drones

MDPI

Article

Improving UAV Mission Quality and Safety through Topographic Awareness

Jamie Wubben *,†, Christian Morales †, Carlos T. Calafate †, Enrique Hernández-Orallo †, Juan-Carlos Cano † and Pietro Manzoni †

Computer Engineering Department (DISCA), Universitat Politècnica de València (UPV), 46022 Valencia, Spain; chmogue@inf.upv.es (C.M.); calafate@disca.upv.es (C.T.C.); ehernandez@disca.upv.es (E.H.-O.); jucano@disca.upv.es (J.-C.C.); pmanzoni@disca.upv.es (P.M.)

* Correspondence: jwubben@disca.upv.es

† These authors contributed equally to this work.

Abstract: The field of Unmanned Aerial Vehicles (UAVs) has progressed greatly in the last years. UAVs are now used for many applications and are often flown automatically. One commonly implemented feature in an automatic flight is that of following a mission at a stable altitude. However, this altitude is almost always referenced from the take-off location and does not take terrain profile levels into account. This is a critical and dangerous issue because if the terrain level changes abruptly (e.g., mountain regions or buildings in a city), this can lead to crashes or an unintended (illegal) high altitude. Our aim for this work is to provide a solution such that a constant altitude above ground level is maintained. To this end, we make use of the readily available Digital Elevation Models (DEMs). These models, which contain the terrain elevation, help us in dynamically adjusting the VTOL UAV altitude so that it remains nearly constant in relation to the ground. Results have shown that with the use of our method, the altitude can be maintained sufficiently constant while introducing a limited increase in flight time and battery consumption that is proportional to the terrain's irregularity. In a moderately changing terrain, the error could be reduced to just ± 5 m.

Keywords: VTOL UAVs; navigation optimization; digital elevation models; on-board; real time processing

Citation: Wubben, J.; Morales, C.; T. Calafate, C.; Hernández-Orallo, E.; Cano, J.-C.; Manzoni, P. Improving UAV Mission Quality and Safety through Topographic Awareness. *Drones* **2022**, *6*, 74. https://doi.org/10.3390/drones6030074

Academic Editor: Abdessattar Abdelkefi

Received: 10 February 2022
Accepted: 9 March 2022
Published: 11 March 2022

1. Introduction

In the last decade, interest in the use of Unmanned Aerial Vehicles (UAVs) has grown steadily. Nowadays, UAVs, colloquially called drones, are a common tool for numerous applications such as site inspection, mapping, security, agriculture, filming, racing, and many more [1].

In many applications, the UAV flies automatically from one Global Positioning System (GPS) waypoint to another. Throughout its path, which is often defined using mission waypoints, the UAV has to maintain a steady altitude. This is a common built-in feature in many higher-end UAVs, and it is provided by the flight controller, i.e., a small on-board computer with multiple sensors that regulates the speed of the propellers to accomplish its flight goals.

The most commonly used open-source flight controller is called Pixhawk [2], which often uses the Ardupilot firmware [3]. Through the use of Ardupilot (or any other flight controller firmware for that matter), we can maintain a constant altitude. However, this constant altitude is referenced from the take-off point of the UAV (i.e., from the origin). As shown in Figure 1, this is problematic if the terrain is not flat. This is because if the terrain rises, there is a chance that the UAV will crash into the ground. On the other hand, if the terrain descends, the drone will be flying at a higher altitude; this can decrease the performance of the UAV sensors (e.g., camera feeds) or cause it to surpass the legal constraint regarding maximum flight altitude (for example, in Europe, it is at 120 m [4]).

Therefore, many UAV applications require the maintenance of a constant Above Ground Level (AGL) altitude. In this work, we focus on providing this functionality because it is currently not implemented in commercially available flight controllers. In particular, our solution is based on Digital Elevation Models (DEMs), i.e., files with topographical data. Through the use of these readily available terrain elevation models, we can manipulate the drone in such a way that a constant altitude above ground level is consistently maintained. We must point out that third-party applications that consider the topography in mission planning already exist (e.g., FlyLitchi [5]). However, many applications require drones to be capable of making ad hoc decisions and deviating from the original flight plan (e.g., to avoid collisions or due to a change in the itinerary). Therefore, in this work, we opted for a more holistic approach so that we could grant current and future applications this freedom in addition to simplifying the operator's tasks.

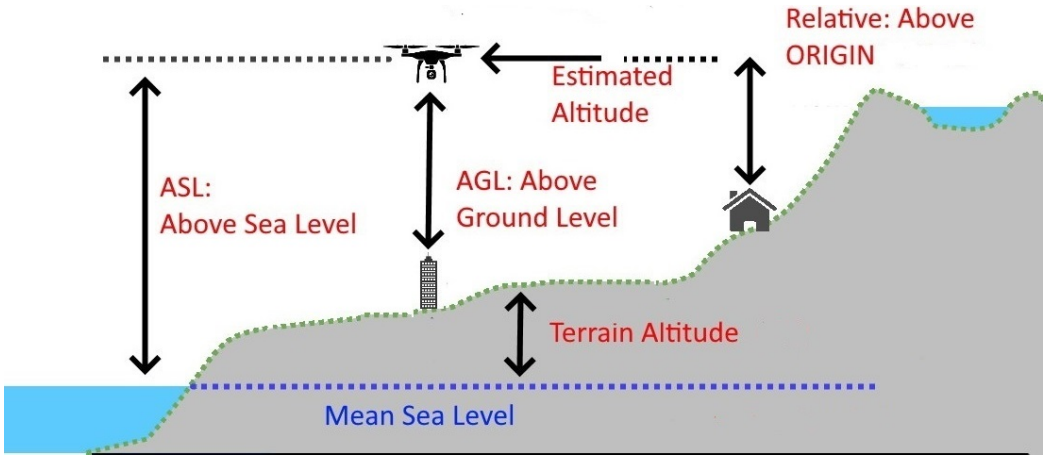


Figure 1. The different types of altitude.

This research work is related to two main areas: drones and topography. A close relationship has been established between these areas as there are numerous applications in which the use of drones allowed for the improvement of topography and the accuracy of elevation models, as can be seen in [6].

As for research dedicated to reversing this relationship, i.e., using topography to aid drone flights, this is still scarce. The main use of topography in an autonomous flight is to be able to trace routes and avoid collisions. For the time being, its main field of application consists of pre-programmed flights with fixed and constant altitudes, but without considering obstacles. This simplified approach is applicable in many cases where the terrain is mostly flat, and it is widely used in applications such as precision agriculture [7].

Nevertheless, fully autonomous flights where aircraft are endowed with decision-making and route selection capabilities are not far from becoming a reality. We can already find articles such as [8] that talk about its potential and viability as a means of delivery. Other works such as [9] defend their usefulness as a blood-transport solution in emergency situations, given the speed and efficiency of these aircraft.

Furthermore, drones as a means of transportation are close to being a reality, and topography is the key element to help achieve it. One of the first research works incorporating topography into autonomous collision avoidance was described in [10]. This research departed from a theoretical framework from IEEE [10], and it was validated by the US Air Force on an F-16 [11,12].

