drones

MDPI

Article

Precision Landing for Low-Maintenance Remote Operations with UAVs

Miguel Moreira [1,*], Fábio Azevedo [1,2], André Ferreira [1], Dário Pedro [3,4,5], João Matos-Carvalho [6], Álvaro Ramos [3], Rui Loureiro [1] and Luís Campos [3]

1 Beyond Vision, 3830-352 Ílhavo, Portugal; fabio.azevedo@beyond-vision.pt (F.A.); andre.ferreira@beyond-vision.pt (A.F.); rui.mendes@beyond-vision.pt (R.L.);
2 Electrical and Computing Engineering Department, FEUP, University of Porto, 4200-465 Porto, Portugal
3 PDMFC, 1300-609 Lisbon, Portugal; dario.pedro@pdmfc.com (D.P.); alvaro.ramos@pdmfc.com (Á.R.); luis.campos@pdmfc.com (L.C.)
4 Centre of Technology and Systems, UNINOVA, 2829-516 Caparica, Portugal
5 Electrical Engineering Department, FCT, NOVA University of Lisbon, 2829-516 Caparica, Portugal
6 Cognitive and People-Centric Computing Labs (COPELABS), Universidade Lusófona de Humanidades e Tecnologias, Campo Grande 376, 1749-024 Lisboa, Portugal; joao.matos.carvalho@ulusofona.pt
* Correspondence: miguel.moreira@beyond-vision.pt

Abstract: This work proposes a fully integrated ecosystem composed of three main components with a complex goal: to implement an autonomous system with a UAV requiring little to no maintenance and capable of flying autonomously. For this goal, was developed an autonomous UAV, an online platform capable of its management and a landing platform to enclose and charge the UAV after flights. Furthermore, a precision landing algorithm ensures no need for human intervention for long-term operations.

Keywords: landing; computer vision; pattern recognition; UAV; RTK; control; communication; remote control; autonomous navigation

Citation: Moreira, M.; Azevedo, F.; Ferreira, A.; Pedro D.; Matos-Carvalho, J.; Ramos, A.; Loureiro, R.; Campos, L. Precision Landing for Low-Maintenance Remote Operations with UAVs. *Drones* **2021**, *5*, 103. https://doi.org/10.3390/drones5040103

Academic Editors: Andrey V. Savkin and Kooktae Lee

Received: 31 July 2021
Accepted: 8 September 2021
Published: 24 September 2021

1. Introduction

Multirotor Unmanned Aerial Vehicles (UAVs) are a fundamental tool in a multitude of applications [1–4]. The applicability of UAVs in a diverse range of fields is partly due to their ability for hovering, their high maneuverability, and a reasonable payload–size ratio for carrying task-specific sensors or actuators. A significant part of the current research related to UAVs is focused on higher-level tasks, such as navigation or task planning [5,6]. In addition, the high degree of autonomy and reliability of UAVs allows for exploring remote operations with minimum intervention from the operator, which has led to a recent paradigm shift in applications such as site surveying.

Remote operations become more efficient if they can be conducted for longer periods, since UAVs' deployment and assembling overhead becomes less significant. However, operating UAVs remotely for extended periods requires additional capabilities for most typical applications. Procedures like charging (or swapping) batteries or ensuring safe UAV storage must now be achieved without human intervention. Additionally, it is advantageous if the system is fitted with a communications module capable of reaching a central control station [7]. Agriculture [1,8], first-aid response [9–11], surveying and photogrammetry [12,13] or strategic positioning for unscheduled monitoring [14–17] are examples of applications that can benefit from autonomous remote operations. In extensive agriculture fields, one or more landing bases can be placed in optimal locations allowing periodic surveying for monitoring the crops' growth and health [18]. Similarly, first-response entities might decide to deploy UAVs at strategic locations and request their activation when needed by a central control station upon an emergency. The closest (or any

other relevant selection criteria) UAV to the occurrence can be activated and can live-feed useful information in preparation for human assistance [19–22].

The key aspect that will permit the widespread deployment of UAVs and enable their use in almost all sectors of the economy is autonomy. Autonomy is key to achieving low-cost extended remote operations and it is supported by advanced computer vision algorithms that allow very precise landing onto a charging base station (even if its location is unknown to the UAV), a very reliable and fault-tolerant communication system, and an online platform allowing for real-time situation awareness and autonomous decision making.

1.1. UAV Monitoring Platforms

A well-known open-source tool used for manual flight control and autonomous flight planning is called QGroundControl [23]. This provides configuration and support for UAVs that run PX4 [24] or Ardupilot [25] firmware. Its user interface uses a satellite map to display the currently connected UAVs and allows the user to plan a mission by adding waypoints on the map. Aside from these functionalities, this tool offers few advantages since it cannot store data continuously in a database, rather, it can only provide real-time data [26].

Another platform, FlytOS [27], which is a holistic cloud platform to connect UAVs, offers a wide variety of features. For example, this platform provides plugins for autonomous planning, collision avoidance, and thermal camera integration. However, FlyOs is proprietary software that requires the user to implement and start using this custom-made solution. Moreover, this platform lacks an important feature, namely, allowing the user to obtain insights by reviewing previous flights and analyzing them in terms of their success or failure [26].

It can be stated that although there are currently many platforms available for UAV monitoring, a robust and reliable platform implementing a rich set of high-level abstractions, with an open Application Programming Interface (API) that can be extended based on specific application needs, is still unavailable [28–31].

1.2. Algorithms for UAV Precision Landing

Multirotor UAVs are extremely flexible and are used in a variety of operations such as agriculture [32], search and rescue, or inspection [33,34]. However, this flexibility usually comes with a significant limitation: limited range and flight time due to the low battery storage capacity. In scenarios where a repeatable operation is required during periods longer (typically much longer) than the current maximum flight time, a solution that supports the charging or replacement of batteries is needed [35]. Existing landing base solutions are typically available only for small UAVs [36–41].

Different authors have approached the relative position problem (UAV and landing pad) differently, and they have proposed different solutions [42]. In general, researchers assume that Global Navigation Satellite System (GNSS) receivers do not provide an acceptable position error for the application scenarios [43–46]. Hence, the most common technique adopted to solve the relative position problem uses computer vision to compute the position of the multirotor relative to the landing pad.

Usually, when a computer vision approach is used, it is assumed that there is prior knowledge of the pattern structure. In order to detect the desired landing location, visual markers are usually used. One of the most common is the classic "H" which describes a heliport [47–49]. Furthermore, other types of patterns are used for specific cases, such as various circumferences, which may be either concentric [50] or not [51]. When using patterns with circumferences, it is common to utilize ellipse fitting techniques [52] for pattern matching. In order to obtain a heading reference for the pattern, it needs to be asymmetric. The pattern can be asymmetric in its shape or other features, such as color [53–55].

Many fields of mobile robotics use active markers, such as patterns with visible spectrum Light Emitting Diode (LED) [56–58], or other ranges of the light spectrum (e.g.,

infrared) [59–61]. For example, D. Kim et al. [62] uses an active marker on the location of multiple ground robots. The light from the marker is modeled in a way that it is possible to detect the pattern and exclude false positives.

Black and white patterns such as Apriltags [63] and Quick Response (QR) codes [64] are also used as markers. Mengyin Fu et al. [64] uses a mixed marker with a QR code inside of a circle. This way, it is possible to detect the shape of the circle when the vehicle is further away and then the QR code when it is closer, thus allowing to have a pose with a lower error. A similar approach was used by Grobler [65], where the author tests which shape is faster to detect, combined with an Aruco marker.

The usage of fiducial markers can be improved using high contrast materials, as exposed by Aalerud [66], such as reflectors combined with matte materials. Aruco and Apriltags are low-resolution fiducial markers, allowing usage at higher distances; however, their detection algorithms have different computational needs and provide position solutions with different levels of error associated [67].

1.3. Proposed Work

At the core of our complete solution for low-cost extended autonomous remote operations is a self-powered charging landing base responsible for safely enclosing a medium-size UAV and charging its batteries. In addition, the charging landing base contains a communication module used to communicate with both the UAV and a remote control station. Furthermore, it is constructed in a way that is visually identifiable by a flying UAV. Using Real-Time Kinematics (RTK) positioning, the static nature of this charging landing base allows it to be used as a GNSS RTK base, constantly communicating with the UAV (the RTK system rover). To increase the energy efficiency of the system, solar panels charge the landing base. These solar panels are also used to charge the UAV's batteries.

In this document, we focus on UAV computer vision and UAV-landing base communications required for precision landing. Assuming that the GNSS positioning might not be reliable for proper landing, a computer vision algorithm has been developed and implemented. This algorithm ensures that the UAV goes to the desired landing location and aligns its heading. The latter is essential to ensure a proper landing and, most importantly, to be able to charge the batteries without the need for complex alignment mechanisms on the charging landing base. Furthermore, the communications capability in the landing base allows for remote control and monitoring of the UAV and base, even without a direct link between the control center and the UAV.

The remaining manuscript is organized as follows. First, Section 2 presents an overview of the system proposed in this document, from monitoring the UAV via a cloud platform to the landing maneuver onto the charging landing station using computer vision. The application environments are presented in Section 3, with the results presented in Section 4. Section 5 describes and discusses the experimental results. Conclusions and future work to improve our system are discussed in Sections 6 and 7, respectively.

2. Materials and Methods

Three fully independent systems were designed, implemented, and integrated to have an ecosystem capable of operating autonomously without the need for human intervention. The Hexa Exterior Intelligent Flying Unit (HEIFU) UAV, the beXStream fleet management platform, and the ALPHA Landing Base and Charging Station. These components are orchestrated by the platform backend, which coordinates both the landing base (and charging station) and the UAV.

In the following subsections, these three systems are further explained. We start by presenting the overall architecture of the solution, followed by detailed descriptions of each system and how they were integrated to satisfy the requirements of a low-cost solution for extended autonomous remote operations. The precision landing algorithm (with an error below five centimeters) is vital for the whole ecosystem. It ensures that the UAV lands in a position suitable for being charged by the charging station.

2.1. Architecture

This document proposes the architecture depicted in Figure 1. The beXStream platform is the mastermind of the solution, controlling all assets (UAVs and landing bases are covered in this document but can in the future be used to remotely manage any asset, for example, individual sensors and actuators) via different communication channels. Furthermore, it integrates different frontend applications that enable users to monitor and execute actions with ease.

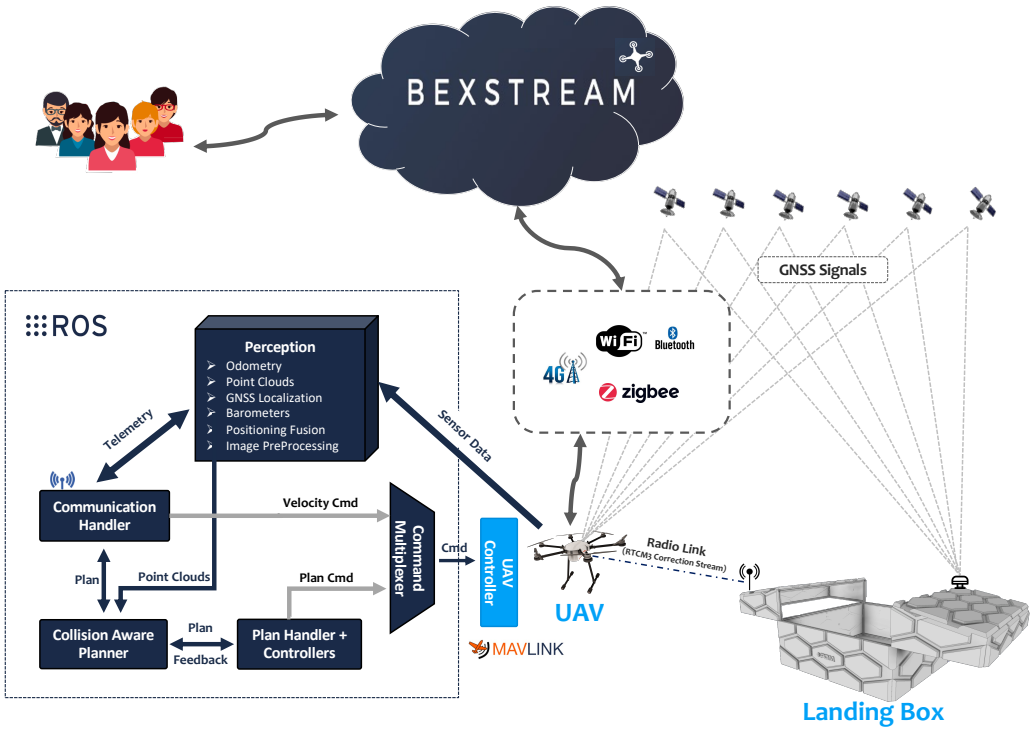


Figure 1. Proposed architecture for remote autonomous operations.

The HEIFU code architecture was developed in a Robot Operating System (ROS), which allows greater flexibility and easier integration of future additional features. Moreover, ROS provides several modules that can be adjusted or reused to fit new purposes. In this section, we use terms intrinsic to ROS in order to explain each component represented in Figure 1. The control for the ALPHA landing base was developed in Node.js since it only requires a primary interface to the platform.

2.2. beXStream platform

The beXStream platform can manage different assets, integrate external APIs and databases, and be easily integrated by third-party developers. A high-level block component architecture of the platform is illustrated in Figure 2. In this figure, it can be observed that the backend is responsible for managing all other modules. It orchestrates transactions between different components that are dockerized and instantiated whenever needed. Depending on the data type, different databases are integrated into the beXStream, such as influxDB [68], elastic search, and MySQL. Diverse frontend applications can be used to access the APIs exposed by the backend. For example, the web frontend can be accessed via the link [69]. To make the whole system scalable, Kubernetes technology was used, allowing maximum utility from containers and build cloud-native applications that can run anywhere, independently of cloud-specific requirements.

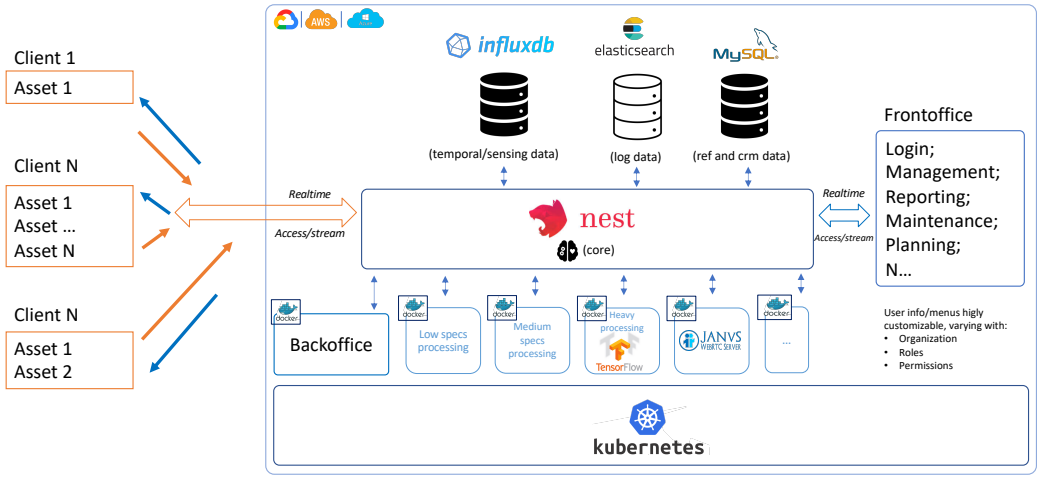


Figure 2. The beXStream platform architecture overview.

For safety issues, it is essential to have the capability to visualize, in real-time, the UAV's video stream. To address this issue, a media gateway based on Janus [70] was integrated on the platform, alongside the modules required to address network connections, which will be addressed on the following sub-modules.

