored path in red. This is a safe option but not an optimal solution, as a shorter path could be computed. This is where the safe path network becomes useful, flying back through nodes N7-N6-N5-N4-N3-N2-N1-N8, ignoring the sawtooth paths.

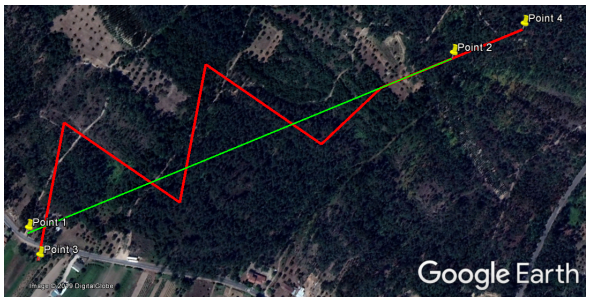

Figure 12: Paths to be Optimized, opened on Google Earth

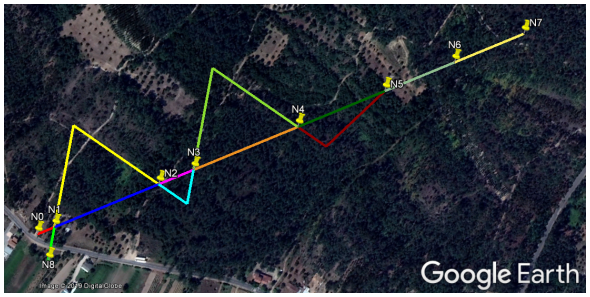

Figure 13: Optimized Paths, opened on Google Earth

The final test involves processing the track from figure 6 into the network. Applying the algorithm transforms one track into a network of 23 tracks and 15 nodes, reducing the number of points from 7513 to 327, a reduction of 95.6 %. The result is shown on figure 14, where each colour represents a different track.

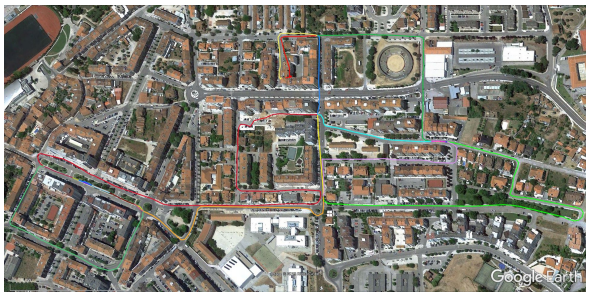

Figure 14: Resulting Network, opened on Google Earth

7. Conclusions

The main objectives for this project were: designing a drone, acquiring user position and tracking the target, according to the mission specifications.

Regarding projecting and building a drone, several drone configurations were analysed and, based on the mission specifications desired, one was selected. A process for the design and component selection was presented, which was followed to choose

the equipment to purchase. Given the testing done, both in manual and autonomous flight, and the good behaviour displayed by the drone, the design process has produced satisfactory results and it is considered valid for other future drone systems. In addition, a model for predicting drone autonomy based on data from component manufacturers was developed, with satisfactory results.

Regarding the development of a mechanism to acquire user position, it was chosen to develop an Android app to acquire user position based on GNSS and transmit that position to a server for processing. It takes advantage of the internal GNSS chip and of the possibility of establishing TCP links to achieve the objective of acquiring user position. As seen, it runs on several smartphones and its accuracy is reasonable across all models.

Regarding the development of a system to track the reference, an algorithm was developed, which takes advantage of existing, open-source projects and frameworks as its basis, namely ArduPilot and MAVLink. Position data is received via TCP from the smartphone, processed and forwarded to the drone under the form of waypoints to fly to, using the mission and communication protocols implemented by MAVLink. By using this successively, tracking is achieved, with good results.

Finally, the post mission flight, which is prone to result in collisions with foreign objects, was analysed, producing a safe path network that can be used to plot the best course home based on flight data collected over time. The algorithm developed brings up some very relevant points in its favour, maximizing the usage of flight data post mission and being versatile. With the right interfacing, it can be used on every mission, by every drone. This means that, with every flight, data is accumulated which later can be used to achieve increasingly safer paths to return to the Launch Point.