In [13], Muliadi et al. proposed an artificial neural network that could be used as a UAV altitude controller. They compared their method with the traditional PID control system. The traditional PID control system involves complex expressions, which are difficult to solve analytically. Their approach overcame this problem, and as they showed (in simulation), with the use of their neural network, a better performance could be achieved.

The research that served as the theoretical basis of that project can be found in [14]. Based on that work, our paper studies and establishes, through mathematical formulations, the feasibility of using elevation models as a data source for autonomous drone flight.

The rest of this work is organized as follows: In Section 2, we explain the format of DEMs, where they can be downloaded, and what they represent. In Section 3, we explain our proposed solution in detail. We then evaluate our proposed solution in Section 4, where we simulate a wide set of use cases, including a challenging mountainous region. Finally, in Section 5, we conclude by summarizing our findings and suggest promising future works.

2. Digital Elevation Models (DEMs)

A Digital Elevation Model (DEM) consists of a database which contains the height above sea level of a given location. A DEM is characterized by its versatility in terms of accuracy/compression. Despite the existence of different formats, the most common one divides a land area into regular-sized squares. For each square, the highest altitude is measured and saved into a large matrix. In the header of the DEM file, the GPS coordinates of the origin as well as the size of each square (also called the resolution) are provided. Since it is easy to retrieve the row and column number when reading the DEM file, the GPS location of each cell can be easily calculated (i.e., by adding the offset from the origin). In this manner, a relatively large land area can be contained in a small file. The size of the DEM file will depend on the resolution; however, to provide a general idea, using a (high) resolution of 4 m^2 for each square, an area of 500 km^2 can be stored in a file of only 250 MB. Considering that these storage requirements are not a problem nowadays, and that VTOL UAVs can only fly for a few kilometers anyway, size is not deemed to be a problem.

DEM files are readily available on the Internet. For our research, we used the DEMs provided by the Spanish government [15]. As one might observe, on the government site (and in general), two types of DEMs exist (see Figure 2): the Digital Terrain Model (DTM), which makes use of the ground level, and the Digital Surface Model (DSM), which also includes buildings, trees, etc. It is obvious that the DSMs represent the real world more accurately (especially in cities), but also become obsolete much faster. Since the file formats of the DTM and DSM files are very similar, our approach works for both DTM and DSM files. Nevertheless, for our experiments, we chose to work only with DTM files.

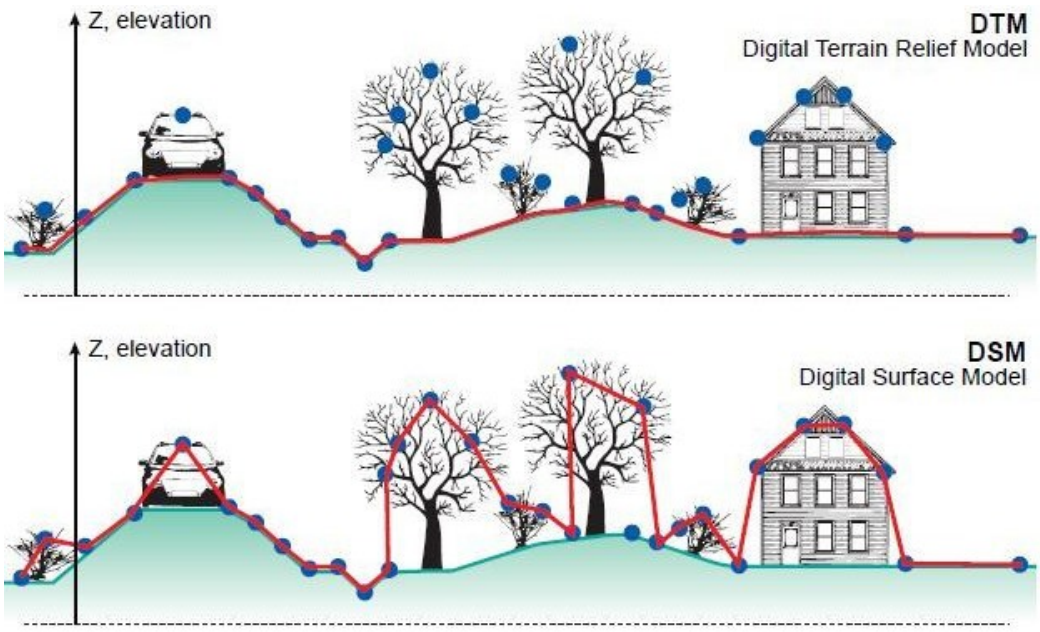


Figure 2. The difference between DTM and DSM; source: [16].

As stated before, in order to create a DEM, the terrain elevation (w.r.t sea level) needs to be measured. There are many methods of measuring terrain elevation, but here, we will only highlight a few of them:

1. **Conventional topographic surveys:** For these surveys, conventional equipment such as optical or laser tachometers and leveling instruments are used. With this method, a highly detailed DEM can be obtained. However, it is a lengthy and costly process, so it is only used for small areas.
2. **Kinematic GPS surveys:** For this method, a GPS receiver is mounted on a vehicle, which moves over a terrain. With the use of the altitude data from the GPS, a relatively large area can be covered.
3. **Optical satellite images:** With the use of satellite images, an estimate of the terrain level altitude can be made. The advantage of this method is that it covers a huge area more quickly; however, the resolution is much lower than the previously mentioned methods.

3. Proposed Solution

The principal objective of our work is to develop a framework that allows a VTOL UAV to maintain a constant altitude above ground level. This is important because if disregarded, the UAV might crash into the ground if the ground level rises. On the other hand, if the ground level decreases, the relative altitude of the UAV will increase. This higher relative altitude can influence the performance of the sensors on the UAV (e.g., camera feed), or the UAV can surpass the legally allowed altitude (120 m in Europe [4] and most countries worldwide).

To address and settle this problem, we propose a solution that allows a VTOL UAV to automatically ascend or descend while it is following a mission so as to adapt to the terrain profile. Before the start of the flight, the mission file (which consists of GPS waypoints) and the DEM file are uploaded to the UAV. During the flight, the UAV makes use of the mission file to determine its direction, and the DEM file to adjust its altitude. It is important to note that the change of altitude is calculated in real time and not before the start of the mission. This approach allows us to (i) spend less time on the calculation at start-up, (ii) make changes to the mission without worrying about the terrain profile, and (iii) make our method compatible with other (future) approaches, which use sensors instead of DEM files to determine the altitude relative to the ground.

3.1. Model

In this work, we propose a control system that regulates the velocity of the UAV in the z axis (or up/down axis). This control system is aimed at minimizing the error between the desired and current flight altitudes, thus keeping the UAV altitude constant with respect to the ground. The terrain elevation, which comes from the DEM file, is used as the reference signal. The relative altitude of the drone is determined by the flight controller, which relies on GNSS and barometric data. The error, which we try to minimize, is determined by the difference between the reference altitude and the relative altitude of the drone:

$$e(t) = reference\_altitude(t) - relative\_altitude(t) \quad (1)$$

The task of our controller is to calculate an output such that this error is minimized. In our system, the output of the controller is the vertical velocity (m/s). In theory, those values can range from $-\infty$ to $+\infty$; in practice, however, the electrical motors have a maximum speed which limits thrust. Of course, different motors have different maximum speeds. Nevertheless, as long as we are not in the drone racing business, most operators prefer a (relatively) slow vertical velocity, mainly for safety and control reasons. Therefore, we limited the output of our control to the range of ± 5 m/s.