2.2.1. Backend

The backend is the core module of the platform, where all the management and security measures are developed. The backend controls the access of the several users that log in via different frontends, as well as the assets. This module controls permission levels and restricts users' access to their organization, so they can only control/view assets that belong to them. Meaning that only users with the proper authorization and organization can alter the asset configurations, send commands or view its stream. Security measures such as multiple logins with the same account are blocked for users and assets. The backend interacts with every module of the system. It creates the configurations needed for the video stream that is then sent to the media gateway, depending on the camera's specifications of the UAV.

2.2.2. Media Gateway

For the management of the video stream, a media gateway was used, in this case Janus-Gateway [70,71]. This gateway allows to receive a stream from an asset and broadcast it to several clients, enabling them to observe the assets' point of view. This module also allows the management of several streams from different assets and switching perspectives between them. The video stream quality is very dependent on the connection quality, whether between the asset and platform or between the gateway and the client. To reduce this sensitivity, the media gateway can implement a buffer for incoming packets that are out of order, thereby increasing the stream's reliability. In order to increase the compatibility, transcoding of the video is done in the media gateway [72]. However, the transcoding is not performed on most assets due to the heavy processing power needed in the process, as it is proportional to the video resolution and framerate.

To start the transmission of the assets' stream, it is necessary to register the asset in the platform through a frontend. Each asset must be registered with a unique name and a password. Once the registration is finalized, an encrypted configuration file is downloaded to the client. This file needs to be inserted in the On-Board Computer (OBC) of the asset. The file contains the access information for that specific asset, needed to communicate with the system, encrypted using AES-128. This file must be in the home directory of the asset. When an asset is turned on, it decrypts the configuration file and sends the login information to the backend module, which replies with a token for this session. For example, if the asset is equipped with a camera, then a session for the stream is created in

the media gateway. The backend will generate a configuration depending on the camera's characteristics and send the information to the media gateway, which will give feedback once the session is created. After the session setup is completed, the media gateway will inform the backend of what port it is expecting the stream on, informing the UAV that it can begin streaming towards that port. A flow chart of the process between a UAV (example asset) and a client can be visualized in Figure 3. This stream is transmitted using Real-time Transport Protocol (RTP) or Real-time Streaming Protocol (RTSP). Session Traversal of UDP Through NAT (STUN) and Traversal Using Relays around NAT (TURN) may be used to resolve connections problems. The communication between the client and the media gateway is established by WebSocket Secure (WSS). At the same time, the connection between the backend and the media gateway is made by Representational state transfer (REST).

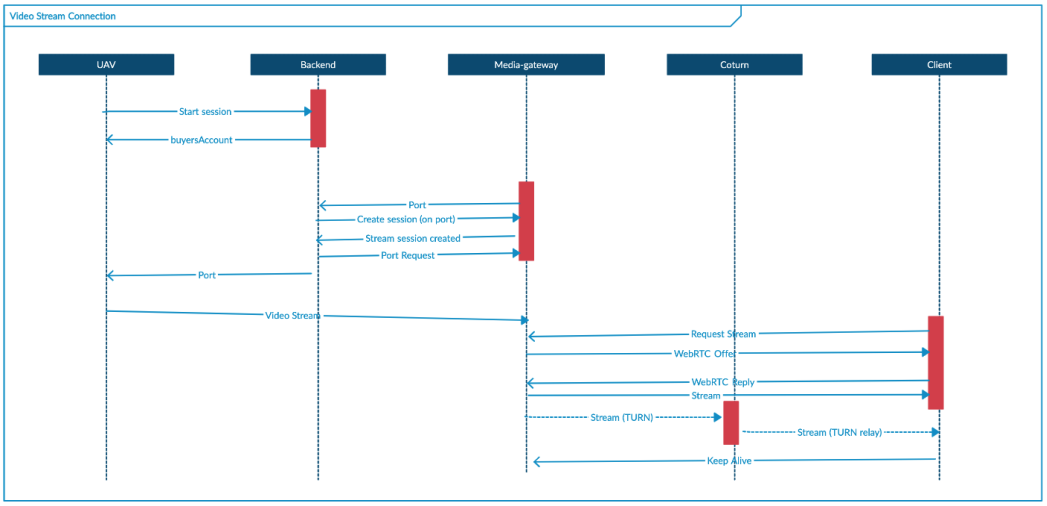


Figure 3. Message exchanges to establish a video stream connection between the UAV and the platform. The dashed lines represent the possible flow the RTP/RTSP streams when using STUN or TURN.

Communication performance is an essential factor for the remote applications the authors are trying to address. The article [5] details the communication pipelines used, which allows telemetry data for the UAV to tolerate a delay and fit the Ultra Reliable Low Latency Communications (URLLC) requirements [73]. The tests show that in the majority of the European countries is possible to achieve a Round Trip Time (RTT) of approximately 100 ms. Furthermore, if lower RTT is required, a new national instance of the presented backend and media gateway can be deployed, which shortens the physical traveling distances of the packages between the UAV to the backend and the backend frontend application. From the multiple deployment experiences, the average RTT is not as critical as the jitter and the network congestion are, which becomes more probable with the increase of the physical distance and the different network hopping.

2.2.3. Frontend

The frontend was developed in Angular, exposing a website for the registration of UAVs, managing users/organizations, and visualizing streams. It allows visualizing details of the UAV and the landing platform, allowing the user to design different types of missions and push them to the system. The mission planning screen can be visualized in Figure 4. Using this page, a user can quickly draw the region of interest and remotely push the mission to the UAV. Furthermore, on the website, a video player is used with baseline characteristics to comply with all browsers. The stream is received over Web Real-Time Communication (WebRTC), which uses a peer-to-peer connection. In most scenarios, the clients will be behind a Network Address Translation (NAT), which means the platform

cannot address the client directly. In order to circumvent the NAT rules, a STUN server may be used, which serves to inform the corresponding peer of what their public IP is. By sharing the STUN responses, the peers can communicate even when behind NATs. However, if the client is behind a symmetric or carrier-grade NAT, then a TURN server must be used to establish the connection. In order to ensure connectivity, a TURN server is deployed in the system, based on a Coturn server [74].

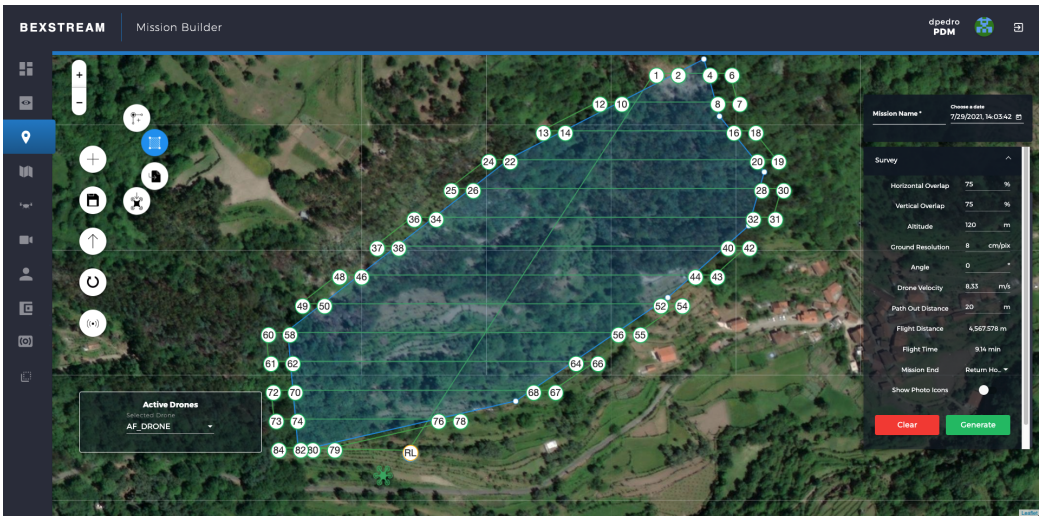


Figure 4. Frontend module of the beXStream platform to generate survey mission and transmiting it to a UAV.

2.3. HEIFU UAV

The HEIFU (Figure 5) is the second component of the ecosystem. It is a hexacopter designed with maximum flexibility to support any operation (e.g., precision farming, search, and rescue). The UAV is equipped with an OBC running Ubuntu and ROS, being an open platform that allows the integration of different inputs and is used to run multiple tasks simultaneously, such as image processing or data relaying. Equipped with a depth camera or a Light Detection And Ranging (LiDAR), HEIFU is capable of ambient perception to do collision avoidance and path planning. In addition, HEIFU can be used with different communication systems, such as a mobile network or Wi-Fi connection.

Figure 5. HEIFU UAV executing an autonomous flight.

The main HEIFU specifications are:

- Cube Orange autopilot (running Ardupilot): used to control low-level operation and is responsible for ensuring a stable flight throughout the whole mission, following the pre-defined path from takeoff to landing. This hardware contains an Inertial Measurement Unit (IMU) and features a GNSS connection to provide the UAV's position and orientation. Furthermore, the Cube connects to the Jetson Nano embedded system via a MAVlink protocol; through the Ultra High Frequency (UHF) receiver, HEIFU can be piloted by a Radio Frequency (RF) controller.
- RTK Positioning: based on a u-blox ZED F9P RTK [75] receiver, it can achieve much better accuracy than conventional positioning since it counts on a base station capable of delivering RTK differential correction information. For this purpose, the RTK base station can be either a portable solution, like the beRTK [76] or simply using the ALPHA Landing Base, providing a control accuracy in the range of 10 cm. The positioning accuracy is paramount in core applications of the HEIFU portfolio (like precision farming [77]).
- Jetson Nano: used to control the high-level operation. It receives data from the distance sensor (depth cameras), the Red Green Blue (RGB) camera, and communicates with external devices via a WiFi link.
- WiFi/Mobile Network Communications: the HEIFU counts on a native 4G modem. If available, WiFi communications can also be used. HEIFU also features Bluetooth and is prepared to carry a 5G modem to benefit from the higher bandwidth.
- An RGB camera on a gimbal stabilizer.

For a better understanding of the communication between various system layers, Figure 6 represents the connectivity to the communication layers.

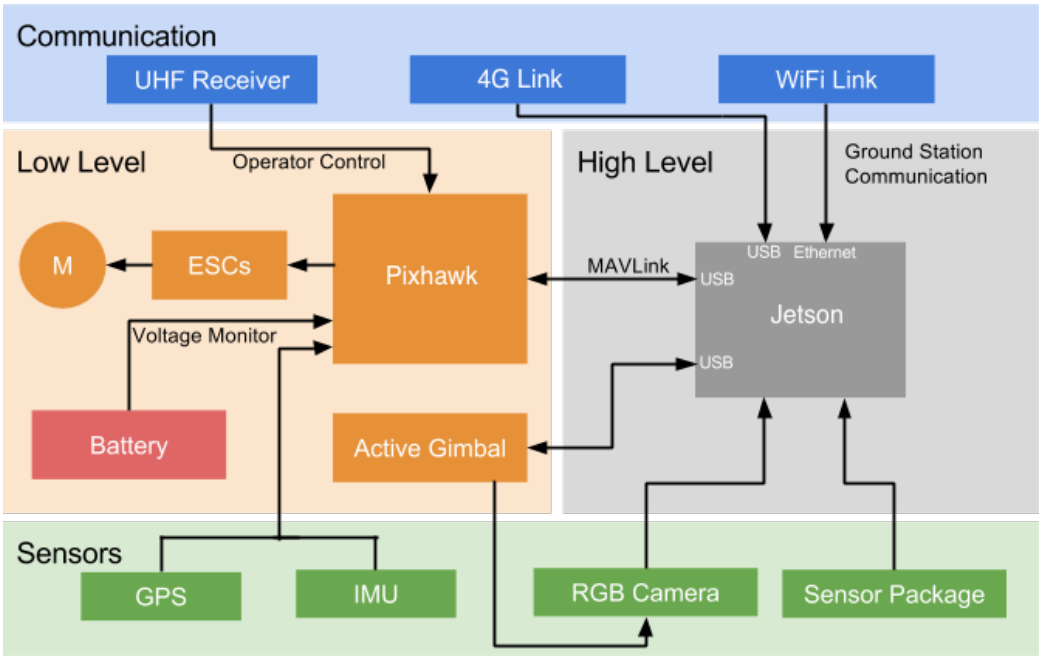


Figure 6. HEIFU high-level connectivity diagram to the communication layers.

2.4. ALPHA Landing Base

The ALPHA landing base is the last component of the ecosystem. It acts as a charging station for the UAV, eliminating the task that usually involves human operators, namely, the recharge of the UAV´s batteries. Its main features are:

- Solar Panels and respective Maximum Power Point Tracker (MPPT) for battery charging and self-powering;
- Power-grid connection;

- Smart Charging System, which decides which power source to use and if the internal batteries require charging;
- Internet connection (via WiFi or ethernet);
- Weather Station: to estimate the wind speed and detect the precipitation levels;
- Weight scale on the landing area. It helps the user quickly estimate the UAV payload and allows the landing area to detect if the drone has landed and estimated the maximum mission time for a specific payload and battery status;
- GNSS positioning and RTK Corrections;
- Landing Base with multiple trackable patterns;
- Mission scheduling.

The ALPHA landing base is equipped with lead-acid batteries that can be charged directly via power-grid or via solar panels placed on top of the shell, depicted in Figure 7). In the first picture (left), the UAV is enclosed inside the landing area. In the second (middle), the UAV is ready for a mission, and the landing platform has open the side panels. Finally, in the last picture (right), the landing area is fully open, and after the UAV has received the mission, it can safely takeoff.

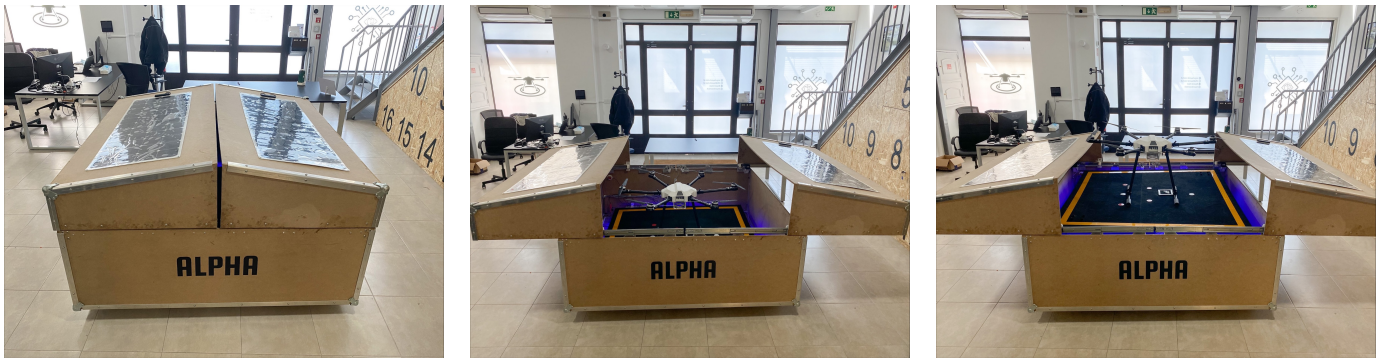


Figure 7. Landing platform prototype fully closed (**left**). Landing platform at the middle of the opening operation, with side panels open and the lift on the bottom position (**middle**). Landing platform fully opened, and ready for the UAV takeoff (**right**).

The landing base can also use different connections to communicate with the beXStream platform, allowing remote control. A set of sensors such as the weather station and a scale were incorporated on the landing base to provide helpful information that facilitates routine checks. The precision of the landing maneuver is of the utmost importance because even a small error (more than 10 cm) can compromise the automatic charging and cause the collision between the top covers of the Landing Base and the UAV. For this reason, the landing base has an RTK base module that sends corrections to the UAV, improving its positioning. Additionally, a set of three trackable patterns were introduced to the landing base that is fused with the positioning odometry, increasing its accuracy.

To perform autonomous missions on a UAV stored in the landing area, the messages depicted in Figure 8 are exchanged. These messages are explained in the following steps:

1. An availability check for the landing area and UAV to perform a mission. On this point, the UAV should check if it has enough battery (defined by a battery charge threshold), and the landing area should check the weather conditions.
2. Open the landing area. This is done in two steps:
   - Open side panels.
   - Lift the ground platform.
3. Send a mission to UAV.
4. Perform mission.
5. Landing procedure.
6. Confirm UAV successful landing.
7. Close the landing area. This is done in two steps:

(a) Lower the ground platform.
(b) Close side panels.