This project, while achieving the objectives set, has a lot of potential for further advancements. ArduPilot should be investigated better to improve tracking. Another idea for follow-up work would be to integrate a camera into the system with its own reference tracking algorithm. The inclusion of additional sensors, such as LiDAR to perform altitude control, is another suggestion. Finally, the creation of a drone hybrid would be an interesting way of increasing autonomy and versatility.

Acknowledgements

The author would like to thank Instituto de Telecomunicações (IT) for financially contributing to this project and allowing the usage of its laboratories in Instituto Superior Técnico (IST); Aeroclube de Valdonas for allowing the usage of its airfield for testing; and António Raimundo for his contribution to the project, namely helping to build the drone.

References

[1] Brett Holtman. The first air bomb: Venice, 15 July 1849. Airminded website, August 2009. `airminded.org/2009/08/22/the-first-air-bomb-venice-15-july-1849/`, accessed on March 27th.

[2] Luke Dormehl. The history of drones in 10 milestones. `www.digitaltrends.com/cool-tech/history-of-drones/`, September 2018. Online, last accessed on March 27th.

[3] European Global Navigation Sattelite Systems Agency. GNSS Market Report 2017, Issue 5. Luxembourg Publications Office of the European Union, 2017. ISBN 978-92-9206-032-9.

[4] Andrew Chapman. Types of Drones: Multi-Rotor vs Fixed-Wing vs Single Rotor vs Hybrid VTOL. `www.auav.com.au/articles/drone-types/`. Online, accessed on March 27th.

[5] Finton Corrigan. Quick Drone Parts Overview Along With Handy DIY Tips. `www.dronezon.com/learn-about-drones-quadcopters/drone-components-parts-overview-with-tips/`. Online, accessed on March 27th.

[6] Thomas Corke. *Design of Aircraft*, chapter 1, pages 15 – 18. Pearson Education, 2003.

[7] MIT Electric Vehicle Team. A Guide to Understanding Battery Specifications. `web.mit.edu/evt/summary_battery_specifications.pdf`, December 2008. Online, accessed on March 27th.

[8] Rita Vallejo, José Sanguino, and António Rodrigues. Posicionamento com GNSS em cenários de multi-constelação. In *8º Congresso do Comité Português Da URSI - Drones e veículos autónomos: desafios do presente e do futuro*, 2014.

[9] National Maritime Electronics Association. NMEA 0183 Standard v4.11. `www.nmea.org/content/nmea_standards/v411.asp`. Online, accessed on March 27th.

[10] José Sanguino. The NMEA 0183 Standard - Navigation Systems notes. `fenix.tecnico.ulisboa.pt/downloadFile/1970943312277120/The%20NMEA%200183%20Standard.pdf`. Online, accessed on March 27th.

[11] ArduPilot. Choosing a Ground Station. `ardupilot.org/rover/docs/common-choosing-a-ground-station.html`. Online, accessed on March 27th.

[12] ArduPilot. ArduPilot Overview. `ardupilot.org/ardupilot/`. Online, accessed on March 27th.

[13] MAVLink. MAVLink Developer Guide. `mavlink.io/en/`. Online, accessed on March 27th.

[14] MAVLink. Mission Protocol. `mavlink.io/en/services/mission.html`. Online, accessed on March 27th.

[15] ArduPilot. RTL Mode. `ardupilot.org/copter/docs/rtl-mode.html`. Online, accessed on March 28th.

[16] ArduPilot. Smart RTL Mode. `ardupilot.org/copter/docs/smartrtl-mode.html`. Online, accessed on March 28th.

[17] Stephen Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter 3, pages 92–108. Prentice Hall, 2003.

[18] Pavel Senin. Dynamic Time Warping Algorithm Review. `www.researchgate.net/publication/228785661_Dynamic_Time_Warping_Algorithm_Review`, December 2008. Online, accessed on March 27th.