ORCA – Online Research @ Cardiff

Monocular visual autonomous landing system for quadcopter drones using software in the loop

Miguel Saavedra-Ruiz[*], Ana Maria Pinto-Vargas[†] ,

Victor Romero-Cano, *Member, IEEE* [‡]

[*]Mila - Quebec Institute of Artificial Intelligence, Université de Montréal, Canada

miguel.angel.saavedra.ruiz@umontreal.ca

[†] Alternova Tech SAS, Medellín, Colombia

ana.pinto@alternova.io

[‡]Robotics and Autonomous Systems Laboratory, Faculty of Engineering,

Universidad Autónoma de Occidente, Cali, Colombia

varomero@uao.edu.co

Abstract

Autonomous landing is a capability that is essential to achieve the full potential of multi-rotor drones in many social and industrial applications. The implementation and testing of this capability on physical platforms is risky and resource-intensive; hence, in order to ensure both a sound design process and a safe deployment, simulations are required before implementing a physical prototype. This paper presents the development of a monocular visual system, using a software-in-the-loop methodology, that autonomously and efficiently lands a quadcopter drone on a predefined landing pad, thus reducing the risks of the physical testing stage. In addition to ensuring that the autonomous landing system as a whole fulfils the design requirements using a Gazebo-based simulation, our approach provides a tool for safe parameter tuning and design testing prior to physical implementation. Finally, the proposed monocular vision-only approach to landing pad tracking made it possible to effectively implement the

system in an F450 quadcopter drone with the standard computational capabilities of an Odroid XU4 embedded processor.

Index Terms

Autonomous landing, quadcopter, target tracking, software-in-the-loop, simulation, Sim2Real.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have recently become popular due to their potential in terms of performing complex tasks such as infrastructure inspection [1], target detection [2], [3], or search and rescue [4]. The use of these gadgets has led to both substantial improvements in the efficiency of these processes and a reduction in human casualties while performing hazardous labours. The deployment of UAVs in such applications requires a complete suite of sensors such as GPS, laser rangefinders, radar and cameras [5], which can be used to endow the vehicle with environmental awareness and the capability to perceive events of interest. However, the use of many peripherals in a UAV requires an extensive amount of on-board computational resources and power that are not always available owing to the vehicle's dimensions and the high implementation costs.
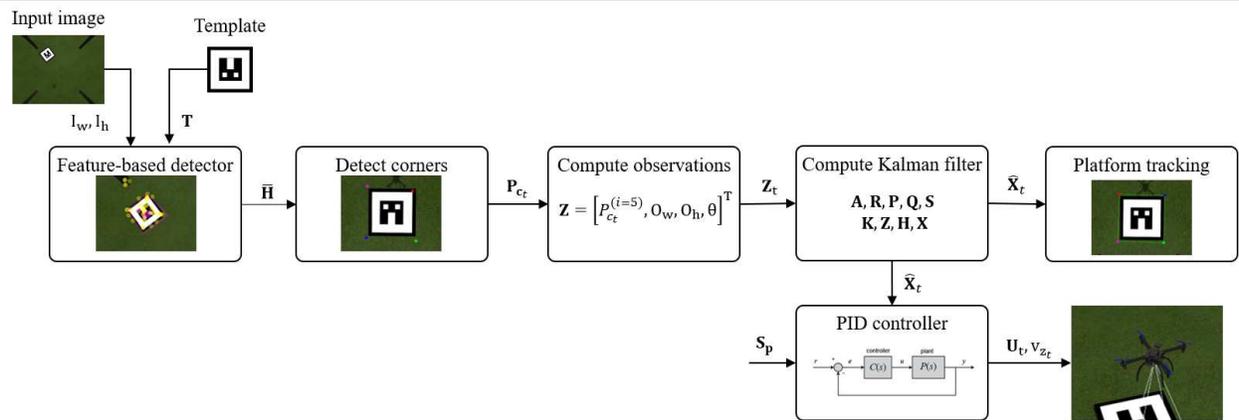
Fig. 1: Autonomous landing system pipeline with image-based visual servoing.

Cameras have been proposed as a feasible alternative to overcome these issues, as they have a relative low price and enable the estimation of content-rich representations of the environment. For instance, cameras have been widely employed in various tasks such as mapping [6] and object tracking [7]. Further research efforts have been conducted to make use of cameras in

the development of visual-based autonomous landing systems for UAVs. Autonomous landing maneuvers remain a crucial task for rotorcraft, and allow the development of complete, end-to-end autonomous flying vehicles that are capable of performing complex assignments such as those mentioned above.

Most state-of-the-art visual-based landing systems have shown unprecedented results that are comparable to the performance of UAVs with a full suite of sensors [10]. The employment of natural landmarks to land a rotorcraft in unstructured environments is a strategy commonly used for emergency landing situation [6], [11]. These methods rely on the use of vision-based SLAM algorithms such as ORB_SLAM2 [12] for localization of the vehicle and mapping of appropriate landing spots in the environment. Nevertheless, these techniques are prone to deliver low spatial resolution and be computationally expensive, hindering the performance of autonomous landing applications.

On the other hand, the utilization of artificial landmarks is one of the most traditional techniques used in landings on both static [13] and moving platforms [14]. The extraction of information from a landmark, such as the relative pose or template coordinates, is broadly used in the application of image-based visual servoing (IBVS), a technique that performs the majority of the control calculations in 2D image space [15] and reduces the computational load of a small rotorcraft while landing.

Visual servoing is commonly used with classical computer vision methods to track features over several image frames and to create stable control references to land the aircraft on a desired target. Despite the potential of IBVS in terms of autonomous landing, additional assumptions are required, for example that the features in the image are static features of the object, or that the object does not leave the field of view [7]. Furthermore, the implementation of visual-based autonomous landing systems requires rigorous assessments in simulated environments to identify possible perils before deploying the whole system in the real world.

Lately, deep learning has been proposed as an alternative to replace feature-based methods with Convolutional Neural Networks (CNNs) for landmark detection [8]. The use of CNN has exhibited robustness with diverse lighting conditions, scale variations and rotations. Notwithstanding the potential of deep learning-based object detectors, these models typically require extensive amount of human-labeled datasets and vast computational resources that are usually available only with off-board computing strategies [9].

In this work, we address these problems by proposing a complete monocular visual-based

perception and control strategy for the autonomous landing of a UAV in a Gazebo-based simulated environment. This system aims to mitigate the current limitations on classic computer vision-based methods created by changes in the appearance in the image by using a Kalman filter to estimate the position of the template throughout the landing process. Additionally, the use of only IBVS techniques for the control of the aircraft reduces the computational cost of the system and eliminates the need for expensive 3D position reconstruction calculations, thus allowing for real-time control of small UAVs with low-cost computers.