8. Start charging UAV's batteries.

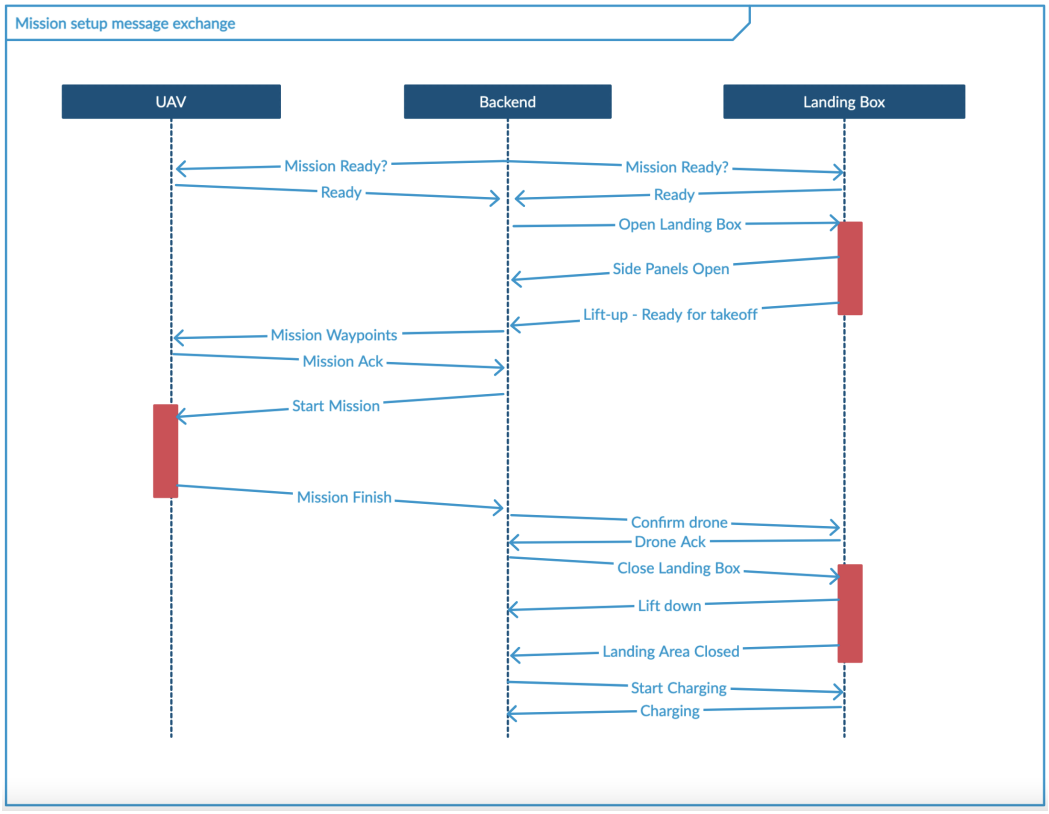


Figure 8. Messages exchanged between the landing area, the platform (backend component), and the UAV to perform an autonomous mission without human intervention.

2.5. Computer Vision Positioning System

A precision positioning system and maneuver were developed for successfully landing the HEIFU on a specific landing pad that allows storage and charging. This maneuver needs to finish with a rendezvous at the centimeter-level accuracy in order to align the electrical contacts of the UAV and the landing pad, thus enabling the charging of the UAV's battery. As stated previously, the landing error needs to be smaller than 10 cm. However, the authors want to land with an error smaller than five centimeters. Despite providing this level of positioning error, an RTK might not be enough due to setup errors. The setup errors can be mainly caused by a wrongly setting of the landing pad center coordinates. Since the landing pad does not have any information about its orientation, setting the translation between the landing pad center and its GNSS antenna is also not possible. To solve this problem, the setup of the landing pad enclosure needs to be placed on specific previously geo-referenced points, which is labor-intensive and expensive work. As an alternative that allows a more relaxed landing box's position setup, a visual pattern was constructed to be processed through computer vision for extended position accuracy. The computer vision positioning system is composed as follows:

- A visual pattern created using reflective material drawn on the landing pad surface;
- A bottom facing camera on UAV capturing images at 30 Frames Per Second (FPS).

Using the previous knowledge of the visual pattern, computer vision techniques are used to obtain the position of the UAV referenced to it. This method provides a way to successfully land the UAV even if the landing pad's global coordinates are not accurate at a centimeter level. The landing pad only needs to be detected by the UAV camera, correcting the final landing position.

2.5.1. Visual Marker Description

Since the pattern needs to be seen by the camera from the beginning to the end of the maneuver, the visual marker used (Figure 9) is a combination of three elements, each having a specific function, allowing the continuous identification and usage of the marker for a continuous UAV's positioning.

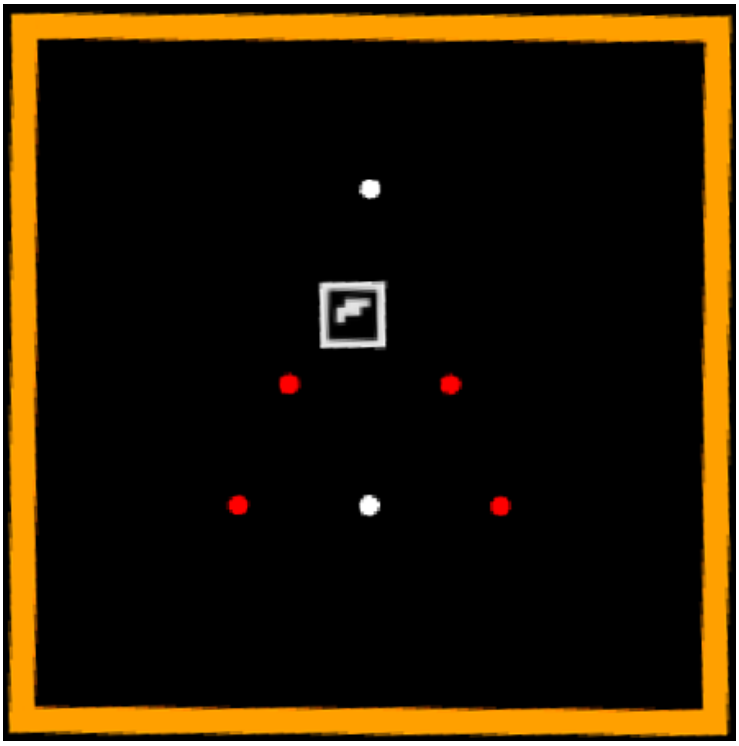


Figure 9. Visual marker installed on the landing pad.

The three main elements are:

- **Masking frame**—Its primary purpose is to mask the image by using only the area of the image where the marker is, reducing false positives and processing time.
- **Custom pattern** defined by six points—The marker has a dimension that allows detection during the first step of the landing maneuver, even when the UAV is at an altitude of 5 m.
- **Fiducial marker**—An Aruco marker [78,79] is placed in a position on the landing pad where the UAV camera can have it on the image after the successful landing at the desired position.

The combination of the three elements on the same marker an event timeline defined by three main events:

- Initially, at an altitude of approximately five meters, the masking frame and the custom pattern are the only detected elements due to the altitude. Additionally, since the Aruco is not a circle, it is rejected by the custom pattern algorithm.
- At mid-distance (less than two meters), the masking frame is not seen, but since the pattern has a black background, the pattern occupies all the image, so there is no need for it. At this distance, the Aruco can be detected alongside the custom pattern, both providing position through vision.
- In a final phase, at an altitude smaller than 50 cm, only the Aruco is seen on the image until the end of the maneuver, when the UAV finally lands.

In order to achieve maximum contrast between the pattern elements and the background of the marker, the elements are constructed using a high visibility reflective tape and the background with black velvet vinyl.

The custom pattern combines six points divided into two sets of different colors (red and white). The used colors ease the process of grouping the points by color during the pattern detection process. The two colors are at the two ends of the Hue-Saturation color spaces (white is a high hue and low saturation, and red is a low hue and high saturation). Due to the usage of the reflective material, all the elements have a high *Value* channel.

The marker (Figure 10) was designed keeping in mind that it should not be fully symmetric since this would not allow obtaining the heading, which is needed for the alignment with the charging pins. The symmetry axis is defined by the two white points (P1 and P5). Two circles and a triangle are the geometric forms that constrain the pattern design:

- P1, P2, and P3 are the vertices of the triangle that is inside of a circle centered on the middle of the landing pad;
- P4, P5, and P6 are the points where a circle intersects the previous triangle.

The start and stop altitudes where the different elements of the marker are detected are constrained by their size. The most critical constraint for the marker size definition is the landing gear. Since the landing gear needs to be ready in the final phase, the camera should not capture it on the image since it will occlude the marker. Therefore, the camera position and lens aperture are chosen not to be captured in the image. Using the camera Field of View (FOV) and height, the Aruco size is chosen to fill almost all images, in this case, a marker with ten centimeter side. The custom pattern size needs to be compromised between being big enough to be seen at the highest altitude possible and still being detected while the Aruco is also detected. The chosen size to meet this requirement is the circle which comprises P1, P2, and P3 with a diameter of 75 cm.

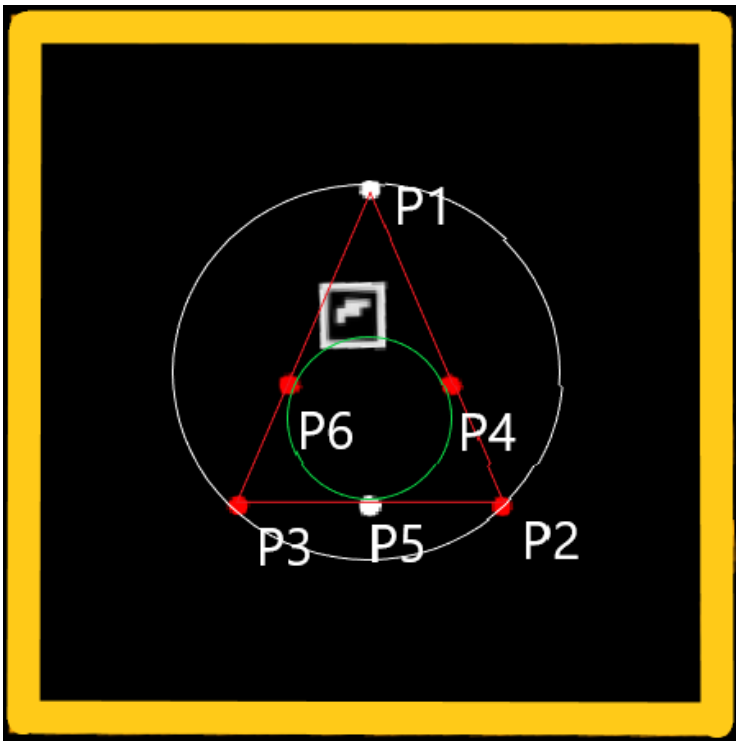


Figure 10. Description of the elements of the custom pattern.

2.5.2. Image Processing

The UAV carries an Nvidia Jetson Nano as OBC, allowing the usage of the Graphics Processing Unit (GPU) with Compute Unified Device Architecture (CUDA), improving the execution time. The image processing pipeline depicted in Figure 11 can be divided into four main parts:

1. Image preparation at green—the lens distortion is removed to assess the angular and linear relations of the pattern points. The image is converted to the Hue Saturation Value (HSV) colorspace for more invariance to color thresholding;
2. Area of interest detection and custom pattern identification in yellow;
3. Aruco marker detection in gray;
4. UAV pose computation using the pattern position on the image plane.

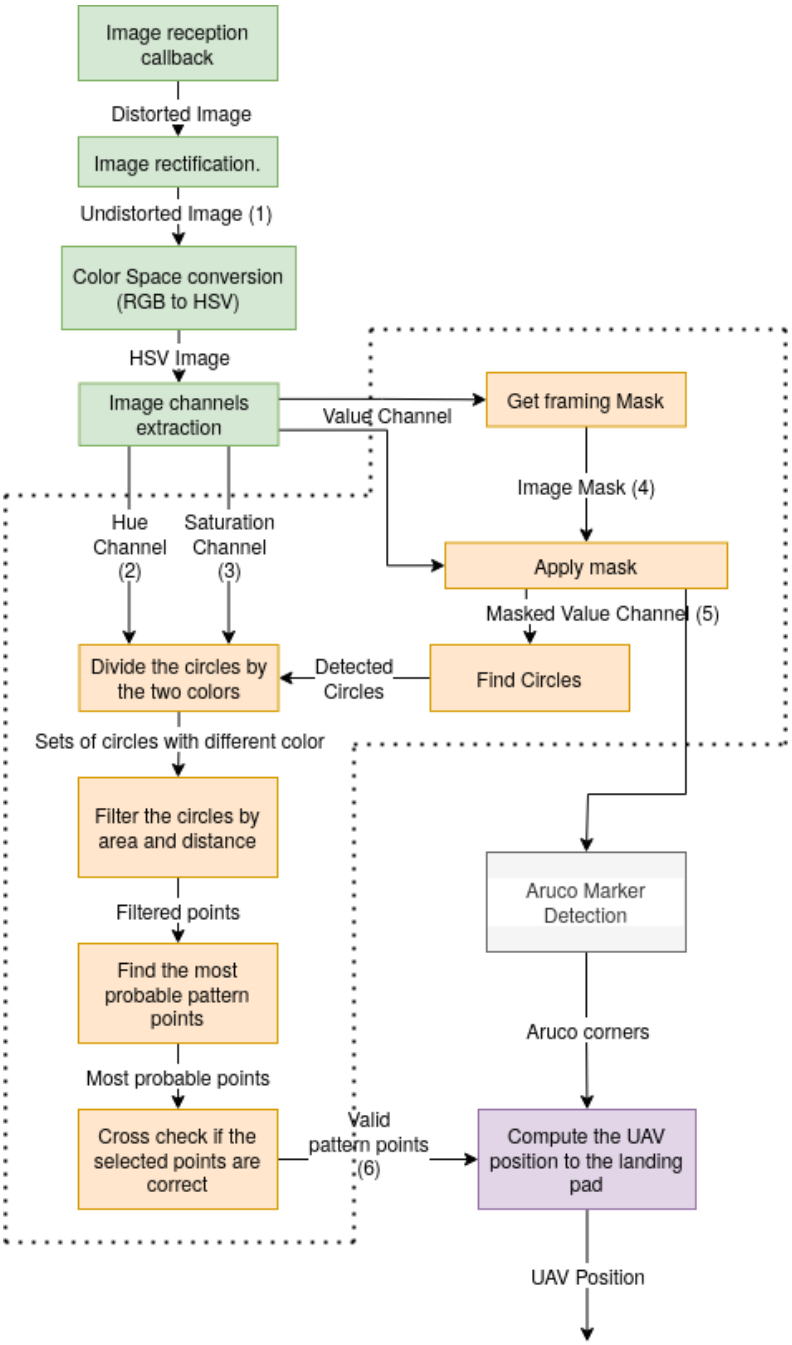


Figure 11. Image processing pipeline for UAV position estimation.

The captured image is affected by the lens's characteristics due to its distortion. [80] To analyze the frame and assess angular and linear relations of the pattern points, it is necessary to rectify them. In the case of the camera used on the final tests (Intel Realsense D435i), the image is rectified on-board, thus making the undistortion process unnecessary. Nevertheless, this step is not omitted from the pipeline since it is necessary for the majority

of the cameras. Furthermore, the rectified image (Figure 12) is converted from the RGB colorspace to HSV due to its flexibility to changes of illumination, which is more suitable for outdoor image color thresholding [81].