A general PID controller can be described through the following equation:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{2}$$

The output signal $u(t)$ is thus given by the sum of three terms, which all depend on the error signal: (1) the proportional term (P), which corrects for the current error; (2) the integral term (I), which grows over time and can thus ensure that the steady state error becomes zero; and (3) the derivative term (D), which allows for the prediction of a future error and thus increases the stability of the system. Since the ground level (i.e., the reference signal) in our application will change over time, we do not have to be concerned about a steady-state error. Therefore, we removed the integral term (by setting $K_i = 0$), which allowed us to simplify Equation (2) to a PD controller, as follows:

$$u(t) = K_p \cdot e(t) + K_d \frac{de(t)}{dt} \tag{3}$$

The equation above is continuous; however, a digital system can never be continuous, and we are thus forced to discretize the equation. For our drone (and all drones alike that use ArduCopter [3]), the vertical velocity vector is updated every 200 ms. Thus, the controller equation can be discretized considering this sampling period (T_s). Furthermore, we change the naming—from the generic output signal, u, to the more appropriate name, v_{ver}—in order to clarify that we use a PD controller to control the vertical velocity.

$$v_{ver}[t] = K_p \cdot e[t] + K_d \frac{e[t] - e[t-1]}{T_s}, \qquad T_s = 200\,\text{ms} \tag{4}$$

At this point, we have managed to design the vertical controller that allows the UAV to maintain its altitude. However, we also need the UAV to move towards its targets, i.e., to follow the planned mission. The current target (or waypoint) is given in GPS coordinates. In order to move the UAV towards the target, we simply need to define a direction vector, $\overrightarrow{u}$, that starts at the drone and ends at the target. After normalization, this vector can be scaled such that the UAV moves at the planned (or max) speed (in our case, max_v_{hor} = 15 m/s, as this is the default setting in the flight controller). Thus, we have the following expression to obtain the horizontal velocity vector $\overrightarrow{v_{hor}}$:

$$\begin{aligned} \overrightarrow{u} &= \overrightarrow{target} - \overrightarrow{UAV} \\ \hat{u} &:= \frac{\overrightarrow{u}}{||\overrightarrow{u}||} \\ \overrightarrow{v_{hor}} &= max\_v_{hor} \cdot \overrightarrow{u} \end{aligned} \tag{5}$$

Given the right parameters, this should be enough to move the UAV from its current position to its target while adjusting its altitude so that its relative altitude to the ground remains as constant as possible. However, during our simulated tests, we observed two issues: firstly, when the change in terrain height is significant, the UAV needs more time to ascend/descend; otherwise, a crash might still occur. In practice this means that if the velocity in the z axis is close to its limit, max_v_{ver} (5 m/s in our proposal), we need to stop moving towards the target to avoid a likely collision. Secondly, in Equation (1), we denoted the error signal as the difference between the reference altitude and the real altitude, at the same time instance. During the actual flight, as we will show later, an improvement can be made by looking ahead to where the UAV is aiming to go, and then take that altitude as reference. These small adjustments have been implemented in our proposal, as shown in Figure 3.

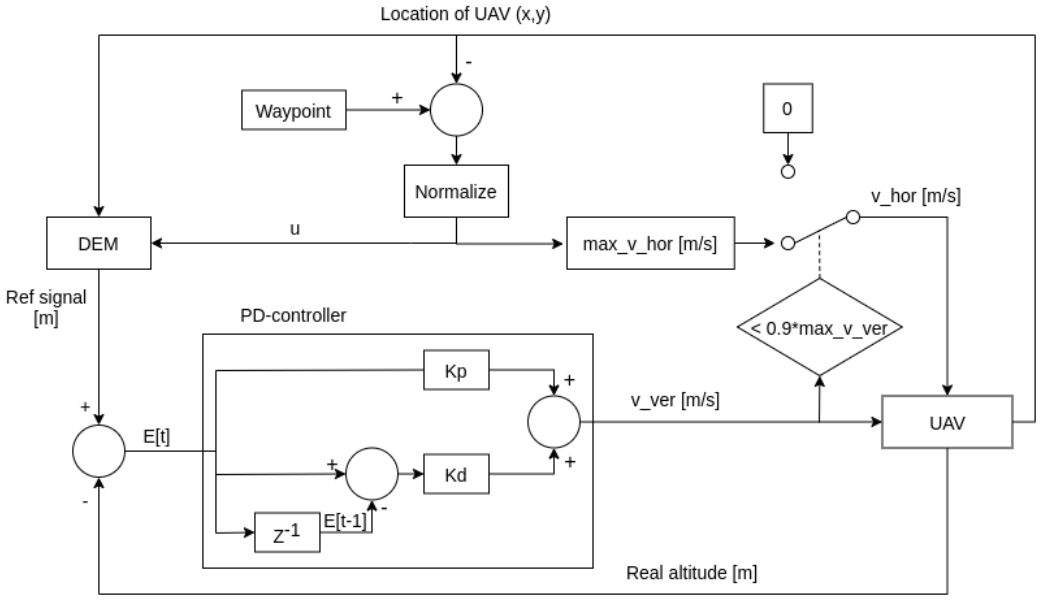


Figure 3. Block diagram of the proposed method.

The flight controller of the UAV (block on the right) provides us with the current (absolute) altitude of the drone. The difference between the reference signal and the current altitude denotes the error signal. This error signal is fed into the PD controller, which in turn calculates the vertical velocity needed to minimize the error signal. The vertical velocity signal also controls whether the horizontal velocity is set to 0 m/s or not. In case the vertical velocity is high (more than 90% of the maximum allowed), the switch (on the right side of the block diagram) will toggle and set the horizontal velocity to zero. This gives the UAV the required time to adjust to higher altitude differences before moving towards the waypoint again. Notice that we intended to proceed with the planned mission as quickly as possible, only reducing horizontal speed when strictly necessary. The flight controller also provides us with the location of the UAV. This location, together with the location of the waypoint, is used to calculate the direction vector, $\overrightarrow{u}$. The horizontal velocity vector is obtained by scaling the direction vector, $\overrightarrow{u}$, with the appropriate horizontal speed. Finally, the DEM block calculates the reference signal. This calculation is based on the DEM file, which contains the absolute altitude level for a given location. We used the current location of the UAV as well as its direction vector, $\overrightarrow{u}$, to calculate its future position. The ground level altitude of this future position is fetched from the DEM file, and the reference signal is calculated by adding the desired above ground level offset.