Fig. 1 illustrates the general workflow of the proposed method. Initially, the system computes the homography matrix between the current image frame and the predefined template, using a feature-based detector. Next, the homography matrix is used to compute the corners and the centroid of the object in the current image frame. These points are then passed to a Kalman filter estimation module. Finally, the Kalman filter estimations are used to track the template in the image frame, and as a process variable for a set of three PID-based controllers that perform the safe landing of the vehicle.

The full system was developed and assessed in a Gazebo-based simulated environment in order to bridge the gap between real-world deployment and theory, and to reduce the number of risks while the vehicle is tested. All the parameters for the vision and control systems employed in the Gazebo-based simulation were directly transferred to the real-world quad-rotor in a zero-shot[†] sim2real (simulation to reality) fashion in order to validate that these simple approaches can be effectively transferred to the vehicle without additional tuning [16]. Overall, the principal contributions of this work can be summarized as follows:

1) A complete, flexible, Gazebo-based simulation of a visual-based landing system for low-cost UAVs;

2) The implementation of a Kalman-filter-based methodology for landing platform tracking using monocular vision in both a simulated and a physical drone;

3) A control strategy for quadcopter landing that is seamlessly implemented using the popular PX4 software-in-the-loop (SITL) Gazebo interface, which facilitates its transfer to a physical drone.

This paper consists of five sections, as follows: Section II presents related work. In Section III, the feature-based detector and Kalman filter are explained. Section IV describes the proposed

---

[†]It refers to when parameters are learned or set in a source domain (simulation) and tested without *fine-tuning* in a target domain (real-world).

control strategy for the landing maneuver, while Sections V and VI contain the simulated and experimental results, respectively. Finally, Section VII presents the conclusion.

II. Related Work

Autonomous landing for multi-rotor aircraft is a problem that has been extensively studied. Various approaches have relied on the use of vision-based techniques to identify the salient features in an image and to land the vehicle on both static [18]–[20] and moving platforms [10], [14], [15]. Classic computer vision methods, such as feature-based extraction and description or homography-based approaches [21], [22], are commonly used to estimate the relative pose of the vehicle with respect to a landing platform at a relative low computational cost.

Spatial information can be extracted from natural and artificial landmarks. In [6], the authors proposed the use of natural landmarks for the detection and reconstruction of landing sites based on the texture of the ground. Visual-based SLAM techniques are also exploited to assemble world's representations and find feasible landing spots for the rotor-craft in unstructured environments as shown in [11]. Similarly, the use of artificial landmarks can alleviate the autonomous landing task by providing references with known dimensions for detection and tracking over several image frames [23].

The use of markers has been exploited to provide a traceable reference for landing control systems and to enhance the position estimation of aerial vehicles through visual inertial odometry (VIO). In [7], the authors estimated the relative pose of the aircraft with respect to a spherical target and used an extended Kalman filter to fuse these measurements with IMU data to accurately locate the vehicle within the space.

Kalman filters are not exclusively employed to fuse information from multiple sensor sources but also to estimate the states of a system from a unique noisy source [24], [25]. These estimations are used in IBVS, with linear control strategies such as nested PIDs [6], [18] and nonlinear ones like sliding mode controllers [15], to accurately land an aerial vehicle. The utilization of Gaussian estimators for IBVS provides numerically stable and continuous references for controllers, even when the object of interest is outside the field of view of the camera.

Further research efforts have concentrated on the use of deep learning methods to detect and track landmarks in images using CNN-based architectures [9], [26], [27] or to automate the complete landing task with deep reinforcement learning (DRL) agents [28]. However, the use of artificial neural networks requires substantial computational resources for real-time inference

and thousand of human-labeled images based on the task at hand [9]. Likewise, visual-based 3D reconstruction techniques tend to be computationally expensive for on-board computers in small UAVs [11], and need to satisfy various assumptions to achieve accurate pose estimations.

We aim to reduce the computational load when performing IBVS with the use of a vision-based tracking system, and to produce a stable reference for a set of nested PID-based controllers similar to those in [6]. The idea behind the detection and tracking system is to produce a 2D image-based reference for the controller, thus avoiding expensive 3D pose reconstructions as in [15]. In this work, the use of the Kalman filter is restricted to filtering 2D estimations of the landing pad from noisy observations, unlike the application of VIO in most other related work. Contrary to commonly used simulation tools like *RotorS* [17], which provide Gazebo-based simulation environments for multi-rotor drones with no interface with a real flight controller, our implementation utilises the SITL provided by PX4, which runs the Pixhawk flight stack, and therefore provides direct support to the physical robot deployment process.

III. VISION SYSTEM

This section describes our detection and tracking system for the autonomous landing of a UAV, which is an extension of our previous work in [21], [33]. We first explain how the feature-based object detector detects the landing platform when comparing the platform's template with the input image. Next, we describe how the system translates the corners and the centroid of the detected platform from the homography matrix to a vector that contains the system observations. Finally, we explain how a tailored Kalman filter is used to estimate the pose of the landing platform, even when no detection has been obtained.

A. Feature-based object detection

Object detection is a crucial task in robotic perception. Feature-based detectors and descriptors are widely used, due to their speed in computing the salient features of images. For increased robustness in object detection, these features should be invariant to rotation, scale and affine transformations over several frames [29]. To find correspondences between two images, we consider a set of features in the template image $\boldsymbol{F_T} \in \mathbb{R}^n$ and the current frame $\boldsymbol{F_S} \in \mathbb{R}^m$, where $n, m \in \mathbb{Z}$ represent the number of features in each image. Each feature in the template and scene frames is associated with a descriptor $\boldsymbol{D_T} \in \mathbb{R}^{n \times k}, \boldsymbol{D_S} \in \mathbb{R}^{m \times k}$, where k is the dimension of the descriptor for each feature.
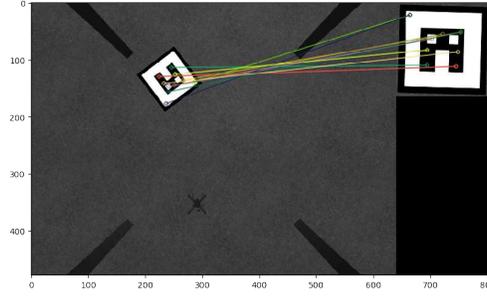
Fig. 2: Feature matching between the features of the template $\boldsymbol{F_T}$ and the features of the scene image $\boldsymbol{F_S}$

With a set of descriptors, it is possible to compute matches between image pairs by performing distance calculations, such as the Euclidean distance between the descriptors of the template and the scene, as shown in (1). Two features are matched when the closest descriptors between two images in the descriptor space have been found. As a result, similar features in the template image (x_i, y_i) are matched with the similar pair (x_i', y_i') in the current image frame, as illustrated in Fig. 2.

$$\boldsymbol{d_i} = min(\sum_{i=1}^{n}\sum_{j=1}^{m}\sqrt{(\boldsymbol{D_T}^{(i,k)})^2 - (\boldsymbol{D_S}^{(j,k)})^2}) \tag{1}$$

1) Homography Matrix: Finding correspondences between image pairs allows us to compute the homography matrix $\overline{\boldsymbol{H}} \in \mathbb{R}^{3 \times 3}$. This matrix is a transform that maps points from one image frame (template) to the corresponding points in the other image frame (scene). To compute the homography, at least four matches are needed. Then, knowing the homography between two images and the dimensions of the template $\boldsymbol{T} = [w_T, h_T]^T$, it is possible to apply a perspective transform that maps the template position from the template image to the scene image using (2).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \overline{\boldsymbol{H}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

In (2), $[x, y, 1]^T$ are the coordinates of points (e.g. corners) in the template image and $[x', y', 1]^T$ are the same points mapped in the scene image, where $\overline{\boldsymbol{H}} : \mathbb{R}^3 \to \mathbb{R}^3$. Object detection with feature-based methods and homography calculations tends to speed up the process and can provide a reliable estimation of the location of the object of interest in the current image frame.