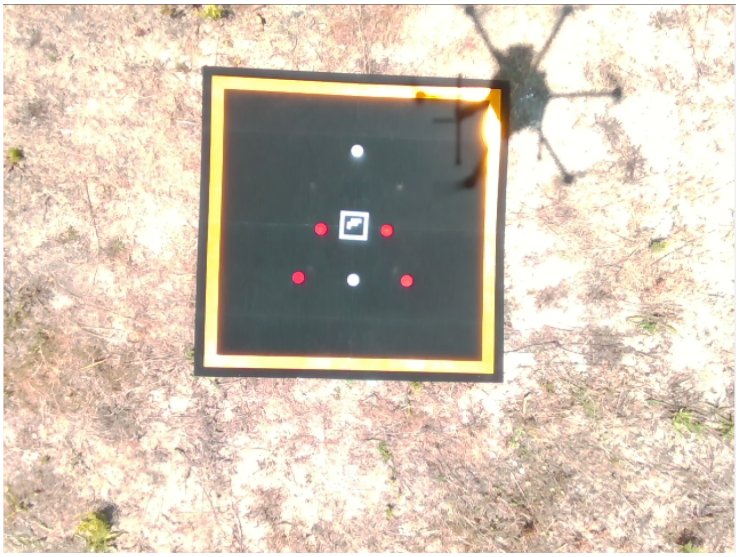


Figure 12. Landing area ground pattern image after applying the rectification algorithm.

The three HSV channels were decomposed as depicted in Figure 13. From the HSV channels, the *Value* is used to mask the potential points since the visual marker is constructed with highly reflective material. Therefore, from the visual analysis of the *Value* image, it is possible to conclude that the ground where the landing base is can make it difficult to detect the marker due to false positives identified.

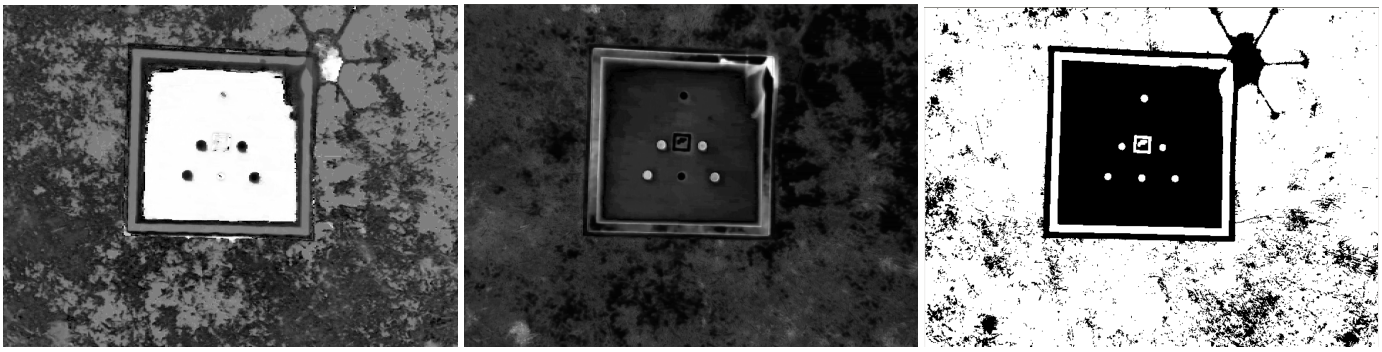


Figure 13. HSV figure decomposed channels (from **left** to **right**, hue, saturation and value).

Using a masking frame is beneficial to deal with noisy backgrounds. It is also the first element from the visual marker the algorithm searches for since it will help detect the other elements. Detecting the masking frame consists of searching for contours on the value channel image that form a square. Since for a given scenario, other squares can appear on the image, all the detected squares are drawn on the masking image (left on Figure 14). To ignore the lens flare (visible on the top right corner of the masking frame on the value channel image from Figure 13), the masking frame is eroded, re-positioning the corners of the square, reducing the mask area, and reducing the effect of the lens flare that could create false positives. The image from the right is the value channel from Figure 13 with the mask applied, allowing to highly reduce the potential points in the image and, therefore, reducing the processing time.

Figure 14. Masking image on the (**left**) and masked value channel image on the (**right**).

From this point, the masked value channel image is used by two threads: the Aruco detection and the custom pattern detection. The Aruco detection uses the OpenCV [82] Aruco library. Thus, this process will not be described in this document. However, since the custom pattern points are circles, the next step is to find circles to exclude the Aruco pattern from the list of possible points.

The list of possible points is used in order to group them into two different sets according to their color. These are done using the characteristic exposed previously. The red and white colors from the circles are at the two ends of the Hue-Saturation. This gives flexibility to the circles' threshold since a strict color value does not need to be used. In order to achieve even higher flexibility, K-means [83] is used, trying to find two sets of points organized using the color. The *Hue* value is defined by an angle on the color spectrum. Due to its circular representation, values closer to the angular limits (0 and 360) translate to the same color. To solve this discontinuity problem, the K-means uses the sine of this *Hue* angular value.

Since the visual landing pattern is known, using the two sets of points makes it possible to use the known angular and linear relations between the points to find the most probable matches (Figure 15).

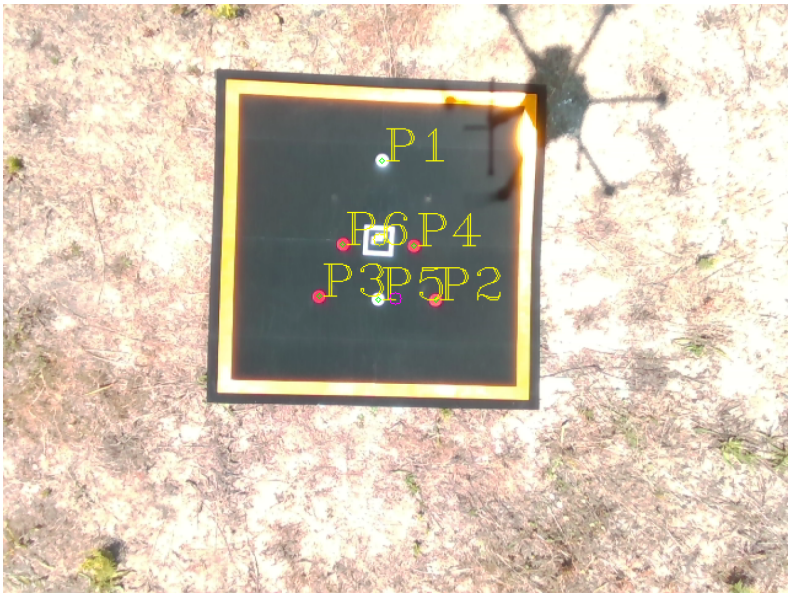


Figure 15. Landing ground pattern with the tagged custom pattern points overlaid.

The information regarding the position on the world frame and the position on the image frame of the detected points is known. Since this marker is planar and usually parallel to the camera frame, the usage of the OpenCV solvePnP method is not the best

choice, as it can output ambiguous 3D position solutions due to parallax. However, the image plane does not suffer from this effect, so a simple ratio between the size of the pattern in the real world and the image is used to convert the image horizontal position error to the world frame:

$$WorldPosition_{xy}(m) = imagePosition_{xy}(px) \times \frac{worldPatternSize(m)}{imagePatternSize(px)} \quad (1)$$

In order to obtain the altitude of the camera to the pattern, the information from the lens' focal length is used:

$$WorldPosition_{z}(m) = focalLength(px) \times \frac{worldPatternSize(m)}{imagePatternSize(px)} \quad (2)$$

Due to the parallelism between the landing pad and the camera frame, there is no need to calculate roll and pitch angles for this maneuver. This approximation is enough to center and orient the UAV in relation to the landing base, as the camera is mounted on a gimbal. The UAV's position obtained through this method is the output of a Kalman filter that deals with the offset between the position obtained using vision and the position obtained through the UAV's GPS. Thus, the vision position is used as an update. Only the variation of the global position is considered to predict, thus reducing the effect of a possible base position setup error.

2.6. Control and Trajectory

Computing the camera position in relation to a known marker has an error correlated with the distance and the image resolution due to the pixel error. A higher distance translates to one pixel capturing more area, i.e., discretizing the position on the world referential, introducing error and noise on the obtained position. In order to ease the maneuver, the allowed position error regarding the landing pad center is not constant throughout all the maneuvers. This high flexibility at higher altitudes enables pattern detection even if the provided landing pad coordinates have an error of a couple of meters. The defined volume of acceptance has a conic profile as depicted in Figure 16.

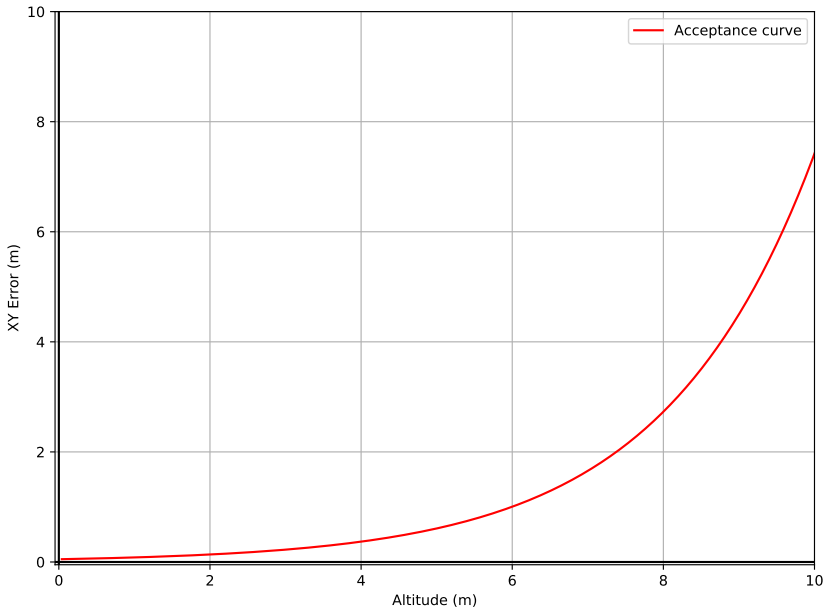


Figure 16. Cross-section representation of conical volume of acceptance.

The volume depicted on Figure 16 is defined by Equation (3), which is has input of the UAV altitude a.

$$AllowedPosError(a) = minPosError \times e^{maxPosError \times a} \quad (3)$$

The $minPosError$ parameter defines the 2D error when the UAV is landed, and the $maxPosError$ defines the aperture of the cone. Those two parameters define the allowed position error during the maneuver.

For a smoother control, the maneuver is performed using a control in the velocity space. The speed is limited to a variable value to adjust and smooth the descending maneuver. During all the maneuvers, the UAV is always trying to center itself with the landing pad. The altitude setpoint is adjusted according to the horizontal and orientation error. If the UAV fails to meet the maximum allowed orientation error, it will stop descending and adjust its orientation. In the case that the UAV horizontal error is not accepted, it will only descend, setting the vertical setpoint to a value that is:

$$AltitudeSetpoint(a) = a - AllowedPosError(a) \quad (4)$$

Figure 17 depicts the scenario where the UAV is outside of the acceptance cone. To ease visualization, the dark blue line defines the perimeter of the semi-circle to state that the setpoint is translated to a vertical position that corresponds to the acceptance curve value for the current UAV altitude. Thus, as described previously, the position setpoint is not (0, 0, 0), but (0, 0, 4.392), reducing the vertical component of the velocity vector, smoothing the descending trajectory.

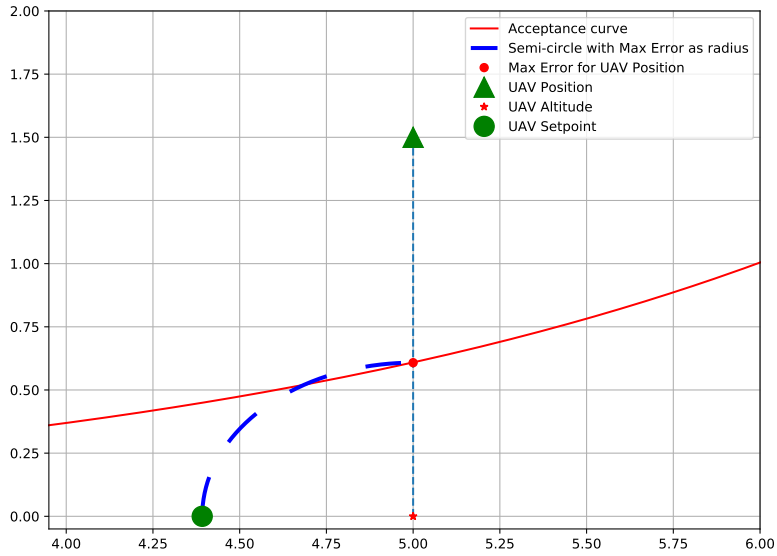


Figure 17. Example of scenario where the UAV is outside of the cone during the landing maneuver.

3. Application Environments

This section describes where the proposed system algorithms were used. It is organized into two parts, where the first one describes the considered simulation environment, and the second one presents the real environment.

3.1. Simulation

The simulation environment was constructed under the open-source Gazebo 3D robotics simulator due to its compatibility and straightforward integration with ROS [84–87]. The UAV and landing base station models were added to the Gazebo simulator, as can be seen in Figure 18. The camera setting for the simulated UAV is simplified since it is not mounted on a 3D gimbal. It allows testing the effect of nonplanar images on the computer

vision position algorithm. In order to be able to view all simulated UAV information by topics, services, and actions (for example, acceleration sensors, image provided by an RGB camera) in real-time, a 3D visualization tool for ROS, known as RViz, was used [88]. The algorithm can be directly applied and tested using this process without significant differences between real and simulated scenarios.

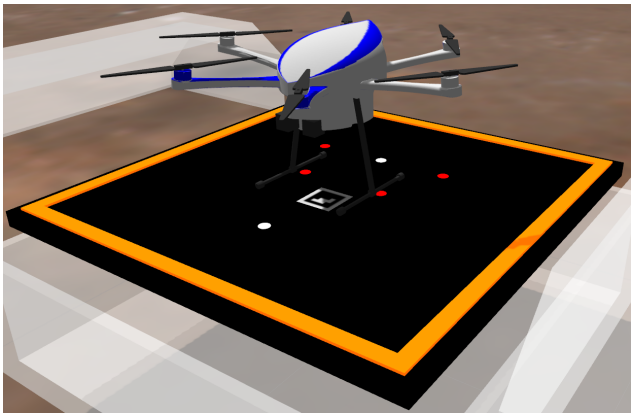


Figure 18. HEIFU prototype in Gazebo simulation.

When the UAV needs to go to a base station to charge the battery, it receives the closest landing base geographic position to navigate it, calculating a trajectory to the desired destination. At this stage, only the geographic position is taken into account in the UAV-based relative positioning estimation. Below a certain distance threshold, as explained in Section 2.6, the visual information from the camera is used to detect the landing base, helping the UAV positioning relative to the base station with a Kalman Filter (data fusion between the camera and GPS information), until the complete landing (Figure 18). The results presented below are obtained in the simulation environment, where the ground truth is the Gazebo model position.

3.2. Real World

Given the positioning errors between the measured data and what is expected in the simulation environment (due to the ability to obtain the ground truth of the simulated UAV), in the real environment, the veracity and robustness of the algorithm proposed in this document were tested.

The landing process is similar to the one described for the simulation environment. When the UAV receives the position of the base station, it should land on top of it, as can be seen in Figure 19.