3.2. Parameters

In our model, there are five parameters: the proportional gain K_p, the derivative gain K_d, the look ahead distance (m), the maximum horizontal velocity (max_v_{hor}), and the maximum vertical velocity (max_v_{ver}). Hence, we need to find the optimal values for these parameters, as the chosen values will determine how fast the drone responds to changes as well as the stability of the system. We decided to set the two maximum speed parameters to 15 m/s for the horizontal speed and 5 m/s for the vertical speed. As previously stated, these values are a good compromise between safety and speed, and they are often used in VTOL drone applications. This section will explain how we adjusted the values for the other three parameters. Our tuning aimed at meeting our specific goals, which include achieving an adequate trade-off between the vertical error (overshooting included) and the mission time. It is important to note that, depending on the characteristics of the UAV, these values might vary. Thus, the fine-tuning of these parameters for other types of UAVs will be necessary. Auto-tuning methods do exist, but they require deriving a mathematical model for the UAV, which we currently do not have. Therefore, we will estimate the values

for our parameters based on some commonly known rules. In this section, we will explain the process we went through so that results can be replicated.

First, it must be noted that determining the optimal parameter values based on the simulation of an entire mission is impractical. This is due to various reasons:

1. Simulating an entire mission takes a long time;
2. It is difficult to determine the true effect of a change in the parameters;
3. One can be fixed on solving a specific problem (i.e., the mission), instead of the solution being applicable to different missions (i.e., parameters are optimized for just one mission instead of a general solution);
4. When the entire mission is considered, the horizontal movement of the UAV also plays a role.

Therefore, we obtained the parameter values using a classic control theory technique, i.e., by observing the step response of the system. The step response is the response (i.e., the output) of the system when a step (i.e., sudden change in the reference signal) is applied. Such step response is then plotted on a graph (see Figure 4), and it is common to normalize the output. Using this graph, we can determine two important values: (i) the rise time and (ii) the overshoot. The rise time explains how fast the system will converge at the new reference signal. The overshoot refers to the output exceeding the new desired reference signal. In our application, both metrics are important. We do not want a high overshoot (or undershoot) because this will cause the UAV to fly too high (which might violate legislation) or too low (which might result in a crash). However, the response of the UAV should also be sufficiently quick (i.e., short rise time) for those cases where the terrain profile is constantly changing, and the UAV should be able to keep up with that change.

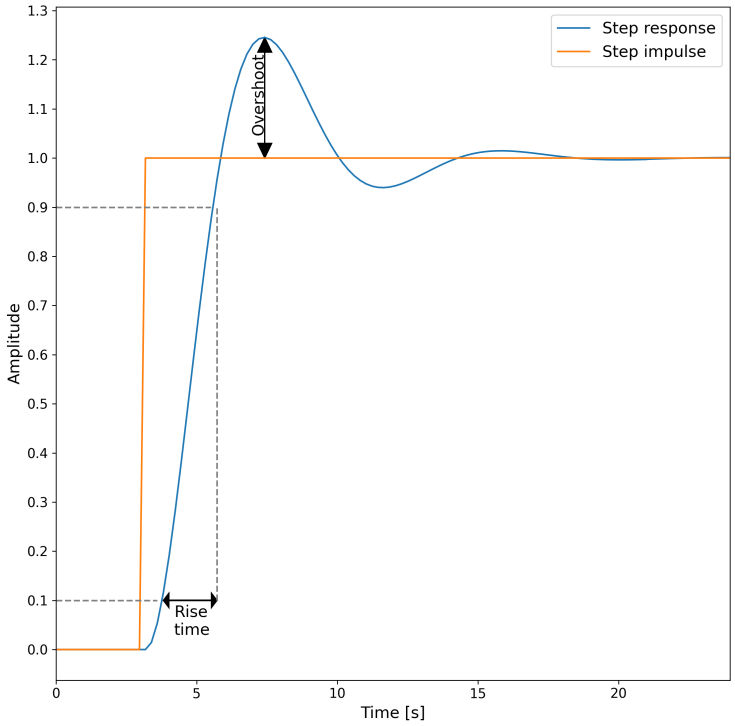
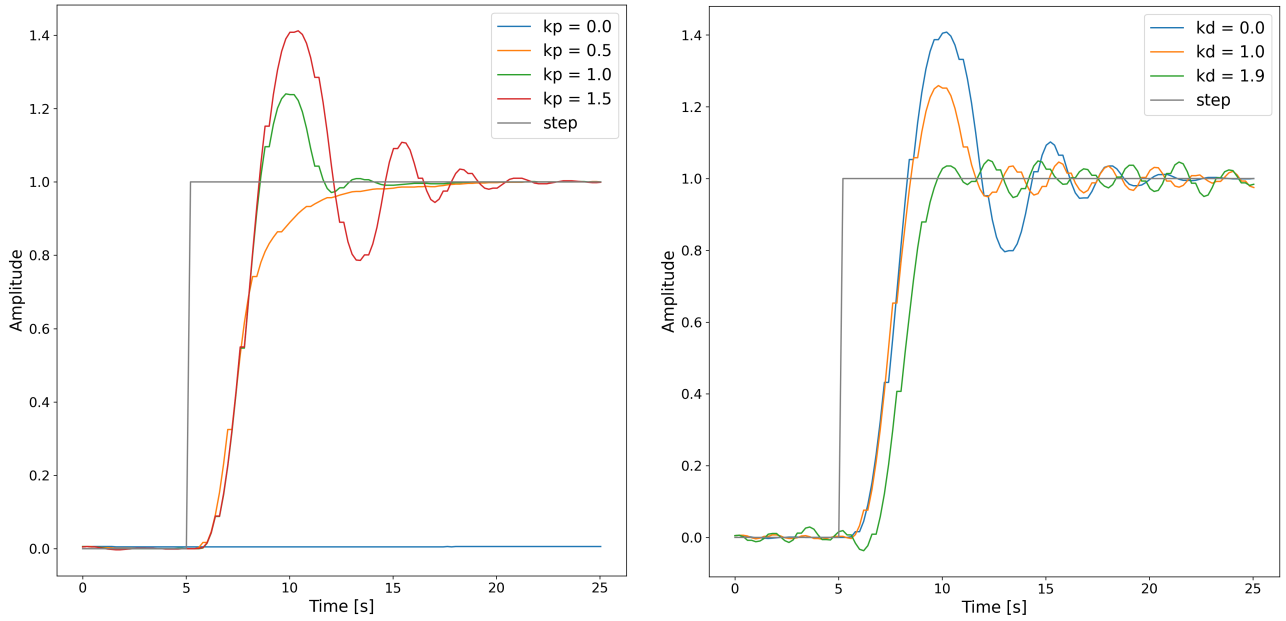


Figure 4. Two important metrics for the step response.