B. System observations

Using the homography matrix, the corners and centroid of the template detected in the current image frame can be computed. $\boldsymbol{P_{ct}} \in \mathbb{R}^{5 \times 2}$ is defined as a vector of coordinates, where each row corresponds to a x, y point at time index t.

These points are used to determine the observations that will be fed into the Kalman filter. The vector of observations at time t is defined as $\boldsymbol{Z}_t = [\boldsymbol{P_{ct}}^{(i=5)}, O_w, O_h, \theta]$, where $\boldsymbol{P_{ct}}^{(i=5)}$ are the x, y centroid coordinates of the landing pad; O_w, O_h are the width and height of the template, respectively; and θ represents the angle of the template with respect to the x axis of the image, as shown in Fig. 3.
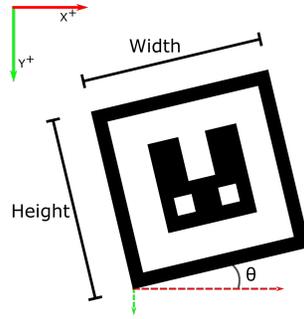
Fig. 3: System's observations and coordinate system. The vector $\boldsymbol{Z}_t$ is created at each time-step based on the width and height (O_w, O_h), the orientation θ and the position of the centroid (x, y) of the template in the current image frame.

C. Kalman filter

The Kalman filter is an estimator that infers hidden states from indirect, inaccurate and uncertain observations. It is possible to use the Filter to handle noisy observations from the detection module and produce a continuous estimate of the template position at each time step t [6].

We assume that we have a Linear Dynamic System (LDS) for a landing platform such as in (3), where x_t is the x coordinate of a pixel at time index t and Δt is the time between two consecutive image frames. Similarly, in (4), $\dot{x}_t$ corresponds to the x velocity component of a pixel in the image.

$$x_t = x_{t-1} + \Delta t \dot{x}_{t-1} \tag{3}$$

$$\dot{x}_t = \dot{x}_{t-1} \tag{4}$$

The set of states $\boldsymbol{X} \in \mathbb{R}^{10}$ is given by (5), with x_c, y_c as the position of the centroid of the template in the image frame. The filter states are the same as the vector of observations plus their first-order derivatives.

$$\boldsymbol{X} = [x_c, y_c, O_w, O_h, \theta, \dot{x}_c, \dot{y}_c, \dot{O}_w, \dot{O}_h, \dot{\theta}]^T \tag{5}$$

Knowing the transition dynamics and states of the filter, the motion model of the system is then given by (6). The matrix $\boldsymbol{A} \in \mathbb{R}^{10 \times 10}$, shown in (7), is the state transition matrix of the system and $\boldsymbol{w}$ is a white noise random vector such that $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{Q})$. $\boldsymbol{Q} \in \mathbb{R}^{10 \times 10}$ is defined as the covariance matrix of the process noise [31]. For the sake of notation, $\mathbf{I}$ represents the identity matrix.

$$\boldsymbol{X}_t = \boldsymbol{A}_{t-1}\boldsymbol{X}_{t-1} + \boldsymbol{w}_{t-1} \tag{6}$$

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{I}_{5 \times 5} & \Delta t \mathbf{I}_{5 \times 5} \\ \mathbf{0}_{5 \times 5} & \mathbf{I}_{5 \times 5} \end{pmatrix}_{10 \times 10} \tag{7}$$

Likewise, the measurement model of the filter is given in (8). $\boldsymbol{H} = \mathbf{I}_{5 \times 10}$ is defined as the observation matrix and $\boldsymbol{v}$ is a white noise random vector such that $\boldsymbol{v} \sim \mathcal{N}(0, \boldsymbol{R})$. As for $\boldsymbol{Q}$, here $\boldsymbol{R} \in \mathbb{R}^{5 \times 5}$ is the covariance matrix of the observation noise.

$$\boldsymbol{Y}_t = \boldsymbol{H}_t \boldsymbol{X}_t + \boldsymbol{v}_t \tag{8}$$

With the motion and measurement models defined, it is possible to formulate the pose estimation process of the platform by giving the Kalman filter equations (9)-(15). In this set of equations, $\boldsymbol{P}$ is defined as the covariance matrix of the posterior estimate, $\boldsymbol{Y}$ is the innovation vector, $\boldsymbol{K}$ is the Kalman gain and $\boldsymbol{S}$ is the covariance matrix of the innovation. Additionally, $\check{\boldsymbol{X}}$ represents the predicted states and $\hat{\boldsymbol{X}}$ the corrected states after a measurement update.

The first two equations (9)-(10) are used in the prediction step, and give an estimate of the states $\boldsymbol{X}$ at each time step regardless of whether an observation was obtained.

$$\check{\boldsymbol{X}}_t = \boldsymbol{A}_t \boldsymbol{X}_{t-1} \tag{9}$$

$$\boldsymbol{P}_t = \boldsymbol{A}_t \boldsymbol{P}_{t-1} \boldsymbol{A}_t^T + \boldsymbol{Q}_t \tag{10}$$

When an observation is obtained by the detection module, the correction phase (11)-(15) is computed immediately after the prediction step. This step aims to correct the error in the estimations using an observation of the template in the current image frame at time t.

$$\boldsymbol{Y}_t = \boldsymbol{Z}_t - \boldsymbol{H}_t \check{\boldsymbol{X}}_t \tag{11}$$

$$\boldsymbol{S}_t = \boldsymbol{H}_t \boldsymbol{P}_t \boldsymbol{H}_t^T + \boldsymbol{R}_t \tag{12}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_t \boldsymbol{H}_t^T \boldsymbol{S}_t^{-1} \tag{13}$$

$$\hat{\boldsymbol{X}}_t = \check{\boldsymbol{X}}_t + \boldsymbol{K}_t \boldsymbol{Y}_t \tag{14}$$

$$\boldsymbol{P}_t = (\mathbf{I}_{10x10} - \boldsymbol{K}_t \boldsymbol{H}_t) \boldsymbol{P}_t \tag{15}$$

The set of states produced by the Kalman filter can be applied in an IBVS module to control the landing procedure and to obtain the position of the landing platform in the current image frame, as shown in Fig. 4. These states are computed in the 2D image frame to reduce the computation carried out by the on-board computer of the UAV. Furthermore, since we are not estimating the relative pose between the vehicle and the landing platform, we are not feeding IMU data to the tracking module; this allows for the use of a linear Kalman filter and avoids the calculation of Jacobians at each time-step. Pseudo-code for the vision-based detection and tracking system is given in Algorithm 1.