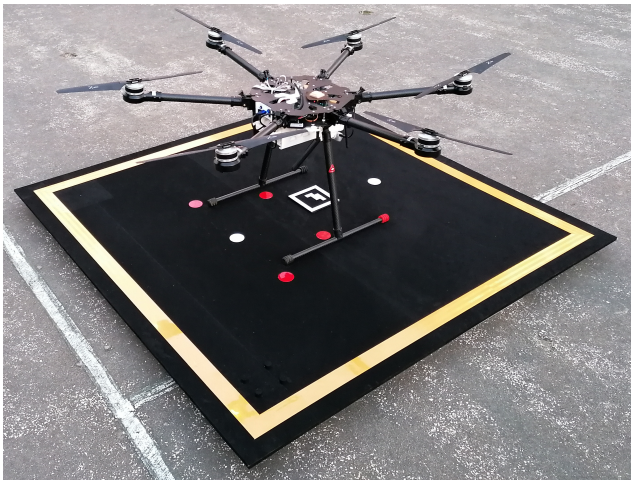


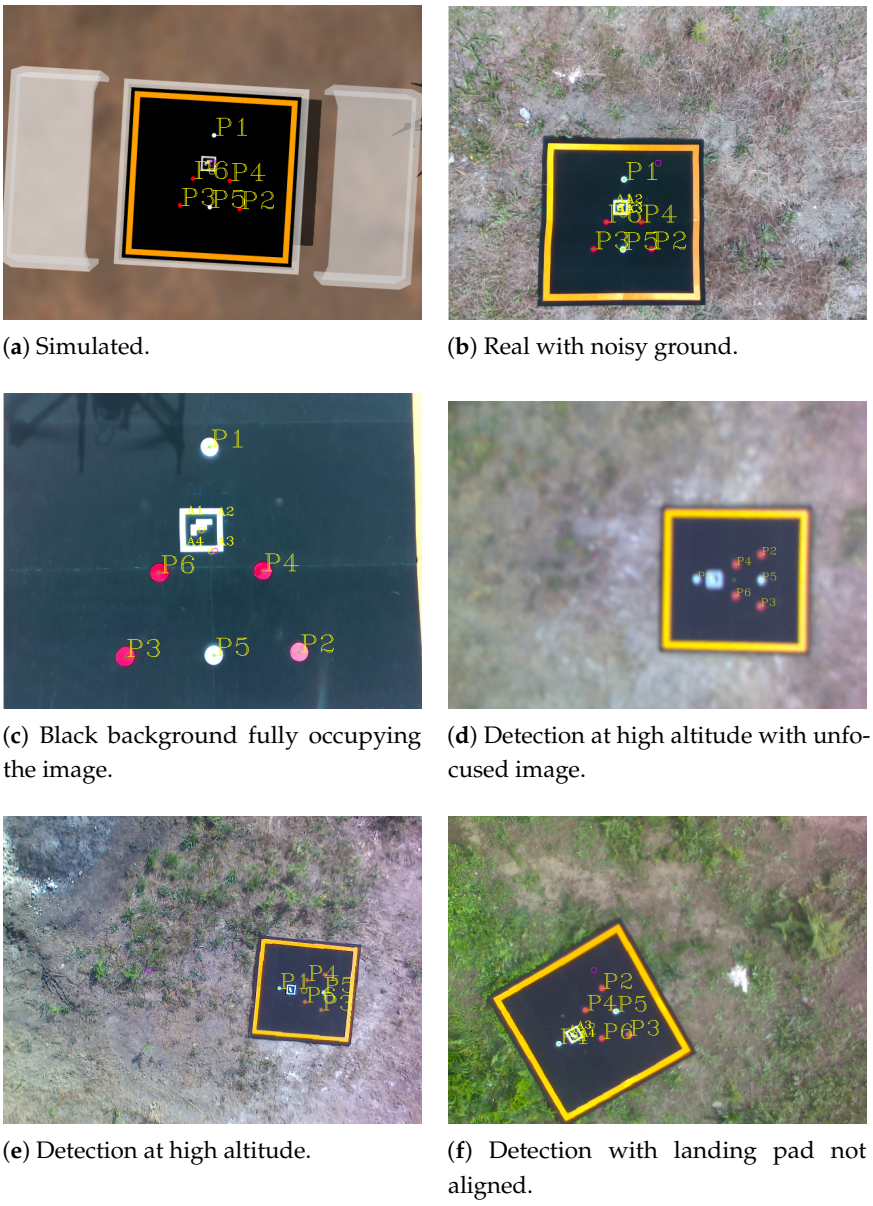
Figure 19. HEIFU prototype testing in real environment.

There are several factors to consider in the real environment that cannot be easily replicated in the simulated environment (although the Gazebo simulator is very close to the real environment). For example, variables like wind, GPS error, noise obtained by the images coming from the camera, or the inertia of the UAV with the real payload cannot be accurately predicted and might affect the simulation behavior.

4. Results

This section presents the main results of the different components of this work. The evaluation of the method was divided into two components: the pattern detection on the image and the error of the position obtained through this method. The pattern detection results are from tests performed in simulation and real environments. Due to ground-truth constraints, it was impossible to use the RTK position as ground-truth on the real-world tests. Thus, only the images will be presented as results for this scenario.

Figure 20 has examples of the visual marker detection in different environments proving that it is capable of detecting with different conditions (i.e., altitude, focus, brightness, background noise).



(**a**) Simulated.

(**b**) Real with noisy ground.

(**c**) Black background fully occupying the image.

(**d**) Detection at high altitude with unfocused image.

(**e**) Detection at high altitude.

(**f**) Detection with landing pad not aligned.

Figure 20. Detection of the marker in different environments and conditions.

The Aruco marker, as stated previously, was designed to be visible on the final stage of the maneuver until the end. Due to this constraint, when landed, the marker must be visible on the camera, as presented in Figure 21.

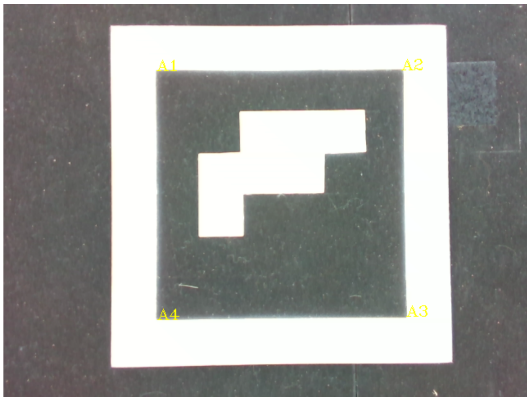


Figure 21. Aruco marker detection when the UAV is landed.

Using information from flights performed on the simulator, a statistical representation of the number of frames where the Aruco and the custom pattern were detected per second related to the UAV altitude are presented in Figure 22. As described in Section 2.5.1, initially only the custom pattern is detected, with the detection rate getting higher as the altitude decreases. The number of invalid pattern detection decreases on pairs with altitude. In the figure, it is also possible to observe the "handover" between the detection of the custom pattern and the Aruco, which always allows having position through the computer vision method. The used camera resolution constrains these results. A higher resolution camera could detect the marker at higher altitudes, with the drawback of increasing the processing time.

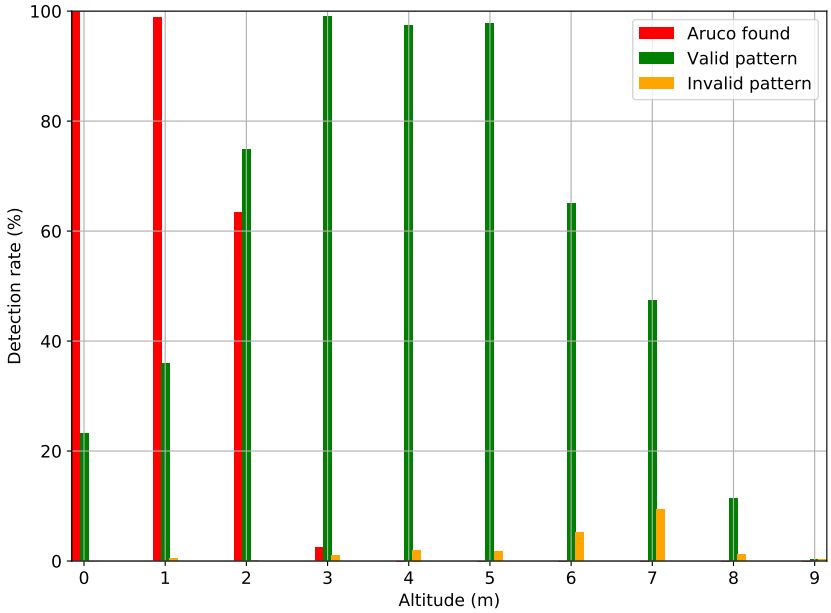


Figure 22. Relation between altitude and visual marker elements detection rate on simulated environment.

On the real environment tests, the detection rate (depicted in Figure 23) does not perform as expected. However, the custom pattern detection performs similarly to the simulated scenario data, with a lower success rate and starting at a lower altitude. This performance can be related to two factors, the lighting conditions, and the rejection conditions. First, due to different lighting, the custom marker elements are not detected at

a higher altitude (higher than five meters), only being detected at a higher rate for an altitude of around three meters. Second, due to the pattern points' strict rejection and attribution rules, the invalid pattern detection rate is higher in the real scenario, even if all points are detected. This is done by design to favor the most accurate detected frames to achieve higher accuracy. The Aruco detection rate also performs differently from the simulation, decreasing along with the altitude. This is due to phenomenons such as lens flare or neighbor pixel saturation, as presented in Figure 24.

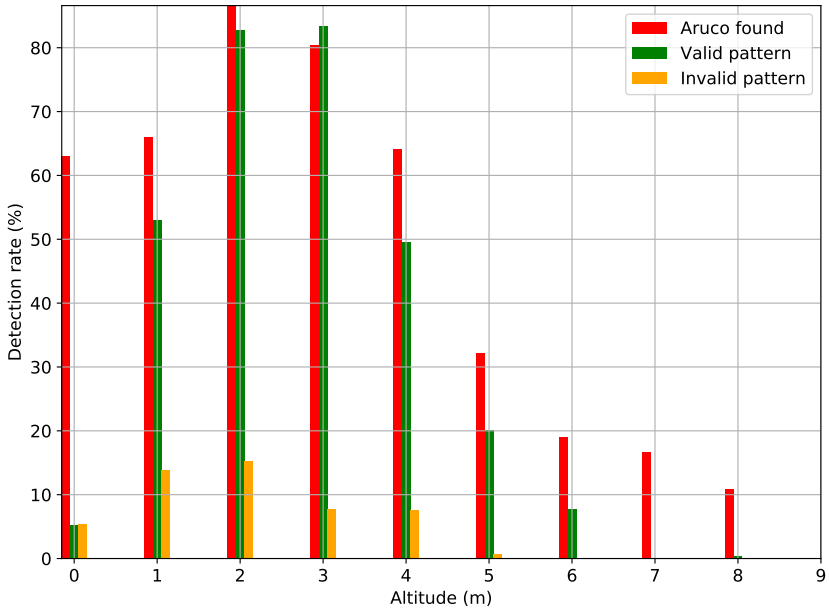


Figure 23. Relation between altitude and visual marker elements detection rate on real environment.

Figure 24. Saturated image with Aruco marker on real world.

As detailed at the beginning of this section, only data from the simulation environment will be used to evaluate the positioning quality of this method. In the simulation environment, the maneuver performs the same way every time since there is no effect on lighting changes or GNSS receiver position noise. However, the setup error is simulated, providing a landing pad position with an error to the maneuver. The trajectory described in the plot in Figure 25 is from a flight where the UAV does a straightforward maneuver with a profile similar (Figure 26) to the safety cone described in Section 2.6. The landing pad's pattern detection is easily identified in the following plots since the estimated position is updated and approximates the ground truth.

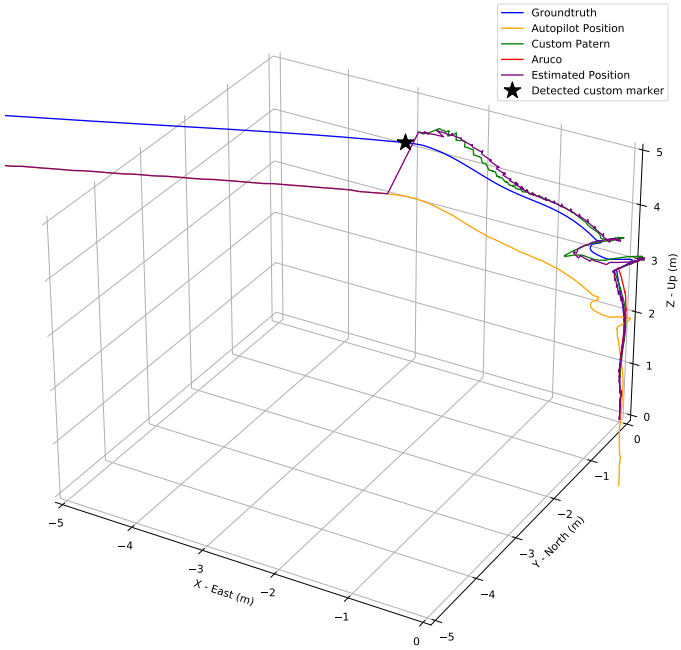


Figure 25. 3D trajectory of the UAV on the simulated environment during one maneuver

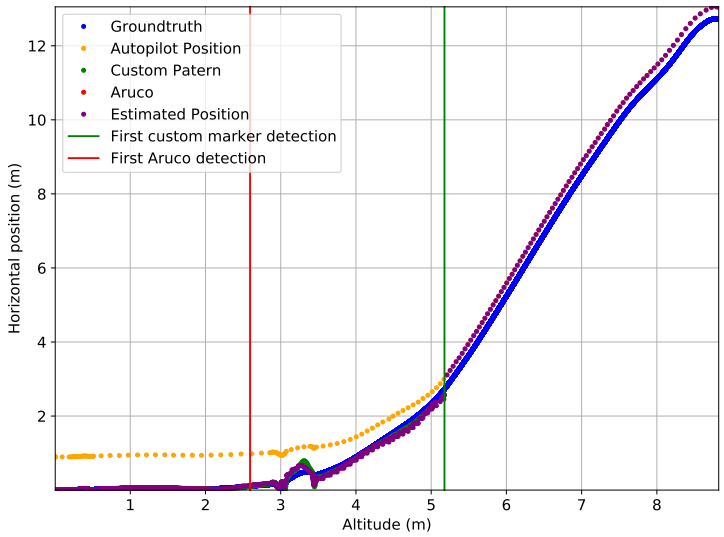


Figure 26. 2D trajectory of the UAV on the simulated environment during one maneuver, depicting the conical volume of acceptance

Between the three and four meters of altitude, on Figure 26, there is an anomaly that does not follow the safety cone. This phenomenon is caused by the combined control of heading and position. Since the camera is not centered with the UAV rotation axis, the UAV rotation moves the camera away from its centered position inside the cone. In addition, the position correction induces a roll and pitch movement that, since the camera is not mounted on a gimbal, causes an increase of the position error, as exposed later in this section.

Figure 27 presents the decomposition of the 3D position of the UAV during the final stage of the landing maneuver. From these plots, it is possible to notice the slow descending of the UAV on the final 50 cm. This requirement is implemented to prevent the loss of the Aruco marker from the image plane since it fills almost the entire image area. On the vertical plot it is possible to notice the jump on the estimated position as soon the custom marker is detected. The horizontal plot also depicts a position spike caused by the UAV

attitude change. In the simulation environment, the UAV does not have a gimbal, causing an increase of the position error due to the perspective effect.

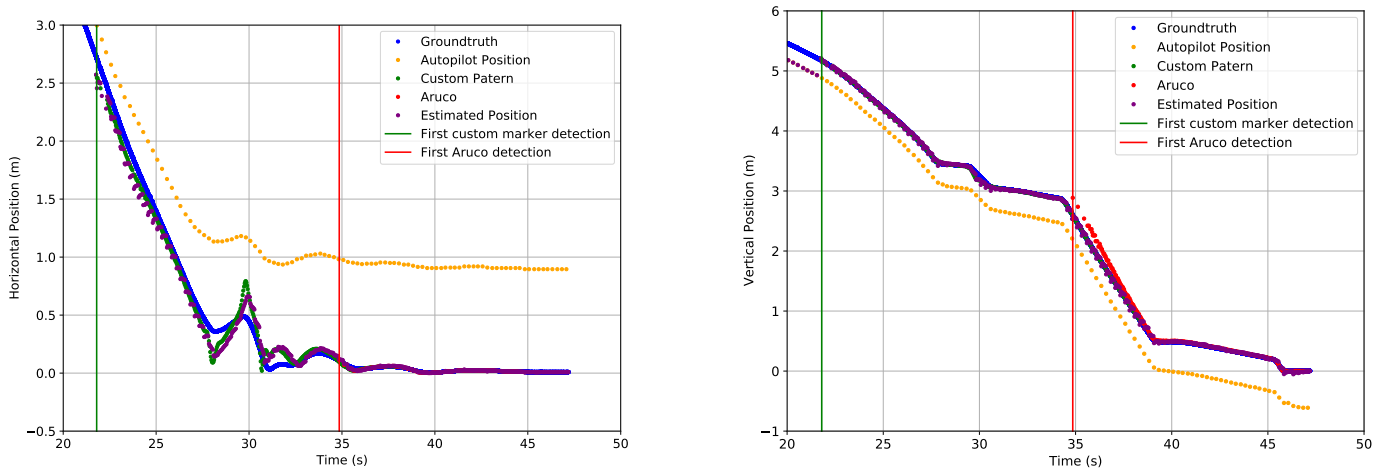


Figure 27. Positions along time on the final stage of the maneuver. Horizontal position along time (**left**). Vertical position along time (**right**).