After (roughly) defining our step response (i.e., low overshoot but also short rise time), we proceeded by varying the parameters to observe how they influence the step response. We started with parameter K_p and then set K_d (Equation (4)) to its minimum (the look-ahead distance was not used in the step response since we were directly changing the reference signal). In Figure 5a, we show the step response for several values of parameter K_p. In this figure, we can observe how parameter K_p influences the step response. We started by

setting K_p to zero. Obviously, in this case, the UAV will not change its altitude and will just remain flying according to its original reference signal. We then increase the value of K_p and K_d by steps of 0.1 (we do not show all the possibilities to keep the figure clear). When we raise the value of K_p, we can see the rise time decreasing. At a value of 0.5, the UAV reaches the new altitude without any overshoot. However, the rise time is still too long. If we increase the value of K_p, we can reduce the rise time further. In Figure 5a, it seems that in terms of rise time, there is no difference between $K_p = 1.0$ and $K_p = 1.5$. However, it is always important to take both K_p and K_d into account. As stated above, we changed both parameters by a step size of 0.1 (up to 2.0), and after 400 experiments, our results showed that with $K_p = 1.5$ and $K_d = 1.9$, the best results could be achieved. In Figure 5b, we show the influence of the parameter, K_d. As expected based on the control theory, if we increase the value of the K_d parameter, the overshoot will be reduced. However, a value that is too high will result in an unstable system that would tend to oscillate. As we can see, the overshoot decreases with a higher value of K_d. Increasing the value of K_d to beyond 1.9 (best choice) only introduces more unwanted oscillation. As shown in Figure 6, for the response of the system with the optimal parameters (w.r.t. overshoot and mission time), some oscillation remains for our control approach. However, since the terrain level is constantly changing in our application, this oscillation will not prevail. In addition, the influence of the look-ahead distance can only be clearly observed in the context of an actual mission (longer time periods). Therefore, we will show its influence in the next section.



(**a**) (**b**)

Figure 5. Analysis of the influence of parameters K_p and K_d on the step response of the UAV. (**a**) Influence of parameter K_p ($K_d = 0$). (**b**) Influence of parameter K_d ($K_p = 1.5$).

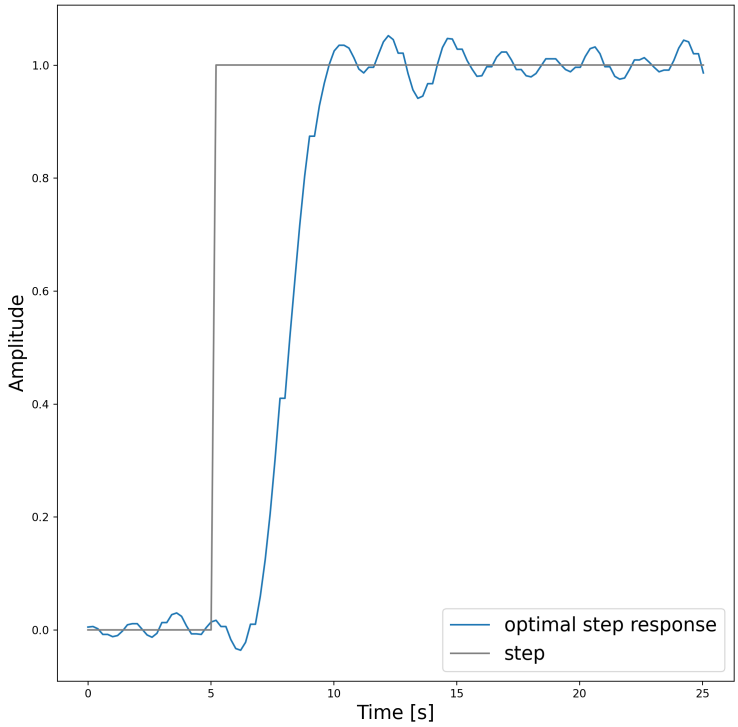


Figure 6. Step response with optimal parameters, i.e., $K_p = 1.5; K_d = 1.9$.

4. Evaluation

After modeling our proposed solution and obtaining the optimal parameter values, we tested our approach under different conditions. In order to experiment safely, we relied on our realistic UAV simulator/emulator called ArduSim [17]. We experimented with two different scenarios: a rural area with some hills (Cumbre de Calicanto, Valencia, Spain) and a mountainous region (Mulhacén, Granada, Spain). Each scenario was simulated several times, with different values for the look-ahead parameter. During our experiments, we measured the terrain altitude and the above ground level altitude of the UAV. We start by introducing the ArduSim simulator, after which we will go over the two scenarios.

4.1. ArduSim

ArduSim is an open-source multi-UAV flight simulator/emulator. The code is available online [18] under the Apache License 2.0. The simulator is a large Java project, and it was formally presented in [17]. Below, we briefly highlight the key characteristics of ArduSim:

- **Simulator and GCS:** ArduSim can be used in two operation modes. It can function as a simulator and as a Ground Control Station (GCS) for real UAVs. The protocol (in this case, the model we created) is uncoupled from the operational mode. This ensures quick deployment on real UAVs since the protocol does not need any changes once it works in the simulated environment.
- **Automated testing:** As a simulator, ArduSim can be executed with or without a graphical user interface (GUI). The mode with a GUI is typically used when developing the protocol. Once the protocol is developed and a lot of experiments need to be performed, the mode without a GUI can be used, and tests can be run automatically.
- **UAV-to-UAV communication:** ArduSim uses the 802.11a standard to communicate, both between UAVs themselves and between the UAVs and the ground station. When ArduSim is used as a simulator, communication is accomplished using virtual links. Whenever protocols are thoroughly tested, they can be deployed on real UAVs. In this case, ArduSim will send the messages via User Datagram Protocol (UDP) broadcasts.

- **Scalability:** ArduSim was designed to be a multi-UAV flight simulator. Therefore, a lot of effort was put into scalability. This resulted in a simulator that is able to run up to 100 UAVs in near-real time, and up to 256 UAVs in soft-real time on a high-end PC (Intel Core i7-7700, 32 GB RAM).
- **API:** Inside ArduSim's codebase, an Application Programming Interface (API), developers offer many common functions (taking off, moving to a GPS location, landing, etc.) to control the UAV.
- **Data logging:** ArduSim extensively logs data in various formats after a flight in order to make it development- and debug-friendly.

Our ArduSim tool acts as both a real-time simulator and as an onboard software element. When deployed on real UAVs, it is expected to run on an embedded system deployed on the UAV itself (e.g., Raspberry Pi), which communicates with the flight controller using a serial connection. The communications protocol used for such connection is MAVLink [19], a lightweight messaging protocol for communicating with drones that follows a modern hybrid publish-subscribe and point-to-point design pattern. This way, we offer a universal software that can be adopted by other users without requiring any changes to the Ardupilot firmware itself.

4.2. Scenario A: Rural Area

Our first scenario (Cumbres Calicanto) is located in a rural area close to Valencia, Spain (see Figure 7a). We chose this region for its moderate set-off (see Figure 7b). The top of the hill is around 210 m high; one side the slope of the hill is gentle, in the center there are some abrupt peaks and troughs, and on the other side of the hill, the slope is much steeper.

(**a**) (**b**)

Figure 7. Scenario A: Rural area. (**a**) Localization of Cumbres Calicanto. (**b**) Bird's eye view of Cumbres Calicanto.