Algorithm 1 Landing platform detector and tracker

1: **Inputs:**
   $\overline{\boldsymbol{H}} \in \mathbb{R}^{3\times3}$ Homography matrix
   $\boldsymbol{T} \in \mathbb{R}^2$ Dimensions of the template
2: **Initialization of the Kalman Filter:**
   $\boldsymbol{A} \in \mathbb{R}^{10\times10}, \quad \boldsymbol{R} \in \mathbb{R}^{5\times5}, \quad \boldsymbol{P} \in \mathbb{R}^{10\times10}, \quad \boldsymbol{Q} \in \mathbb{R}^{10\times10}, \quad \boldsymbol{K} \in \mathbb{R}^{5\times5},$
   $\boldsymbol{S} \in \mathbb{R}^{10\times10}, \quad \boldsymbol{H} \in \mathbb{R}^{5\times10}, \quad \boldsymbol{Y} \in \mathbb{R}^5, \quad \boldsymbol{X} \in \mathbb{R}^{10}$
3: **Outputs:**
   $\boldsymbol{X} \in \mathbb{R}^{10}$ State vector
4: **Variables:**
   $\boldsymbol{P_c} \in \mathbb{R}^5$ Corners and centroid of the template
   $\boldsymbol{Z} \in \mathbb{R}^5$ Observations of the template
   O_w Width of the template in the image frame
   O_h Height of the template in the image frame
   θ Estimated angle w.r.t X axis
5: **for each** frame **do**
6:     **for** $1,\dots,5$ **do**
7:        $\boldsymbol{P_{ct}} \leftarrow \text{computePoint}(\overline{\boldsymbol{H}}, \boldsymbol{T}, point = i)$
8:     **end for**
9:     $\boldsymbol{P_{ct}} \leftarrow \text{sortCorners}(\boldsymbol{P_{ct}})$
10:     $O_w, O_h \leftarrow \text{computeObjectDims}(\boldsymbol{P_{ct}})$
11:     $\theta \leftarrow \text{computeAngle}(\boldsymbol{P_{ct}})$
12:     **if** $\theta > 90 \deg$ **then**
13:        $\theta_{new} \leftarrow \theta/90$
14:        $\theta \leftarrow \theta - \theta_{new} \times 90$
15:     **end if**
16:     $\boldsymbol{Z} \leftarrow \{\boldsymbol{P_{ct}}^{(i=5)}, O_w, O_h, \theta\}$
17:     // KF Prediction step
18:     $\check{\boldsymbol{X}}_t \leftarrow \boldsymbol{A}_t \times \boldsymbol{X}_{t-1}$
19:     $\boldsymbol{P}_t \leftarrow \boldsymbol{A}_t \times \boldsymbol{P}_{t-1} \times \boldsymbol{A}_t^T + \boldsymbol{Q}_t$
20:     **if** detection is valid **then**
21:        // KF Correction step
22:        $\boldsymbol{Y}_t \leftarrow \boldsymbol{Z}_t - \boldsymbol{H}_t \times \check{\boldsymbol{X}}_t$
23:        $\boldsymbol{S}_t \leftarrow \boldsymbol{H}_t \times \boldsymbol{P}_t \times \boldsymbol{H}_t^T + \boldsymbol{R}_t$
24:        $\boldsymbol{K}_t \leftarrow \boldsymbol{P}_t \times \boldsymbol{H}_t^T \times \boldsymbol{S}_t^{-1}$
25:        $\hat{\boldsymbol{X}}_t \leftarrow \check{\boldsymbol{X}}_t + \boldsymbol{K}_t \times \boldsymbol{Y}_t$
26:        $\boldsymbol{P}_t \leftarrow (\boldsymbol{I}_{10x10} - \boldsymbol{K}_t \times \boldsymbol{H}_t) \times \boldsymbol{P}_t$
27:     **end if**
28:     $\boldsymbol{X}_t \leftarrow \hat{\boldsymbol{X}}_t$
29:     **return** $\boldsymbol{X}_t$
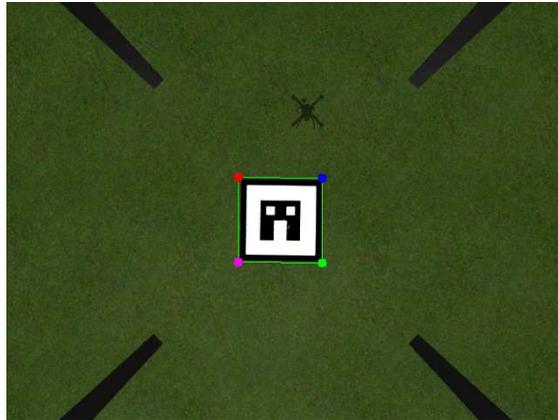30: **end for**

Fig. 4: Estimation of the platform position using the Kalman Filter vector of states X

IV. CONTROL SYSTEM

This section describes the PID-based controller used to autonomously land our rotorcraft. The IBVS controller uses the 2D output of the estimation module as a reference to compute position-velocity control signals to land the vehicle. These signals are sent to the native position-velocity loops implemented in the PX4 flight-stack, which transforms the positions into speeds and then converts them into thrust commands for the vehicle's engines, to guarantee correct control of the aircraft.

A. PID-based controller

In order to ensure that the aircraft moves towards the landing platform and lands on it, a control strategy is required. Autonomous landing of the vehicle is accomplished by feeding the position estimates of the template from the Kalman filter to a set of three PID-based controllers.

The IBVS PIDs will perform all the calculations in the current image frame. Setting a 2D image-based reference for the controller, and thus avoiding the need for expensive 3D reconstructions, increases the computational speed in on-board computers, allowing for real-time control over the approaches of the vehicle to the landing pad [15]. High-rate controllers tend to be robust against sudden image changes, and with the Kalman filter output as the reference for control, the system is capable of tracking the landing platform even if it is abruptly moved out of the camera's field of view.

Our approach uses a set of three PID-based controllers attached in a cascade in an outer loop, with the two native controllers already implemented in the Pixhawk flight stack. The controllers

of the flight stack have a standard cascaded position-velocity loop, in which the outer position loop transforms the position inputs to velocity outputs and the velocity outputs are converted in the inner loop into thrust commands for the vehicle's propellers. The idea is to transform pixel coordinate errors into velocity commands and to let the inner controllers of the Flight Controller Unit (FCU) handle the thrust.