The position and heading error curves for this maneuver are depicted on Figures 28–30 illustrate the error of the three sources of position for the estimator and how they change during flight. As described previously, the GNSS position of the landing pad is sent to the UAV with an error, emulating the setup error. This error of 90 cm is the mean value of the horizontal error presented on the autopilot position line of Figure 28. While the only source of pose estimation is the GNSS and IMU from the autopilot, the estimated position follows its value. When the custom pattern starts being detected, the estimated position fixes to the value obtained through its detection using only the GNSS and IMU position variation as an input. Analyzing the vertical error curves from Figure 29, it is also possible to notice that the Autopilot Position error is not constant. This is due to the intrinsic position noise of the GNSS receivers that can be assumed as white noise. Even when the GNSS altitude curve keeps increasing its error, the estimated position error keeps a value constant when the custom pattern is detected. It is also possible to notice that until the UAV does not reach a lower altitude (less than 50 cm), the Aruco position has a high error compared to the pattern position. This is due to the pixel error that is related to distance to the visual marker.

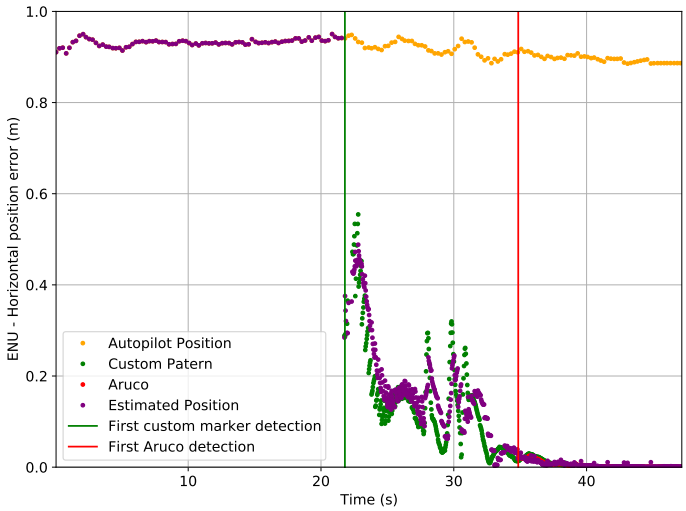


Figure 28. Horizontal position error during the maneuver.

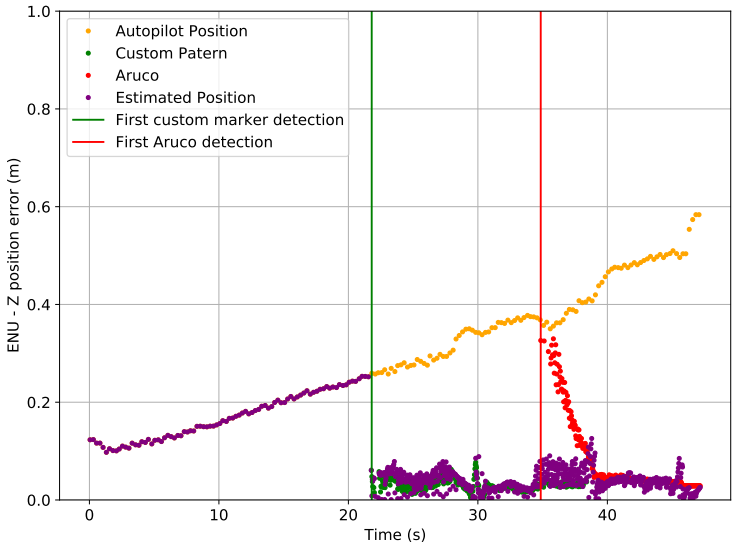


Figure 29. Vertical position error during the maneuver.

During the maneuver, the UAV keeps the heading until it detects the visual marker since it does not have information regarding the landing pad orientation, as exposed in Figure 30.

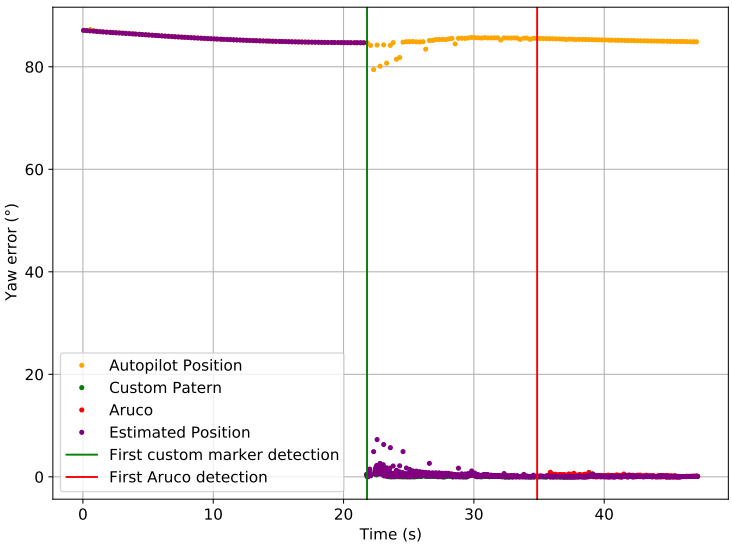


Figure 30. Heading error during the maneuver.

Using the information from various flights performed on the simulated environment, the following error curves were drawn regarding the position obtained through computer vision since it is the only method dependent on the altitude. As the maneuver was started at different altitudes, outliers occurred at different altitudes due to the error caused by the roll and pitch from the position correction.

The pixel error is the leading cause of an error when computing the position using an accurate visual marker detection. Since the visual marker has a fixed size, with the increase of altitude, the size of the marker on the image plane decreases, increasing in this way the distance to which each pixel matches, as described in Figure 31.

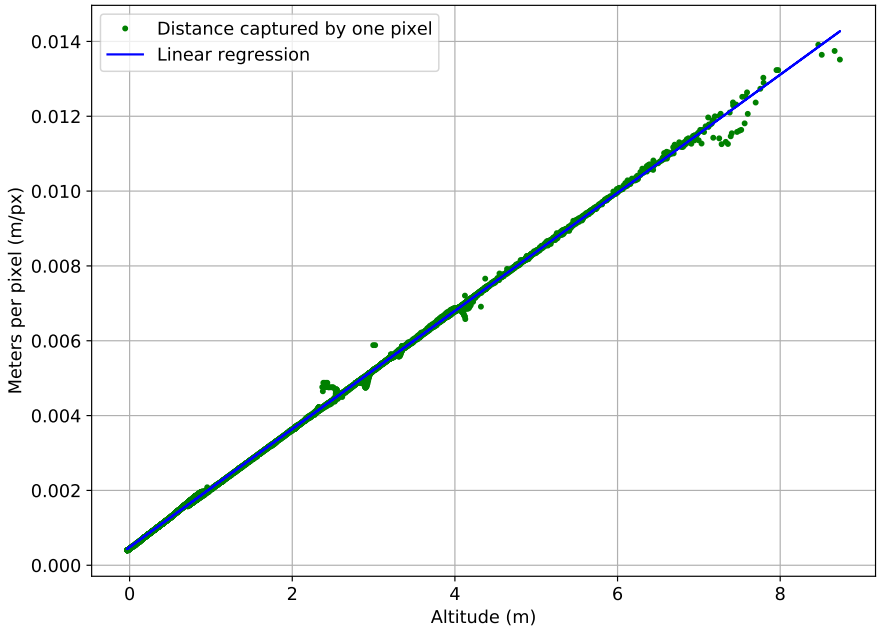


Figure 31. Pixel error vs. altitude

Even so, the effect of pixel error affects the four states that are computed (depicted on Figures 32–34). The one that exposes the effect with a higher magnitude is the plot from the vertical position error in relation to the altitude, namely for the Aruco, where it is possible to see a linear relation between the vertical position error and the altitude.

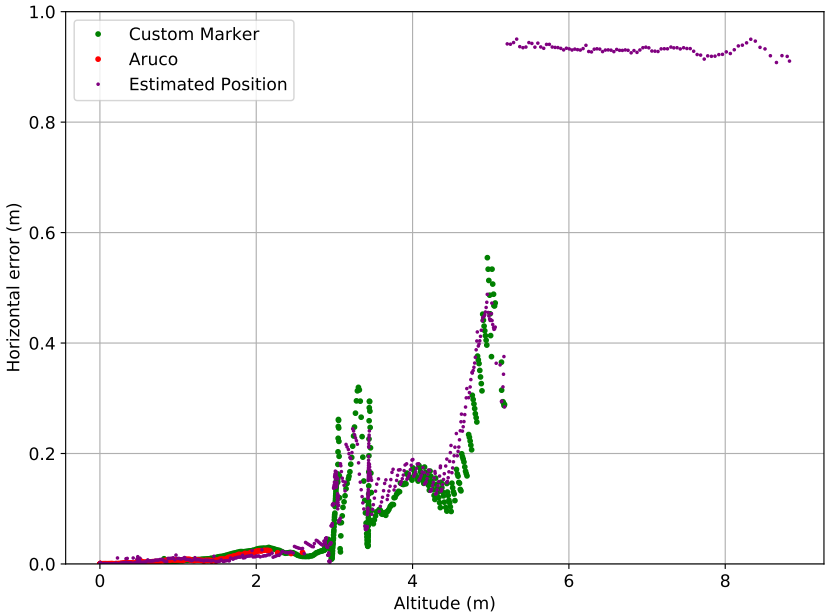


Figure 32. Horizontal position error relation with altitude.

The position error of the sources for the pose estimator is presented in Table 1. Here, it is possible to notice a fixed offset due to the unknown position and orientation of the landing pad for the GPS and IMU positioning system (previously named autopilot position). It is possible to notice that the horizontal position obtained using the custom marker has almost eight times smaller horizontal error and the Aruco almost thirty times smaller when compared with the GPS and IMU positioning. Furthermore, in the case of the Aruco is possible to notice that the vertical error has a higher amplitude than the custom marker due to the effect depicted in Figure 33.

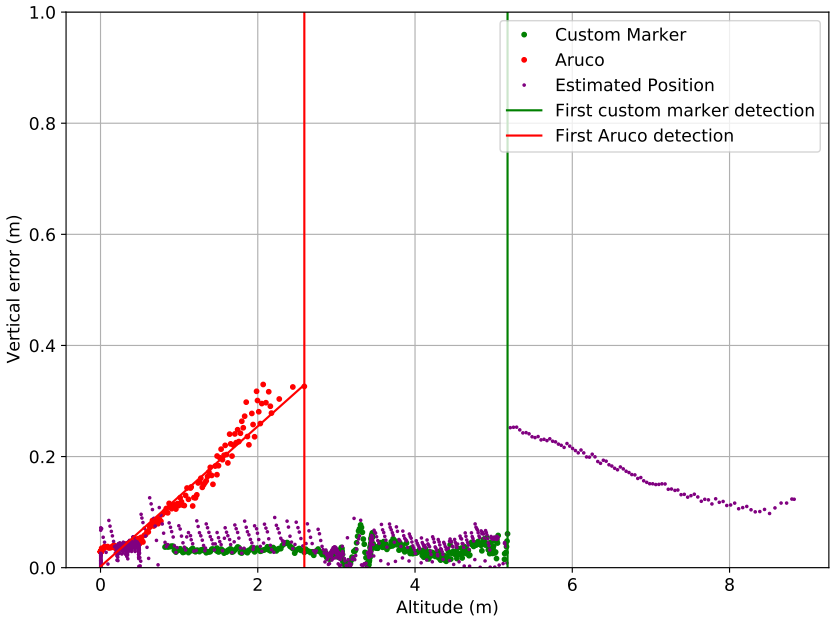


Figure 33. Vertical position error relation with altitude.

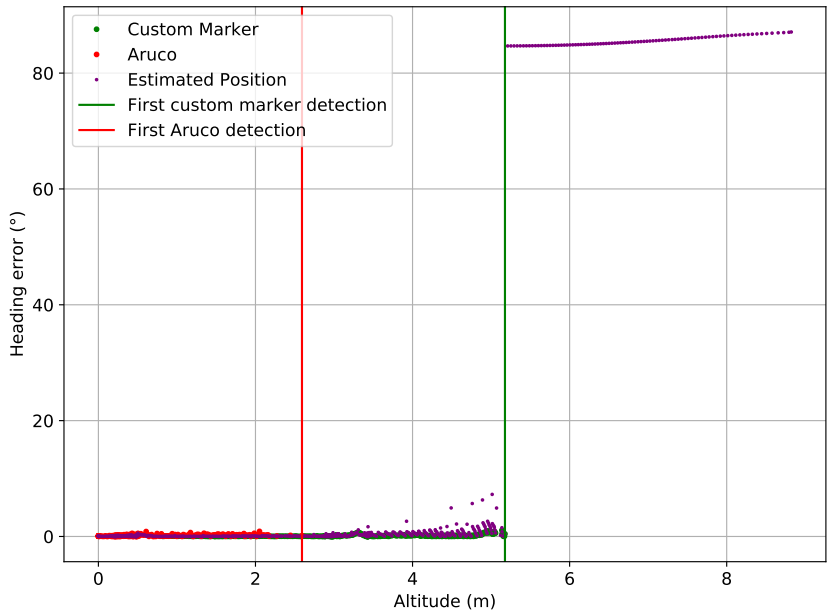


Figure 34. Heading error relation with altitude.

Table 1. Error of the position sources used by the position estimator.

Source	Horizontal (m)	Vertical (m)	Heading (°)
GPS and IMU	0.918 ± 0.017	0.286 ± 0.129	85.283 ± 0.785
Custom Marker	0.116 ± 0.113	0.029 ± 0.011	0.124 ± 0.157
Aruco	0.004 ± 0.006	0.078 ± 0.074	0.155 ± 0.136

Table 2 presents the error of estimated pose during the maneuver:

- Initially, only the GPS and IMU pose is used to compute the estimated pose.
- When hovering the landing pad, the custom pattern is detected, and the computer vision position aids the estimation of the position, reducing the position error.
- In the final stage, while all the three sources are used, the position error is even lower.

- On the *Final Stage* only the Autopilot and Aruco positions are used for the position estimation. Therefore, this line on the table represents only the position statistics for the final stage.

Table 2. Error of the estimated position during the landing maneuver.

Source	Horizontal (m)	Vertical (m)	Heading (°)
Only GPS and IMU	0.931 ± 0.007	0.169 ± 0.047	85.544 ± 0.738
With Pattern	0.304 ± 0.314	0.060 ± 0.058	16.137 ± 32.999
All sources	0.173 ± 0.278	0.052 ± 0.046	9.066 ± 25.949
Final Stage	0.005 ± 0.006	0.041 ± 0.019	0.069 ± 0.061

The *Final Stage* horizontal position error shows that it is possible to position the UAV on the final stage with a position error of 0.005 ± 0.006 m. Figure 35 depicts the horizontal distance of the UAV to the landing pad center on a few maneuvers in the simulated environment. In all the performed tests, the UAV could land with a position error smaller than three centimeters, complying with the requirement of landing with a maximum error of five centimeters.