As mentioned before, the influence of the look-ahead distance can only be measured during a complete mission. In general, looking further ahead will provide the UAV more time to anticipate altitude changes. However, looking too far ahead will introduce unwanted behavior if the terrain has narrow peaks or troughs. During our experiments, we varied the value of the look-ahead distance in the range of 2–20 m. The DEM file that we used had a resolution of 2 m (square of 2 × 2 m). Therefore, we increased the look-ahead distance at two-meter steps each time. During the mission, we measured the error every 200 ms. After the experiment, we calculated the mean error and the standard deviation (see Table 1). Based on those metrics, we could determine that a look-ahead distance of 8 m in this scenario is the most effective one at reducing the error.

The results of the entire mission are depicted in Figure 8, showing that the UAV adapts to the terrain profile correctly. As shown in Figure 8b, the error is normally distributed,

with $\mu = -0.0493$ and $\sigma = 1.6260$. Stated otherwise, in 99.7% of the time, the error was less than 5 m.

Table 1. Influence of the look-ahead distance for scenario A.

Look-Ahead Distance (m)	Mean Error (m)	Standard Deviation (m)
2	0.1834	2.5176
4	−0.1464	2.3930
6	−0.0721	1.7832
8	−0.0493	1.6260
10	−0.0724	1.6399
12	0.0203	1.5128
14	0.0049	1.7969
16	0.0449	1.9540
18	0.1118	2.1984
20	0.1678	2.5435

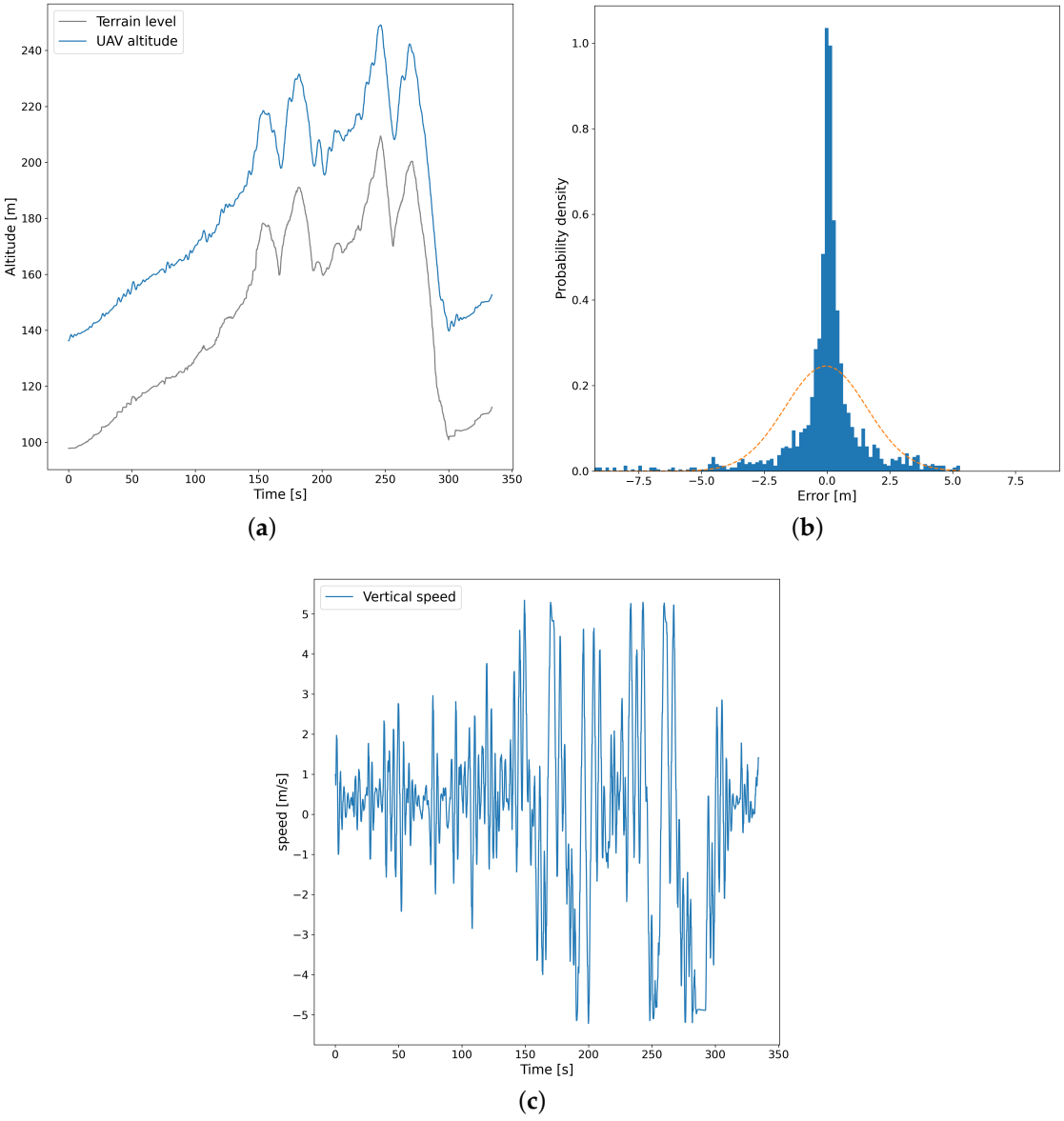


Figure 8. Results for scenario A. (**a**) Altitude of the UAV w.r.t. terrain altitude. (**b**) Error distribution. (**c**) Vertical speed (m/s) of the UAV.

During our experiment, we also tracked various metrics. One of them was the vertical speed of the UAV, which we show in Figure 8c. It is interesting to observe how the vertical speed of the UAV changes w.r.t. the terrain altitude (shown in Figure 8a). As we can see, at the beginning (before 150 s), the terrain level changes slowly, and the vertical speed of the UAV is bounded between -3 m/s and 3 m/s. From 150 s onwards, the terrain level changes much faster. This is also represented in the vertical speed of the UAV, which often reaches its maximum of ± 5 m/s. In addition, we find that changing the altitude levels does not have a significant impact in terms of extra time or energy. To check this, we measured various metrics for the flight using our altitude adjustment approach and without the use of this mechanism. As shown in Table 2, the mission (depicted in Figure 7b) is 3238 m long. Due to the set-off, the UAV had to adjust its overall altitude by 583 m. This change in altitude caused a slight mission delay of about 3% and an increase in energy usage of 3.49%. These payoffs are very small, especially considering that the alternative (not changing the altitude) will result in a crash.

Table 2. Rural flight: influence of adjusting the altitude on flight time and energy.

	Without Adjustment	With Adjustment	Difference
Horizontal distance (m)	3238	3238	0%
Vertical distance (m)	0	583	-
Flight time (s)	324	334	+3.01%
Energy consumed (kWh)	127	132	+3.49%

4.3. Scenario B: Mountain Area

After the promising results in a rural area, we tested our model in a more challenging environment, i.e., a mountainous region. We chose to perform this simulation in the area of Mulhacén, Granada, Spain (see Figure 9a). In these mountains, the ground levels change faster and more often, which makes it more difficult for the UAV to maintain a constant altitude above the ground.