We use a reference vector for the controllers $\boldsymbol{S_p} = [\frac{I_w}{2}, \frac{I_h}{2}, 0]^T$, where $\frac{I_w}{2}, \frac{I_h}{2}$ represent the center of the current image frame and zero corresponds to the desired angle between the aircraft and template measured with respect to the x axis of the image. These controllers command the x, y velocities of the rotorcraft, denoted as $\dot{x}_a, \dot{y}_a$, to center the vehicle with respect to the template detected in the current image frame. The third controller modifies the yaw rate $\dot{\psi}$ of the aircraft in order to align it with the landing pad in the x axis. The error vector $\boldsymbol{e}_t \in \mathbb{R}^3$ of the controllers at time t is given by (16).

$$\boldsymbol{e}_t = \boldsymbol{S_{p_t}} - \boldsymbol{X}_t^{(i=1:3)} \tag{16}$$

Fig. 5 is a simplified representation of the three PID-based controllers and an altitude controller with an ON/OFF strategy to control the descent of the UAV. The error vector $\boldsymbol{e}$ is used to feed the first three controllers and to produce a control effort $\boldsymbol{U}_t \in \mathbb{R}^3$, which is delivered to the cascade controllers of the FCU. The output of the three PID controllers is provided by the vector in (17). Each PID controller was discretized using trapezoidal integration and derivation.
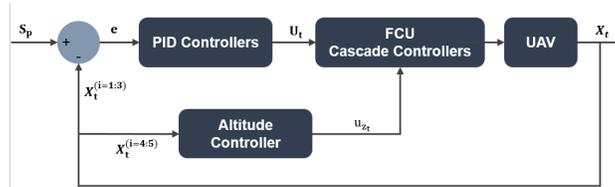
$$\boldsymbol{U}_t = [\dot{x}_a, \dot{y}_a, \dot{\psi}]^T \tag{17}$$

Fig. 5: The proposed control strategy attached to the native FCU controllers. The PID controllers block outputs the control signal $\boldsymbol{U}_t$ with $\dot{x}_a, \dot{y}_a, \dot{\psi}$, whereas the altitude controller outputs u_{z_t} to control the descent of the vehicle.

The ON/OFF altitude controller in Fig. 5 starts to land the aircraft whenever the difference between the height and width of the template estimates $\boldsymbol{X}_t^{(i=4:5)}$ tends to zero (18). u_{z_t} represents

the output of the altitude controller, Z_p is the current height in meters of the vehicle and Z_f is a descent constant. This descent condition guarantees that the aircraft will land only if the template dimensions form a square, which is the actual shape of the landing platform.

$$u_{z_t} = \begin{cases} Z_p - Z_f, & \text{if } |O_w - O_h| < 5 \\ Z_p, & \text{otherwise} \end{cases} \tag{18}$$

Acquiring feedback from the vision-based module closes the visual servoing control loop and allows for the implementation of an on-board end-to-end control strategy for a UAV. Algorithm 2 shows pseudo-code for the controller pipeline for the rotorcraft.

V. SIMULATION RESULTS

This section describes the experiments carried out to assess the different modules of the autonomous landing system using a Gazebo-based simulation. We provide an open-source implementation of our system in Github*.

The system was simulated using the SITL provided by PX4, which runs the Pixhawk flight stack in a Gazebo-based environment. Our implementation relies on the SITL simulation environment presented in [32], where the PX4 on SITL is connected via UDP with an offboard API (ROS), ground station and the gazebo simulator. To obtain accurate results, a custom model of a DJI F450 quad-rotor was implemented to mimic the dynamics and physics involved in a real-world model, as shown in Fig. 6 (a). All the perception and control pipelines of the system, shown in Fig. 1, were implemented in the Robot Operating System (ROS). In addition, a custom Gazebo-world with a landing platform was used to rigorously assess the performance of both the vision and control module.

A. Vision module

The assessment of the detection and tracking system was carried out using three different detector-descriptor algorithms, which are efficient to compute and, orientation and scale invariant [30]: ORB, SIFT and SURF. After extracting landing pad detections from the captured aerial images as explained in Section III-A, the Kalman filter was used to estimate the state of the landing pad. Our evaluation procedure demonstrates the improvements obtained by our tracking

---

*https://github.com/MikeS96/autonomous_landing_uav

Algorithm 2 Landing controller

1: **Inputs:**
$\quad\quad \boldsymbol{X} \in \mathbb{R}^{10}$ State vector
$\quad\quad I_w$ image Width
$\quad\quad I_h$ image Height
2: **Initialization:**
$\quad\quad \boldsymbol{PID_{x_a}} \in \mathbb{R}^3$ PID parameters for $\dot{x_a}$
$\quad\quad \boldsymbol{PID_{y_a}} \in \mathbb{R}^3$ PID parameters for $\dot{y_a}$
$\quad\quad \boldsymbol{PID_{\psi}} \in \mathbb{R}^3$ PID parameters for $\dot{\psi}$
$\quad\quad \boldsymbol{S_p} \in \mathbb{R}^3$ $x, y,$ and θ setpoints
$\quad\quad Z_p$ initial height of the vehicle
$\quad\quad Z_f$ descent factor
3: **Outputs:**
$\quad\quad \boldsymbol{U} \in \mathbb{R}^3$ PID control efforts
$\quad\quad u_z$ ON/OFF altitude controller output
4: **for each** state vector X **do**
5: $\quad\quad \boldsymbol{e}_t \leftarrow \boldsymbol{S_{p_t}} - \boldsymbol{X}_t^{(i=1:3)}$
6: $\quad\quad errorSize \leftarrow abs(\boldsymbol{X}_t^{(i=4)} - \boldsymbol{X}_t^{(i=5)})$
7: $\quad\quad$ // Update Z position
8: $\quad\quad$ **if** $errorSize < 5$ and $Z_p > 0.2$ **then**
9: $\quad\quad\quad\quad u_{z_t} \leftarrow Z_p - Z_f$
10: $\quad\quad$ **else**
11: $\quad\quad\quad\quad u_{z_t} \leftarrow Z_p$
12: $\quad\quad$ **end if**
13: $\quad\quad$ // Land if the vehicle is at 0.2 meters from the ground
14: $\quad\quad$ **if** $u_{z_t} <= 0.2$ and $(\boldsymbol{e}_t^{(i=1)}, \boldsymbol{e}_t^{(i=2)}) < 20$ **then**
15: $\quad\quad\quad\quad$ land()
16: $\quad\quad\quad\quad u_{z_t} \leftarrow 0$
17: $\quad\quad$ **end if**
18: $\quad\quad Z_p \leftarrow u_{z_t}$
19: $\quad\quad$ **for** $i = 1, \ldots, 3$ **do**
20: $\quad\quad\quad\quad \boldsymbol{U}^{(i)} \leftarrow computePID(\boldsymbol{S_{p_t}}^{(i)}, \boldsymbol{e}^{(i)}, \boldsymbol{X}^{(i)})$
21: $\quad\quad$ **end for**
22: $\quad\quad$ **return** $\boldsymbol{U}_t, u_{z_t}$
23: **end for**

---

module compared with plain detection. All the detector-descriptors were tested with the aircraft hovering at a height of 3.5 meters above the landing pad.