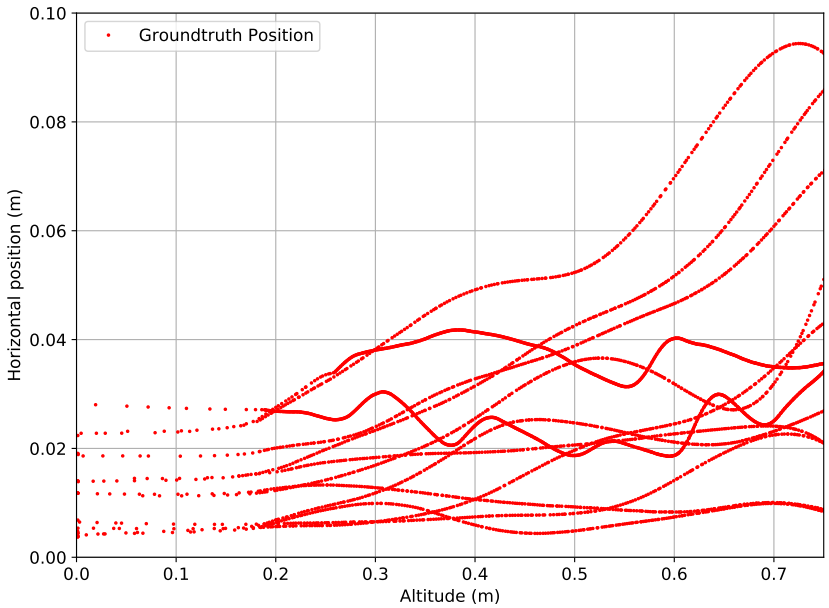


Figure 35. Ground truth position on the final stage of the landing maneuver.

5. Discussion

The communication range between the UAV, platform, and landing base is variable with the area's 3G, 4G, or WiFi coverage. The telemetry data consume a low amount of bandwidth (less than 200 kbytes/s). Additionally, the UAV performs the missions in an autonomous mode, so it has no problem losing connectivity periodically. In the tests performed on the landing base, an ethernet cable was used. The landing base generated a hotspot to ensure connectivity for itself and the UAV because the landing and takeoff are the most crucial operation points.

Furthermore, when looking at the landing results, it was possible to conclude that the algorithm can land below a three cm error both on the simulated world and on real test setups. It was even possible to land without RTK GPS with a final position error smaller than three centimeters on the tests performed in the real environment. With the introduction of RTK GPS, it should be capable of reducing the absolute position error. Regarding the computer vision techniques, the mask detector was able to work in all

scenarios if the square is fully visible on the image frame. However, for other steps of the algorithm, there are points where there is room for improvement:

- **Pattern Detection**—In order to exclude false positives, the verification rules are too strict, rejecting many frames even when the detection is successful.
- **Aruco Detection**—The detection of the Aruco has as input a black and white image. This black and white image is the result of a threshold of the *Value* channel. Currently, the threshold value is predefined and works most of the time. A new way to define the threshold value automatically could be used.
- **Position Estimator**—Currently, the estimator was tuned using information from the simulation data and needs to be tuned using real data.
- **Control**—As described previously, the control maneuver, although effective, is not smooth and fast enough. In the final step, the UAV takes a significant time to finish the maneuver (i.e., eight seconds to descend 25 cm).

During the real tests, the effect of the GNSS position error, wind, and other irregularities exposed the need to improve the control loop of the UAV to smooth the UAV motion.

Commercial solutions, such as the DJI Mavic Pro Precision Landing, proved [89] to be inferior with a position error of 8.76 cm without heading control. On the other hand, Gonçalves et al. [90] tested autonomous landing control with vector fields, using nested fiducial markers on a computer vision system, achieving a horizontal landing error of 14 cm. Wang et al. [91] was capable of land on a stationary landing pad with a position error smaller than 30 cm. Patruno et al. [92] also present a cascading pattern detection algorithm, achieving a position error of 1.37 cm and 1.04 degrees. However, it does not adjust the UAV yaw to a specific orientation.

6. Conclusions

This work proposes an ecosystem that allows UAV autonomous missions without human intervention. To accomplish it, a set of three tools were developed and integrated, which were carefully examined. A critical point is the landing maneuver that has to position the UAV with centimeter-level accuracy for the charging operation to work. For this reason, a positioning system and a maneuver that achieve a common error positioning and lands the UAV in simulated and real environments with less than five centimeter error were developed, complying with the position error requirements.

In the simulated environment, the UAV was capable of landing, achieving the desired position error and detection rate of the visual marker elements. The tests performed in the real world suffered from the lack of detected frames; nevertheless, the UAV was able to land with a position error smaller than five centimeters even without the RTK position. During the real trials, the landing maneuver was tested under different lighting conditions to collect more data to test and make the algorithm work in more scenarios.

7. Future Work

The authors believe that some of the drawbacks perceived and reported in the conclusions can be addressed, and parts of the algorithms can be optimized, such as:

- Improve the threshold for the Aruco marker detection.
- Use a different computer vision position technique that can compute the roll and pitch of the UAV.
- Improve the maneuver control to make it smoother and faster, especially on the final step, when the visual pattern is detected.

Additionally, new features can be added to the ecosystem. For these reasons, this work will be continued with further updates on the components presented in this document. The list below summarizes the key innovative ideas that will drive our future work:

- HEIFU 5G connectivity chip on board.
- Adaption of the ecosystem for the medical area for critical transportation of assets between clinics and hospitals.

- Benchmark of performance updates with the integration of sonar to the landing routine.
- Remote visualization and configuration of the landing area sensors via the beXStream platform.
- Modification of the Landing Area to study other types of UAVs such as Hybrid VTOLs (with multirotor and fixed-wing).

Author Contributions: Conceptualization, M.M., F.A., D.P., Á.R. and J.M.-C.; methodology, M.M., F.A, A.F. and Á.R.; software, M.M.; validation, M.M., F.A., A.F., Á.R. and R.L.; formal analysis, M.M. and F.A.; investigation, M.M.; resources, D.P. and L.C.; data curation, M.M.; writing—original draft preparation, M.M., F.A., D.P. and J.M.-C.; writing—review and editing, M.M., F.A., A.F., D.P., J.M.-C., Á.R., R.L. and L.C.; visualization, M.M.; supervision, F.A., D.P. and Á.R.; project administration, L.C.; funding acquisition, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783221. This project has also received funding from the ECSEL JU under grant agreement No 783119. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Belgium, Czech Republic, Finland, Germany, Greece, Italy, Latvia, Norway, Poland, Portugal, Spain, Sweden. This project was also partially funded by AI4RealAg project. The goal of the AI4RealAg project is to develop an intelligent knowledge extraction system, based in Artificial Intelligence and Data Science, to increase sustainable agricultural production. The project is financed by Portugal 2020, under the Lisbon's Regional Operational Programme and the Competitiveness and Internationlization Operational Programme, worth "1,562,945.17" euros, from the European Regional Development Fund. This work was also partially funded by Fundação para a Ciência e a Tecnologia under Projects UIDB/04111/2020, foRESTER PCIF/SSI/0102/2017, and IF/00325/2015. This document has also received funding from the ECSEL Joint Undertaking (JU) under Grant agreement number: 101007273. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Netherlands, Germany, Spain, Denmark, Norway, Portugal, Belgium, Slovenia, Czech Republic, Turkey.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the article.

Acknowledgments: The authors would like to thank Beyond Vision Research and Development Group and PDMFC Research and Development Team.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
FOV	Field-Of-View
FPS	Frames Per Second
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
LED	Light Emitting Diode
LiDAR	Light Detection And Ranging
HEIFU	Hexa Exterior Intelligent Flying Unit
HSV	Hue Saturation Value
IMU	Inertial Measurement Unit
MPPT	Maximum Power Point Tracker
NAT	Network Address Translation
OBC	On Board Computer
QR	Quick Response

REST	Representational state transfer
RF	Radio Frequency
RGB	Red Green Blue
ROS	Robot Operating System
RTK	Real Time Kinematics
RTP	Real-time Transport Protocol
RTSP	Real-time Streaming Protocol
RTT	Round Trip Time
STUN	Session Traversal of UDP Through NAT
TURN	Traversal using Relays around NAT
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UHF	Ultra High Frequency
URLLC	Ultra Relieable Low Latency Communications
VTOL	Vertical Take-off and Landing
WebRTC	Real-Time Communication
WSS	WebSocket Secure

References

1. Ahirwar, S.; Swarnkar, R.; Bhukya, S.; Namwade, G. Application of Drone in Agriculture. *Int. J. Curr. Microbiol. Appl. Sci.* **2019**, *8*, 2500–2505, doi:10.20546/ijcmas.2019.801.264.
2. Radoglou Grammatikis, P.; Sarigiannidis, P.; Lagkas, T.; Moscholios, I. A Compilation of UAV Applications for Precision Agriculture. *Comput. Netw.* **2020**, *172*, 107148, doi:10.1016/j.comnet.2020.107148.
3. Li, X.; Savkin, A.V. Networked Unmanned Aerial Vehicles for Surveillance and Monitoring: A Survey. *Future Internet* **2021**, *13*, 174, doi:10.3390/fi13070174.
4. Shakhatreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634, doi:10.1109/ACCESS.2019.2909530.
5. Pedro, D.; Matos-Carvalho, J.P.; Azevedo, F.; Sacoto-Martins, R.; Bernardo, L.; Campos, L.; Fonseca, J.M.; Mora, A. FFAU—Framework for Fully Autonomous UAVs. *Remote Sens.* **2020**, *12*, 3533, doi:10.3390/rs12213533.
6. Azevedo, F.; Dias, A.; Almeida, J.; Oliveira, A.; Ferreira, A.; Santos, T.; Martins, A.; Silva, E. LiDAR-Based Real-Time Detection and Modeling of Power Lines for Unmanned Aerial Vehicles. *Sensors* **2019**, *19*, 1812, doi:10.3390/s19081812.
7. Pedro, D.; Matos-Carvalho, J.P.; Fonseca, J.M.; Mora, A. Collision Avoidance on Unmanned Aerial Vehicles Using Neural Network Pipelines and Flow Clustering Techniques. *Remote Sens.* **2021**, *13*, 2643.
8. van der Merwe, D.; Burchfield, D.R.; Witt, T.D.; Price, K.P.; Sharda, A. Drones in agriculture. In *Advances in Agronomy*; Academic Press: Cambridge, MA, USA, 2020; doi:10.1016/bs.agron.2020.03.001.
9. Bravo, G.C.; Parra, D.M.; Mendes, L.; De Jesus Pereira, A.M. First aid drone for outdoor sports activities. In Proceedings of the TISHW 2016—1st International Conference on Technology and Innovation in Sports, Health and Wellbeing, Vila Real, Portugal, 1–3 December 2016; doi:10.1109/TISHW.2016.7847781.
10. Sanjana, P.; Prathilothamai, M. Drone Design for First Aid Kit Delivery in Emergency Situation. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems, ICACCS 2020, Coimbatore, India, 6–7 March 2020, doi:10.1109/ICACCS48705.2020.9074487.
11. Sousa, P.; Ferreira, A.; Moreira, M.; Santos, T.; Martins, A.; Dias, A.; Almeida, J.; Silva, E. ISEP/INESC TEC Aerial Robotics Team for Search and Rescue Operations at the EuRathlon Challenge 2015. In Proceedings of the 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), Bragança, Portugal, 4–6 May 2016; pp. 156–161, doi:10.1109/ICARSC.2016.49.
12. Almeida, J.; Ferreira, A.; Matias, B.; Dias, A.; Martins, A.; Silva, F.; Oliveira, J.; Sousa, P.; Moreira, M.; Miranda, T.; et al. Air and underwater survey of water enclosed spaces for VAMOS! Project. In Proceedings of the OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, USA, 19–23 September 2016; pp. 1–5, doi:10.1109/OCEANS.2016.7761282.
13. Gallo, I.G.; Martínez-Corbella, M.; Sarro, R.; Iovine, G.; López-Vinielles, J.; Hérnandez, M.; Robustelli, G.; Mateos, R.M.; García-Davalillo, J.C. An Integration of UAV-Based Photogrammetry and 3D Modelling for Rockfall Hazard Assessment: The Cárcavos Case in 2018 (Spain). *Remote Sens.* **2021**, *13*, 3450, doi:10.3390/rs13173450.
14. Singh, A.K.; Swarup, A.; Agarwal, A.; Singh, D. Vision based rail track extraction and monitoring through drone imagery. *ICT Express* **2019**, *5*, 250–255, doi:10.1016/j.icte.2017.11.010.
15. Barreto, J.; Cajaíba, L.; Teixeira, J.B.; Nascimento, L.; Giacomo, A.; Barcelos, N.; Fettermann, T.; Martins, A. Drone-monitoring: Improving the detectability of threatened marine megafauna. *Drones* **2021**, *5*, 14, doi:10.3390/DRONES5010014.
16. Pedro, D.; Mora, A.; Carvalho, J.; Azevedo, F.; Fonseca, J. ColANet: A UAV Collision Avoidance Dataset. In *Technological Innovation for Life Improvement*; Camarinha-Matos, L.M., Farhadi, N., Lopes, F., Pereira, H., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 53–62.