(**a**) (**b**)

Figure 9. Scenario B: Mountain area. (**a**) Localization of Mulhacén. (**b**) Bird's eye view of Mulhacén.

For this region, we also experimented with the look-ahead distance. As shown in Table 3, the margin of error is higher due to the increased scenario complexity. In a mountainous region, the look-ahead distance has to be shorter, as expected. In this case, we can obtain the best results with a look-ahead distance of 4 m.

The results of this scenario are shown in Figure 9a. From Figure 10a, it can be observed that the terrain is more challenging, and with the use of our approach, a ground crash can be avoided. However, as also shown in Figure 10b, the distribution of the error is broader than in the previous case. In addition, we tracked various metrics for this flight. As shown

in Figure 10c, the vertical speed of the UAV was often at its maximum in an attempt to maintain a constant altitude in this rapidly changing terrain. Taking this into account, we might have achieved better results (i.e., smaller mean error and standard deviation) if we did not restrict the maximum vertical speed to 5 m/s. However, we did not increase the maximum speed for various reasons. First and foremost, for the obvious safety reasons. We must remember that there is a 200 ms delay before we can update the speed, and then there is an extra delay (due to inertia) before the UAV actually reaches this new speed. If we increased the maximum vertical speed, this would most likely lead to an unwanted oscillating behavior. Furthermore, we want our approach to be applicable to many types of drones, and not all drones have enough power to ascend faster than 5 m/s. Finally, increasing the maximum vertical speed would also lead to higher energy consumption.

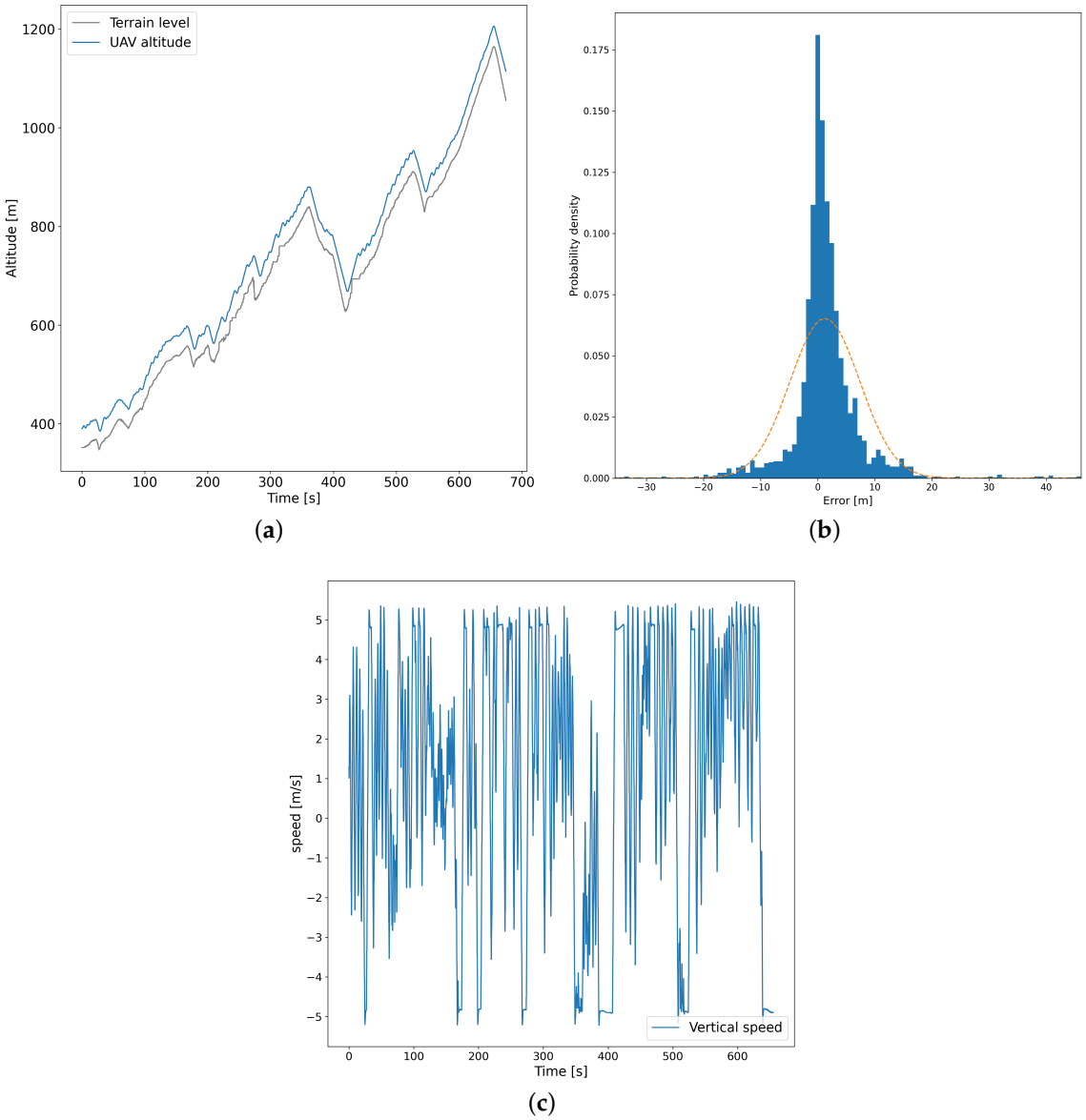


Figure 10. Results of scenario B. (**a**) Altitude of the UAV w.r.t. terrain altitude. (**b**) Error distribution. (**c**) Vertical speed of the UAV throughout time.

Table 3. Influence of the look-ahead distance for scenario B.

Look Ahead Distance (m)	Mean Error (m)	Standard Deviation (m)
0	−1.5206	8.6427
2	−0.1372	7.4118
4	1.2548	6.1141
6	2.9995	6.5989
8	4.4752	7.7420
10	5.9446	9.3839

As expected, the influence of adjusting the altitude on flight time and energy consumption is much higher in this demanding scenario compared to the previous (rural) scenario. As shown in Table 4, the mission depicted in Figure 9b has a horizontal distance of 6193 m. The main difference between this scenario and the previous one is that in the current scenario, the vertical distance traveled is much higher. In order to regulate the altitude above the ground, the UAV had to ascend and descend for a total of 2100 m throughout the mission. This caused the flight time to increase by 29%, and consequently, the energy consumption rose by 30% as well.

Table 4. Mountain flight: influence of adjusting the altitude on flight time and energy.

	Without Adjustment	With Adjustment	Difference
Horizontal distance (m)	6193	6193	0%
Vertical distance (m)	0	2100	-
Flight time (s)	507	655	+28.99%
Energy consumed (kWh)	200	261	+30.59%

5. Conclusions and Future Work

Research in the UAV field has matured over the last years; as a result, many industries now use UAVs in their day-to-day activities. Most of the time, a UAV is still manually controlled by a pilot. Nevertheless, more and more (semi) autonomous applications are being developed. For an autonomous application, it is important that a UAV is able to maintain a constant altitude relative to the ground, even when the terrain altitude varies. This would prevent the UAV from crashing into the ground or from exceeding legal altitude constraints.