The RANSAC algorithm was used to compute the homography matrix $\overline{\boldsymbol{H}}$. Both SIFT and SURF used the Manhattan distance to compute the matches between descriptors, whereas ORB employed the Hamming distance. Figure 7 illustrates the results of the three algorithms. The violin plots show the error between the observations $\boldsymbol{Z}$ and the ground truth of the platform. These plots show the distribution of the error for the five observed states, with the median error
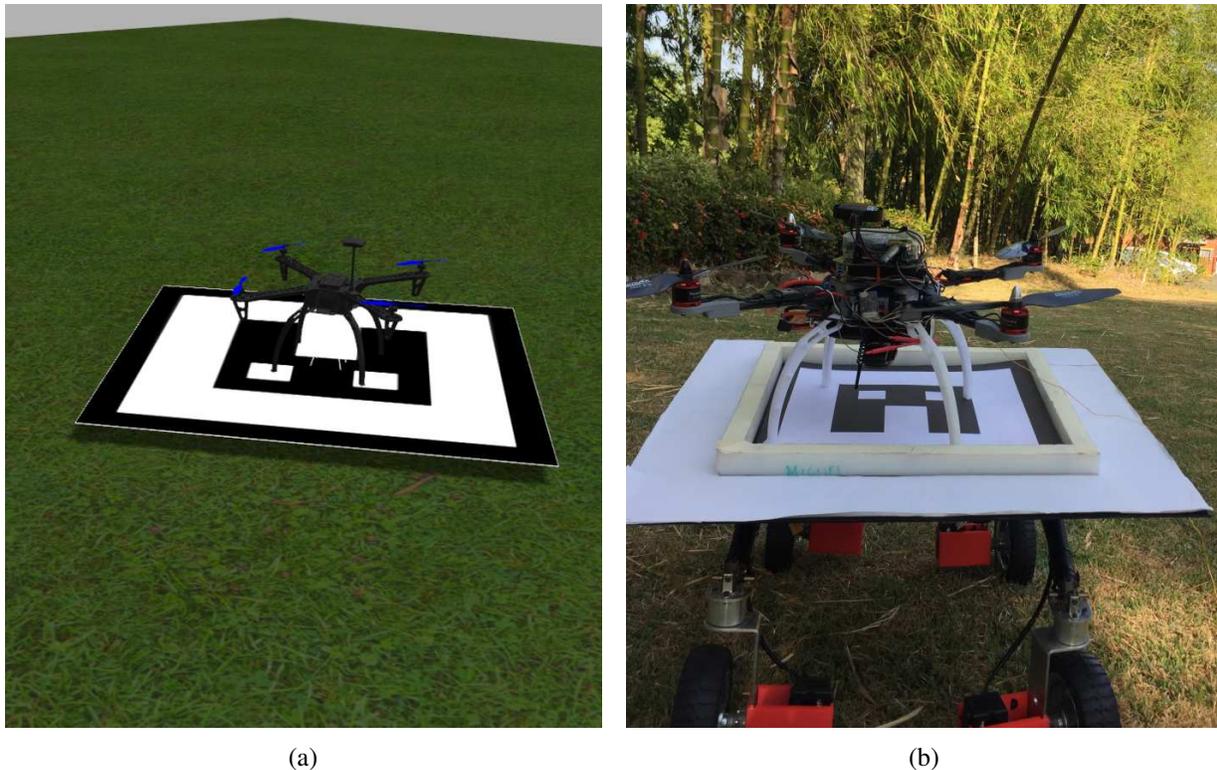
Fig. 6: DJI F450 quad-rotor in the landing pad: (a) Dron in the Gazebo-based simulation; (b) Customized dron with a Pixhawk FCU and Odroid XU4 in field trial.

represented as a white dot, the interquartile range as a broad black bar in the center of the violin, and the lower/upper adjacent values as a thin line.

It can be seen from Fig. 7 (a) and (b) that the centroid coordinates x, y of the landing platform show similar behavior for the ORB and SIFT detectors, with a median value close to zero. In contrast, SURF has more dispersion in its error distribution and a median of above 200 pixels. The best detector for the centroid coordinates is SIFT, as it gives a more uniform distribution compared with ORB and SURF, and most of the error values are clustered close to zero.

Figure 7(c) presents the error in the angle θ, and it can be observed that the SIFT detector gives better performance than the other two detectors. The errors in the width O_w and height O_h can be seen in Fig. 7(d) and (e), respectively. From this figure, it can be seen that the three detectors have very similar behavior for both variables, although SIFT outperforms ORB and SURF with an error distribution close to zero and few outliers.

The SIFT detector-descriptor is better than the other detectors for all observations Z. Although ORB shows similar behavior to SIFT for the first three states, it has a large set of outliers for
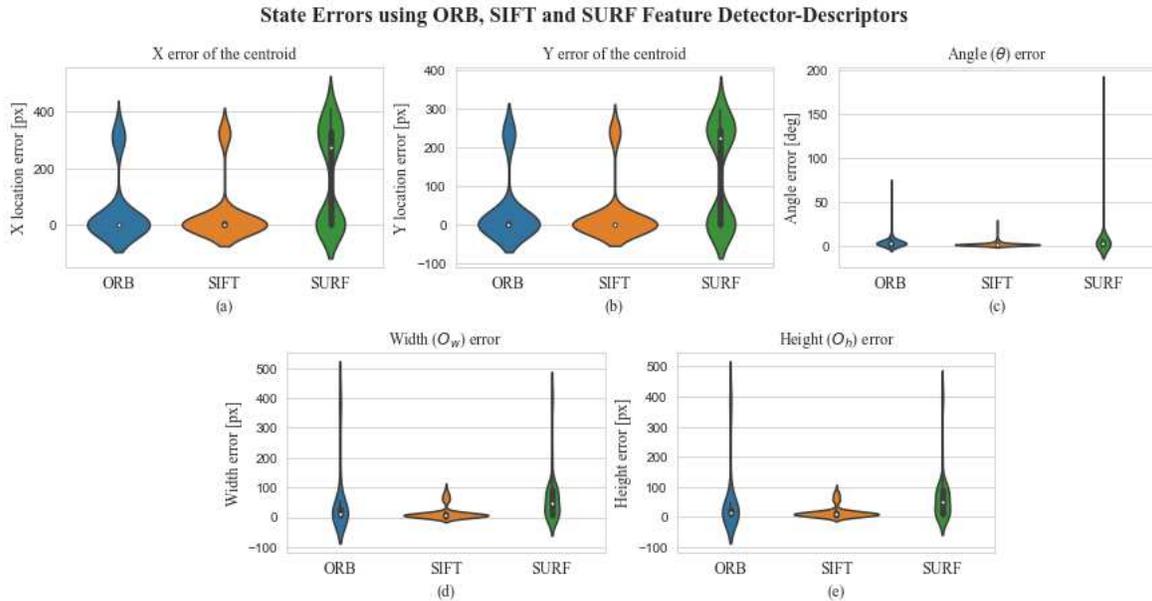
Fig. 7: Estimation errors for the different descriptor-detector algorithms used for generating state observations (Z): ORB to the left, SIFT in the middle and SURF to the right, of each sub-figure respectively. (a) centroid's X error; (b) centroid's Y error; (c) Orientation θ error; (d) width O_w error; (e) height O_h error.

the last two states, while SURF gives the worst performance throughout the observation space.