17. Azevedo, F.; Cardoso, J.S.; Ferreira, A.; Fernandes, T.; Moreira, M.; Campos, L. Efficient Reactive Obstacle Avoidance Using Spirals for Escape. *Drones* **2021**, *5*, 51, doi:10.3390/drones5020051.
18. Allasia, P.; Baldo, M.; Giordan, D.; Godone, D.; Wrzesniak, A.; Lollino, G. Near Real Time Monitoring Systems and Periodic Surveys Using a Multi Sensors UAV: The Case of Ponzano Landslide. In Proceedings of the IAEG/AEG Annual Meeting Proceedings, San Francisco, CA, USA, 17–21 September 2018; Volume 1, doi:10.1007/978-3-319-93124-1_37.
19. Nenni, M.E.; Di Pasquale, V.; Miranda, S.; Riemma, S. Development of a Drone-Supported Emergency Medical Service. *Int. J. Technol.* **2020**, *11*, 656–666, doi:10.14716/ijtech.v11i4.3951.
20. Ballous, K.A.; Khalifa, A.N.; Abdulwadood, A.A.; Al-Shabi, M.; El Haj Assad, M. Medical kit: Emergency drone. In *Unmanned Systems Technology XXII*; International Society for Optics and Photonics: Bellingham, WA, USA, 2020; doi:10.1117/12.2566115.
21. Nafiz Hasan Khan, M.; Neustaedter, C. An exploratory study of the use of drones for assisting firefighters during emergency situations. In Proceedings of the Conference on Human Factors in Computing Systems, Glasgow, UK, 4–9 May 2019; doi:10.1145/3290605.3300502.
22. Aravindadhith, P.S. Design and Fabrication of Emergency Drone Recovery System. *Int. J. Eng. Adv. Technol.* **2020**, *9*, 424–427, doi:10.35940/ijeat.e9551.069520.
23. Dardoize, T.; Ciochetto, N.; Hong, J.; Shin, H. Implementation of Ground Control System for Autonomous Multi-agents using QGroundControl. In Proceedings of the 2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS), Cranfield, UK, 25–27 November 2019; pp. 24–30.
24. PX4. Available online: https://px4.io/ (accessed on 23 August 2021).
25. Ardupilot. Available online: https://ardupilot.org/ (accessed on 23 August 2021).
26. Pino, M.; Matos-Carvalho, J.P.; Pedro, D.; Campos, L.M.; Costa Seco, J. UAV Cloud Platform for Precision Farming. In Proceedings of the 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Porto, Portugal, 20–22 July 2020; pp. 1–6, doi:10.1109/CSNDSP49049.2020.9249551.
27. FlytOS. Available online: https://flytbase.com/flytos/ (accessed on 12 July 2021).
28. Yanmaz, E.; Yahyanejad, S.; Rinner, B.; Hellwagner, H.; Bettstetter, C. Drone networks: Communications, coordination, and sensing. *Ad. Hoc. Netw.* **2018**, *68*, 1–15, doi:10.1016/j.adhoc.2017.09.001.
29. Ollero, A.; Lacroix, S.; Merino, L.; Gancet, J.; Wiklund, J.; Remuss, V.; Perez, I.V.; Gutierrez, L.G.; Viegas, D.X.; Benitez, M.A.G.; et al. Multiple eyes in the skies: Architecture and perception issues in the COMETS unmanned air vehicles project. *IEEE Robot. Autom. Mag.* **2005**, *12*, 46–57.
30. Gupta, L.; Jain, R.; Vaszkun, G. Survey of Important Issues in UAV Communication Networks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1123–1152.
31. Ebeid, E.; Skriver, M.; Terkildsen, K.H.; Jensen, K.; Schultz, U.P. A survey of Open-Source UAV flight controllers and flight simulators. *Microprocess. Microsyst.* **2018**, *61*, 11–20, doi:10.1016/j.micpro.2018.05.002.
32. Matos-Carvalho, J.P.; Fonseca, J.M.; Mora, A. UAV Downwash Dynamic Texture Features for Terrain Classification on Autonomous Navigation. In Proceedings of the 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), Poznań, Poland, 9–12 September 2018; pp. 1079–1083.
33. Prates, P.A.; Mendonça, R.; Lourenço, A.; Marques, F.; Matos-Carvalho, J.P.; Barata, J. Vision-based UAV detection and tracking using motion signatures. In Proceedings of the 2018 IEEE Industrial Cyber-Physical Systems (ICPS), Saint Petersburg, Russia, 15–18 May 2018; pp. 482–487, doi:10.1109/ICPHYS.2018.8390752.
34. Azevedo, F.; Oliveira, A.; Dias, A.; Almeida, J.; Moreira, M.; Santos, T.; Ferreira, A.; Martins, A.; Silva, E. Collision avoidance for safe structure inspection with multirotor UAV. In Proceedings of the 2017 European Conference on Mobile Robots (ECMR), Paris, France, 6–8 September 2017; pp. 1–7, doi:10.1109/ECMR.2017.8098719.
35. Mourgelas, C.; Kokkinos, S.; Milidonis, A.; Voyiatzis, I. Autonomous Drone Charging Stations: A Survey. In Proceedings of the 24th Pan-Hellenic Conference on Informatics, Athens, Greece, 20–22 November 2020; pp. 233–236, doi:10.1145/3437120.3437314.
36. Galimov, M.; Fedorenko, R.; Klimchik, A. UAV Positioning Mechanisms in Landing Stations: Classification and Engineering Design Review. *Sensors* **2020**, *20*, 3648, doi:10.3390/s20133648.
37. Identified Technologies. *Autonomous Mapping Boomerang Drone and Docking Charging Station*; Identified Technologies: Pittsburgh, PA, USA, 2021.
38. Hextronics. *Hextronics Global*; Hextronics: Atlanta, GE, USA, 2021.
39. Skycharge. *Indoor Drone Charging Pad and Infrastructure*; Skycharge: Berlin, Germany, 2021.
40. Airscort Ltd. *ST-1200*; Airscort Ltd.: Jerusalem, Israel, 2021.
41. Noon. 21st Century (1221). UAV Docking Station; Noon. 21st Century (1221): Moscow, Russia. Available online: https://1221.su/uav-docking-station/ (accessed on 30 July 2021).
42. Morim, M.M. Sistema de Apoio ao Processo de Aterragem Autónoma de um VTOL. Master's Thesis, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Eletrotécnica, Porto, Portugal, 2017.
43. Wubben, J.; Fabra, F.; Calafate, C.T.; Krzeszowski, T.; Marquez-Barja, J.M.; Cano, J.C.; Manzoni, P. Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition. *Electronics* **2019**, *8*, 1532, doi:10.3390/electronics8121532.
44. He, K.; Weng, D.; Ji, S.; Wang, Z.; Chen, W.; Lu, Y.; Nie, Z. Improving the Performance of Time-Relative GNSS Precise Positioning in Remote Areas. *Sensors* **2021**, *21*, 292, doi:10.3390/s21010292.

45. Abbas, S.M.; Aslam, S.; Berns, K.; Muhammad, A. Analysis and Improvements in AprilTag Based State Estimation. *Sensors* **2019**, *19*, 5480, doi:10.3390/s19245480.
46. Xing, B.; Zhu, Q.; Pan, F.; Feng, X. Marker-Based Multi-Sensor Fusion Indoor Localization System for Micro Air Vehicles. *Sensors* **2018**, *18*, 1706, doi:10.3390/s18061706.
47. Yang, S.; Scherer, S.A.; Schauwecker, K.; Zell, A. Onboard monocular vision for landing of an MAV on a landing site specified by a single reference image. In Proceedings of the 2013 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GE, USA, 28–31 May 2013; pp. 318–325, doi:10.1109/ICUAS.2013.6564704.
48. Jung, Y.; Bang, H.; Lee, D. Robust marker tracking algorithm for precise UAV vision-based autonomous landing. In Proceedings of the 2015 15th International Conference on Control, Automation and Systems (ICCAS), Busan, Korea, 13–16 October 2015; pp. 443–446, doi:10.1109/ICCAS.2015.7364957.
49. Demirhan, M.; Premachandra, C. Development of an Automated Camera-Based Drone Landing System. *IEEE Access* **2020**, *8*, 202111–202121, doi:10.1109/ACCESS.2020.3034948.
50. Cocchioni, F.; Mancini, A.; Longhi, S. Autonomous navigation, landing and recharge of a quadrotor using artificial vision. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; pp. 418–429, doi:10.1109/ICUAS.2014.6842282.
51. Hui, C.; Yousheng, C.; Xiaokun, L.; Shing, W.W. Autonomous takeoff, tracking and landing of a UAV on a moving UGV using onboard monocular vision. In Proceedings of the 32nd Chinese Control Conference, Xi'an, China, 26–28 July 2013; pp. 5895–5901.
52. Fitzgibbon, A.; Pilu, M.; Fisher, R.B. Direct least square fitting of ellipses. *IEEE Trans. Pattern Anal. Mach. Intell.* **1999**, *21*, 476–480, doi:10.1109/34.765658.
53. Huang, C.M.; Hung, T.S. Visual servoing of micro aerial vehicle landing on ground platform. In Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Diego, CA, USA, 5–8 October 2014; pp. 2071–2076, doi:10.1109/SMC.2014.6974227.
54. Koo, T.J.; Sastry, S. Hybrid control of unmanned aerial vehicles for autonomous landing. In Proceedings of the 2nd AIAA "Unmanned Unlimited" Conference and Workshop and Exhibit, San Diego, CA, USA, 15–18 September 2003; doi:10.2514/6.2003-6541.
55. Cocchioni, F.; Frontoni, E.; Ippoliti, G.; Longhi, S.; Mancini, A.; Zingaretti, P. Visual Based Landing for an Unmanned Quadrotor. *J. Intell. Robot. Syst. Theory Appl.* **2016**, *84*, 511–528, doi:10.1007/s10846-015-0271-6.
56. Censi, A.; Strubel, J.; Brandli, C.; Delbruck, T.; Scaramuzza, D. Low-latency localization by active LED markers tracking using a dynamic vision sensor. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 891–898, doi:10.1109/IROS.2013.6696456.
57. Okano, Y.; Ito, Y. LED marker position detection using Walsh Functions. In Proceedings of the 2010 International Symposium on Intelligent Signal Processing and Communication Systems, Naha, Japan, 6–8 December 2010; pp. 1–4, doi:10.1109/ISPACS.2010.5704652.
58. Breitenmoser, A.; Kneip, L.; Siegwart, R. A monocular vision-based system for 6D relative robot localization. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 79–85, doi:10.1109/IROS.2011.6094851.
59. Wang, X.-H.; Xu, G.-L.; Tian, Y.-P.; Wang, B.; Wang, J.-D. UAV's Automatic Landing in All Weather Based on the Cooperative Object and Computer Vision. In Proceedings of the 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control, Harbin, China, 8–10 December 2012; pp. 1346–1351, doi:10.1109/IMCCC.2012.317.
60. Wenzel, K.E.; Masselli, A.; Zell, A. Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle. *J. Intell. Robot. Syst.* **2011**, *61*, 221–238, doi:10.1007/s10846-010-9473-0.
61. Khithov, V.; Petrov, A.; Tishchenko, I.; Yakovlev, K. Toward autonomous UAV landing based on infrared beacons and particle filtering. In *Robot Intelligence Technology and Applications*; Springer: Cham, Switzerland, 2017; doi:10.1007/978-3-319-31293-4_43.
62. Kim, D.; Choi, J. Multi-robot team outdoor localization using active marker and high frequency signal sources. In Proceedings of the 2011 11th International Conference on Control, Automation and Systems, Gyeonggi-do, Korea, 26–29 October 2011; pp. 192–196.
63. Xu, Z.C.; Hu, B.B.; Liu, B.; Wang, X.D.; Zhang, H.T. Vision-based Autonomous Landing of Unmanned Aerial Vehicle on a Motional Unmanned Surface Vessel. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 6845–6850, doi:10.23919/CCC50068.2020.9188979.
64. Fu, M.; Zhang, K.; Yi, Y.; Shi, C. Autonomous landing of a quadrotor on an UGV. In Proceedings of the 2016 IEEE International Conference on Mechatronics and Automation, Harbin, China, 7–10 August 2016; pp. 988–993, doi:10.1109/ICMA.2016.7558697.
65. Grobler, P.R.; Jordaan, H.W. Autonomous Vision Based Landing Strategy for a Rotary Wing UAV. In Proceedings of the 2020 International SAUPEC/RobMech/PRASA Conference, Cape Town, South Africa, 29–31 January 2020; pp. 1–6, doi:10.1109/SAUPEC/RobMech/PRASA48453.2020.9041238.
66. Aalerud, A.; Dybedal, J.; Hovland, G. Automatic Calibration of an Industrial RGB-D Camera Network Using Retroreflective Fiducial Markers. *Sensors* **2019**, *19*, 1561, doi:10.3390/s19071561.
67. Zhang, Y.; Yu, Y.; Jia, S.; Wang, X. Autonomous landing on ground target of UAV by using image-based visual servo control. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 11204–11209, doi:10.23919/ChiCC.2017.8029145.
68. influxDB. 2021. Available online: https://www.influxdata.com/ (accessed on 30 July 2021).

69. beXStream. 2021. Available online: https://x-stream.github.io/ (accessed on 30 July 2021).
70. Amirante, A.; Castaldi, T.; Miniero, L.; Romano, S.P. Janus: A general purpose WebRTC gateway. In Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications, IPTComm 2014, Chicago, IL, USA, 1–2 October 2014; doi:10.1145/2670386.2670389.
71. Sacoto-Martins, R.; Madeira, J.; Matos-Carvalho, J.P.; Azevedo, F.; Campos, L.M. Multi-purpose Low Latency Streaming Using Unmanned Aerial Vehicles. In Proceedings of the 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Porto, Portugal, 20–22 July 2020; pp. 1–6, doi:10.1109/CSNDSP49049.2020.9249562.
72. Hanhart, P.; He, Y.; Ye, Y.; Boyce, J.; Deng, Z.; Xu, L. 360-Degree Video Quality Evaluation. In Proceedings of the 2018 Picture Coding Symposium, PCS 2018, San Francisco, CA, USA, 24–27 June 2018; doi:10.1109/PCS.2018.8456255.
73. Popovski, P.; Stefanovic, C.; Nielsen, J.J.; de Carvalho, E.; Angjelichinoski, M.; Trillingsgaard, K.F.; Bana, A.S. Wireless Access in Ultra-Reliable Low-Latency Communication (URLLC). *IEEE Trans. Commun.* **2019**, *67*, 5783–5801, doi:10.1109/tcomm.2019.2914652.
74. Janczukowicz, E.; Braud, A.; Tuffin, S.; Fromentoux, G.; Bouabdallah, A.; Bonnin, J.M. Specialized network services for WebRTC: TURN-based architecture proposal. In Proceedings of the 1st Workshop on All-Web Real-Time Systems, AweS 2015—In Conjunction with EuroSys 2015, Bordeaux, France, 21 April 2015; doi:10.1145/2749215.2749218.
75. U-blox. *ZED-F9P Module u-blox F9 High Precision GNSS Module*; U-blox: Thalwil, Switzerland, 2019.
76. Aeriya. *beRTK*; Aeriya: Alfragide, Portugal, 2021.
77. Pathak, S.V.; Mohod, A.G.; Sawant, A.A. Review on effective role of UAV in precision farming. *J. Pharmacogn. Phytochem.* **2020**, *9*, 463–467.
78. Romero-Ramirez, F.; Muñoz-Salinas, R.; Medina-Carnicer, R. Speeded Up Detection of Squared Fiducial Markers. *Image Vis. Comput.* **2018**, *76*, 38–47, doi:10.1016/j.imavis.2018.05.004.
79. Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.; Medina-Carnicer, R. Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. *Pattern Recognit.* **2015**, *51*, 481–491, doi:10.1016/j.patcog.2015.09.023.
80. Carvalho, J.P.L.A.d.M. Improved Terrain Type Classification Using UAV Downwash Dynamic Texture Effect. Ph.D. Thesis, Faculdade de Ciências e Tecnologia (FCT), Universidade Lusófona, Lisbon, Portugal, 2021.
81. Krig, S. *Computer Vision Metrics*; Springer: Berlin/Heidelberg, Germany, 2016; doi:10.1007/978-3-319-33762-3.
82. OpenCV. 2021. Available online: https://www.uco.es/investiga/grupos/ava/node/26 (accessed on 30 July 2021).
83. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137, doi:10.1109/TIT.1982.1056489.
84. Matos-Carvalho, J.P.; Santos, R.; Tomic, S.; Beko, M. GTRS-Based Algorithm for UAV Navigation in Indoor Environments Employing Range Measurements and Odometry. *IEEE Access* **2021**, *9*, 89120–89132, doi:10.1109/ACCESS.2021.3089900.
85. Nakama, J.; Parada, R.; Matos-Carvalho, J.P.; Azevedo, F.; Pedro, D.; Campos, L. Autonomous Environment Generator for UAV-Based Simulation. *Appl. Sci.* **2021**, *11*, 2185, doi:10.3390/app11052185.
86. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154, doi:10.1109/IROS.2004.1389727.
87. Matos-Carvalho, J.P.; Pedro, D.; Campos, L.M.; Fonseca, J.M.; Mora, A. Terrain Classification Using W-K Filter and 3D Navigation with Static Collision Avoidance. In *Intelligent Systems and Applications*; Bi, Y., Bhatia, R., Kapoor, S., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 1122–1137.
88. Kam, H.; Lee, S.H.; Park, T.; Kim, C.H. RViz: A toolkit for real domain data visualization. *Telecommun. Syst.* **2015**, *60*, 1–9, doi:10.1007/s11235-015-0034-5.
89. Yoakum, C.; Cerreta, J. A Review of DJI's Mavic Pro Precision Landing Accuracy. *Int. J. Aviat. Aeronaut. Aerosp.* **2020**, *7*, 1–19, doi:10.15394/IJAAA.2020.1524.
90. Goncalves, V.M.; Mclaughlin, R.; Pereira, G.A. Precise Landing of Autonomous Aerial Vehicles Using Vector Fields. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4337–4344, doi:10.1109/LRA.2020.2994485.
91. Wang, J.; McKiver, D.; Pandit, S.; Abdelzaher, A.F.; Washington, J.; Chen, W. Precision UAV Landing Control Based on Visual Detection. In Proceedings of the 3rd International Conference on Multimedia Information Processing and Retrieval, MIPR 2020, San Jose, CA, USA, 6–8 August 2020; pp. 205–208, doi:10.1109/MIPR49039.2020.00049.
92. Patruno, C.; Nitti, M.; Petitti, A.; Stella, E.; D'Orazio, T. A Vision-Based Approach for Unmanned Aerial Vehicle Landing. *J. Intell. Robot. Syst. Theory Appl.* **2019**, *95*, 645–664, doi:10.1007/s10846-018-0933-2.