In this work, we provided a solution that attempts to maintain a constant altitude relative to the ground. Our approach is based on a PD controller, which uses the altitude data coming from a DEM file as the reference signal. Since drones differ in size, weight, and many other factors, the parameter values of the PD controller will likely differ for each UAV. Therefore, we explained our tuning method in detail so that it could be easily replicated.

Once the PD controller was tuned, we experimented with two different scenarios: a rural area, and a mountainous area. The results of the first (rural) scenario show that the UAV is able to maintain a constant relative altitude above the ground. During that experiment, the error (i.e., the difference between the desired altitude and the actual altitude) was smaller than 5 m 99.7% of the time, and the increase in flight time and energy consumed due to such adjustments was 3.01% and 3.49%, respectively—a performance that we consider sufficient for most applications. In the more challenging scenario, i.e., the mountainous area, the error increased slightly, and the increase in flight time and energy consumed due to the terrain profile adjustments (2100 m in total for vertical mobility) were more significant, reaching an increase of 28.99% and 30.59%, respectively.

Although the results are quite good, more research is necessary to reduce the error in challenging environments. To that end, in the future, we would like to extend our research by investigating automatic tuning methods. Furthermore, to avoid rapid changes in the flight altitude, we will consider smoothing the altitudes between adjacent cells in the DEM.

Finally, we would like to investigate a slightly different approach, where we change the programmed flight altitude and allow the flight controller to adjust the altitude.

Author Contributions: Conceptualization, C.T.C.; funding acquisition, C.T.C.; investigation, J.W. and C.M.; methodology, J.W.; software, C.M.; supervision, C.T.C., E.H.-O., J.-C.C. and P.M.; writing—original draft, J.W.; writing—review and editing, C.T.C., E.H.-O., J.-C.C. and P.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work is derived from R&D projects RTI2018-096384-B-I00 and RTC2019-007159-5, funded by MCIN/AEI/10.13039/501100011033 and the European Regional Development Fund (ERDF) "A Way of Making Europe", and by the Conselleria de Educación, Investigación, Cultura y Deporte, Direcció General de Ciéncia i Investigació, Proyectos AICO/2020, Spain, under the Grant AICO/2020/302.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

DEM	Digital elevation model
UAVs	Unmanned Arial Vehicles
AGL	Above ground level
GCS	Ground Control Station
API	Application Programming Interface
GUI	Graphical user interface

References

1. Giones, F.; Brem, A. From toys to tools: The co-evolution of technological and entrepreneurial developments in the drone industry. *Bus. Horizons* **2017**, *60*, 875–884. [CrossRef]
2. Pixhawk. Pixhawk: The Hardware Standard for Open-Source Autopilots. 2021. Available online: https://pixhawk.org/ (accessed on 26 October 2021).
3. Team, A.D. ArduPilot: Copter Documentation. 2021. Available online: https://ardupilot.org/copter/ (accessed on 26 October 2021).
4. EASA. Easy Access Rules for Unmanned Aircraft Systems. 2021. Available online: https://www.easa.europa.eu/document-library/easy-access-rules/online-publications/easy-access-rules-unmanned-aircraft-systems (accessed on 26 October 2021).
5. Ltd, V.T. Litchi. 2022. Available online: https://flylitchi.com/ (accessed on 9 February 2022).
6. Sze, L.T.; Cheaw, W.G.; Ahmad, Z.A.; Ling, C.A.; Chet, K.V.; Lateh, H.; Bayuaji, L. High resolution DEM generation using small drone for interferometry SAR. In Proceedings of the 2015 International Conference on Space Science and Communication (IconSpace), Langkawi, Malaysia, 10–12 August 2015; pp. 366–369. [CrossRef]
7. Mogili, U.R.; Deepak, B.B.V.L. Review on Application of Drone Systems in Precision Agriculture. *Procedia Comput. Sci.* **2018**, *133*, 502–509. [CrossRef]
8. Aurambout, J.P.; Gkoumas, K.; Ciuffo, B. Last mile delivery by drones: An estimation of viable market potential and access to citizens across European cities. *Eur. Transp. Res. Rev.* **2019**, *11*, 1–21. [CrossRef]
9. Amukele, T.; Ness, P.M.; Tobian, A.A.; Boyd, J.; Street, J. Drone transportation of blood products. *Transfusion* **2017**, *57*, 582–588. [CrossRef] [PubMed]
10. Swihart, D.E.; Barfield, A.F.; Griffin, E.M.; Lehmann, R.C.; Whitcomb, S.C.; Flynn, B.; Skoog, M.A.; Processor, K.E. Automatic Ground Collision Avoidance System design, integration, flight test. *IEEE Aerosp. Electron. Syst. Mag.* **2011**, *26*, 4–11. [CrossRef]
11. Griffin, E.M.; Turner, R.M.; Whitcomb, S.C.; Swihart, D.E.; Bier, J.M.; Hobbs, K.L.; Burns, A.C. Automatic Ground Collision Avoidance System Design for Pre-Block 40 F-16 Configurations. In Proceedings of the 2012 International Symposium on Aerospace Technology, Jeju, Korea, 13–15 November 2012; pp. 4–15.
12. Eller, B.; Stanfill, P.; Turner, R.; Whitcomb, S.; Swihart, D.; Burns, A.; Hobbs, K. Test and Evaluation of a Modified F-16 Analog Flight Control Computer. In *AIAA Infotech@Aerospace (I@A) Conference*; American Institute of Aeronautics and Astronautics: Boston, MA, USA, 2013; p. 4726. [CrossRef]

13. Muliadi, J.; Kusumoputro, B. Neural network control system of UAV altitude dynamics and its comparison with the PID control system. *J. Adv. Transp.* **2018**, *2018*, 3823201. [CrossRef]
14. Kharchenko, V.; Kuzmenko, N. Unmanned aerial vehicle collision avoidance using digital elevation model. *Proc. Natl. Aviat. Univ.* **2013**, *1*, 21–25. [CrossRef]
15. de Información Geográfica, O.A.C.N. Centro de Descargas. 2021. Available online: http://centrodedescargas.cnig.es/CentroDescargas/index.jsp (accessed on 26 October 2021)
16. CDEMA. Digital Elevation Models. 2021. Available online: https://www.cdema.org/virtuallibrary/index.php/charim-hbook/data-management-book/3-base-data-collection/3-2-digital-elevation-models (accessed on 26 October 2021).
17. Fabra, F.; Calafate, C.T.; Cano, J.C.; Manzoni, P. ArduSim: Accurate and real-time multicopter simulation. *Simul. Model. Pract. Theory* **2018**, *87*, 170–190. [CrossRef]
18. GRC. ArduSim: A Novel Real-Time Flight Simulator. 2021. Available online: https://github.com/GRCDEV/ArduSim (accessed on 26 October 2021).
19. Meier, L.; QGroundControl. MAVLink Micro Air Vehicle Communication Protocol. 2007. Available online: https://mavlink.io/en/ (accessed on 11 May 2019).