TABLE I

Comparison between plain SIFT detector and SIFT detector with Kalman filter

States	Using SIFT only		SIFT with Kalman Filter	
	Average	Standard deviation	Average	Standard deviation
Centroid X [px]	41.93	105.12	2.74	4.71
Centroid Y [px]	30.16	78.55	1.12	2.91
Angle [deg]	1.81	2.69	1.31	2.02
Width [px]	15.98	22.40	7.00	9.71
Height [px]	18.08	20.43	9.36	6.20

Finally, Table I shows the average and standard deviation in the errors in pixels between the SIFT detector and the Kalman filter attached to the SIFT detector. The results demonstrate that all of the observations are substantially improved with the Kalman filter, reducing the average error to almost zero and decreasing the standard deviation of each state. This analysis leads to the conclusion that the detection and tracking pipeline can accurately track the landing platform
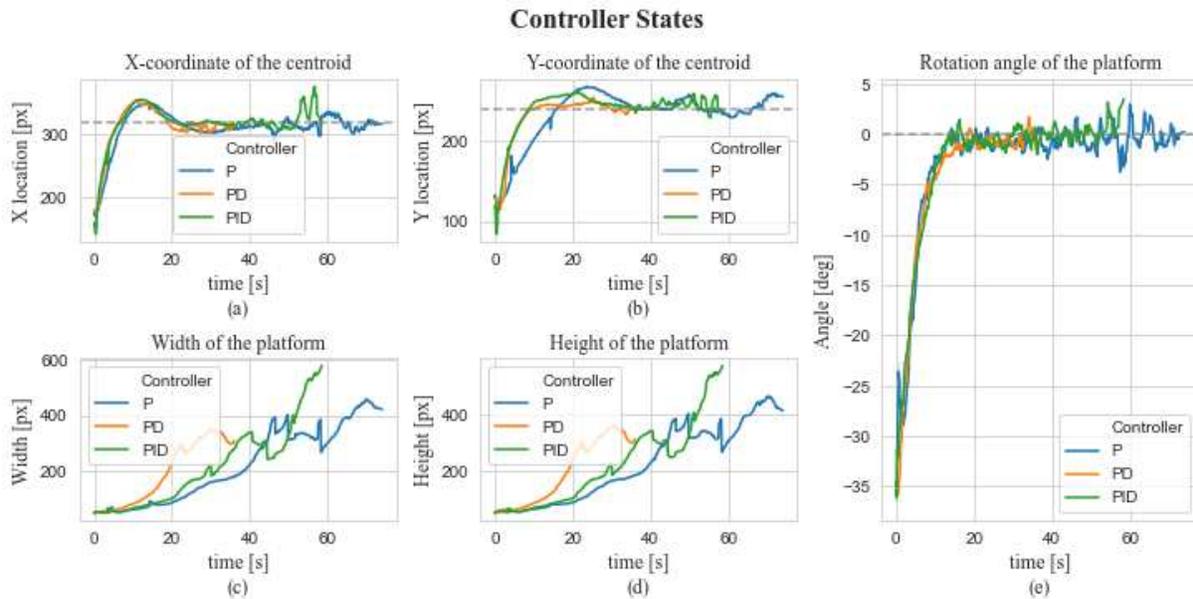
Controller States

Fig. 8: Output of the P, PD and PID controllers for each state in $\boldsymbol{X}_t^{(i=1:5)}$: (a) X coordinate of the centroid; (b) Y coordinate of the centroid; (c) width of the platform O_w; (d) height of the platform O_h; (e) heading θ.
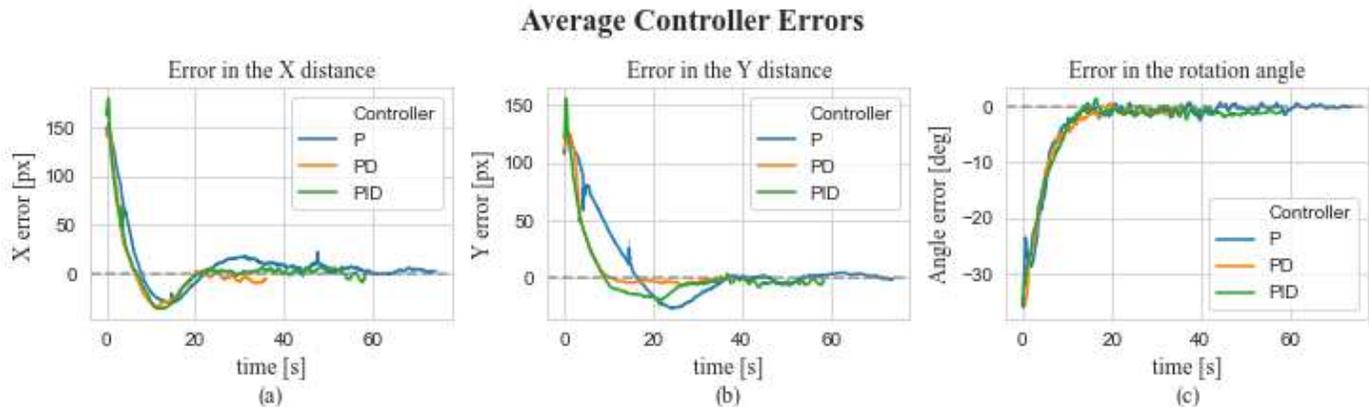
Average Controller Errors

Fig. 9: Error in the P, PD and PID controllers for $\boldsymbol{X}_t^{(i=1:3)}$: (a) X error for the centroid; (b) Y error for the centroid; (c) error in the heading θ.

with a SIFT detector and a linear Kalman filter to facilitate the computations in the on-board computers of a small UAV.

B. Controller

To assess the performance of the IBVS control system, three PID-based strategies were implemented. Various tests were carried out with P, PD and PID controllers to determine which

TABLE II
Controller errors in the landing process

Controller		Centroid X [px]	Centroid Y [px]	Angle [deg]
P	RMSE	30.3725	34.0250	6.7023
	Standard deviation	28.9982	32.9310	6.1946
PD	RMSE	41.8702	40.2214	11.3106
	Standard deviation	41.5917	37.2056	9.4058
PID	RMSE	31.8581	29.0585	7.7358
	Standard deviation	31.6203	29.0283	6.9005

was optimal for the landing procedure. For each control strategy, the gains were tweaked in a gazebo-based simulation until the most stable parameters were found for each controller. Using the best gains, five landing trials were conducted in a custom Gazebo environment, and the results were averaged. The image size was set to 640×320 pixels, and the SIFT detector-descriptor was employed.

Fig. 8 presents the output of the three controllers for each state $\boldsymbol{X}_t^{(i=1:5)}$. The first two figures (Fig. 8(a) and (b)) correspond to the x, y centroid of the landing platform. It can be seen that the three controllers were capable of tracking the 2D reference provided by the vision module and to center the vehicle on the pad. However, the P strategy (blue) operated more slowly than the PD (orange) and PID (green) strategies, which tended to land the aircraft faster.

In a similar fashion, all of the controllers were shown to be capable of aligning the heading of the vehicle with the landing platform, as shown in Fig. 8(e). The estimated width O_w and height O_h of the landing pad, as illustrated in Fig. 8(c) and (d), have a tendency to increase as the altitude controller starts the vehicle's descent. This effect is due to the landing platform becoming bigger in the current image frame as the height of the aircraft decreases.

Although all of the controllers were capable of landing the aircraft, in order to perform a thorough assessment we present, the error for each controller for the states $\boldsymbol{X}_t^{(i=1:3)}$ in Table II. From this table, it can be seen that the RMSE and standard deviation (in pixels) for each controller are strikingly similar for the three states under consideration. The P and PID controllers gave better numerical results than the PD controller. However, this behavior was due to the landing speed of the PD controller; since it is capable of landing more quickly, there are fewer samples to compute the RMSE. The PD controller landed in approximately 36 seconds, around 25 seconds
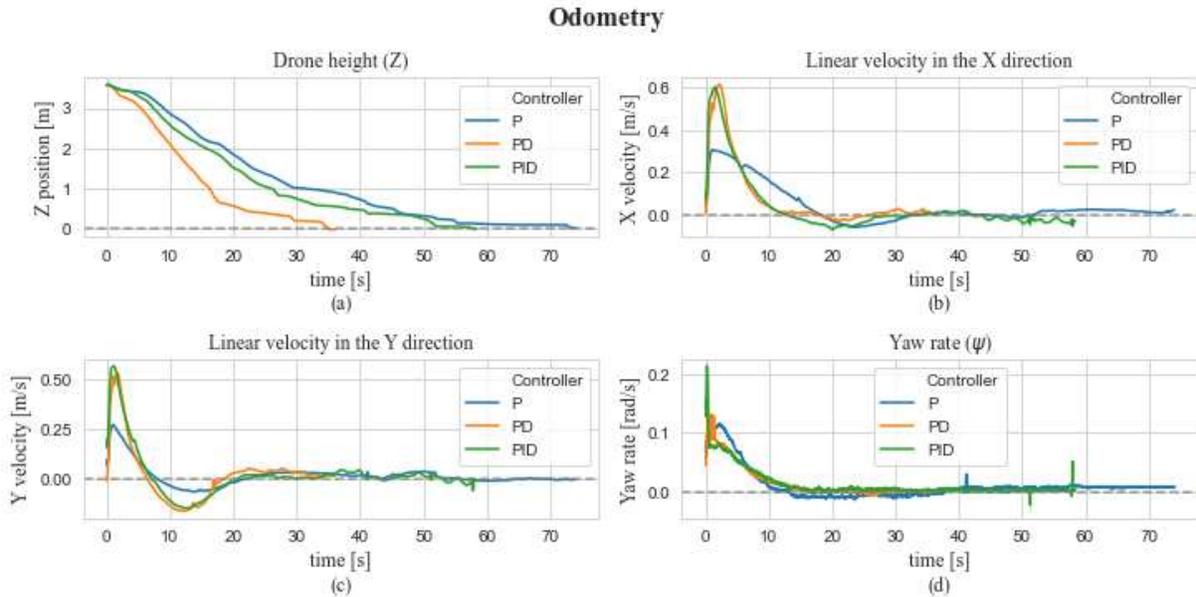
Fig. 10: Odometry of the vehicle during the landing process: (a) height of the vehicle Z_p; (b) X velocity $\dot{X}_a$; (c) Y velocity $\dot{Y}_a$, (d) yaw rate $\dot{\psi}$.

faster than the PID controller.

Fig. 9 complements the information in Table II by presenting the dynamic behavior of the error in the first three states $\boldsymbol{X}_t^{(i=1:3)}$ for each controller while the aircraft is landing. As mentioned above, the P controller is slower than the other two controllers. PD tends to be a faster strategy and has fewer overshoots in its dynamic behavior. The performance of PID seems to be between those of the other two controllers.

The odometry of the vehicle is presented in Fig. 10 for four different variables for each controller. The first plot in Fig. 10(a), shows how the altitude of the vehicle is reduced to zero for each controller. Both of the linear velocities of the aircraft $\dot{x}_a, \dot{y}_a$ undergo substantial variation at the beginning of the tests, as shown in Fig. 10(b) and (c), but when the vehicle is centered with respect to the landing platform, the linear speeds tend to zero. Likewise, the yaw rate $\dot{\psi}$ shown in Fig. 10(e) behaves as expected for the three controllers: its magnitude reduced to zero, which means that the vehicle is correctly aligned with the landing platform.

The position and heading errors between the landing platform and the aircraft were computed for the different trials, and are shown in Table III. It can be seen from the table that the average error in the x, y coordinates is less than 3.0 centimeters for all the control strategies. Similarly, the error in the angle θ between the vehicle and the platform is less than 1.2 degrees. This

TABLE III
Errors in the landing simulation

Controller		Centroid X [m]	Centroid Y [m]	Angle [deg]
P	Average	0.0244	0.0294	0.6531
	Standard deviation	0.0079	0.0178	0.3770
PD	Average	0.0178	0.0274	0.7563
	Standard deviation	0.0194	0.0155	0.8444
PID	Average	0.0288	0.0232	1.1115
	Standard deviation	0.0261	0.0253	0.5323

demonstrates that all of the controllers are capable of achieving a precision landing of the aircraft with small errors over different trials, confirming the efficiency of the vision-based system with various control strategies.

Although all of the controllers were capable of accurately landing the vehicle on the landing platform, the best performance was shown by the PD and PID controllers, as these had more stable responses and lower variations in the different attempts. Although the PD strategy is less accurate than the PID controller, it is the preferred option due to its speed in landing the aircraft.

To assess the robustness of the PD controller under low light conditions and different wind disturbance, Fig. 11 presents the errors obtained over different trials while the aircraft is landing. The error in X illustrated in Fig. 11(a) shows how the aircraft is capable to minimize it towards zero with different wind conditions. Similarly, the error in Y presented in 11(b) demonstrates a similar behavior as 11(a) where the error is minimized, nevertheless, with bigger wind disturbances the aircraft is prone to experience an overdamped response rather than underdamped as Fig. 9 demonstrated. Finally, the angle θ is considerably affected by the wind disturbances in 11(c) as the vehicle is not capable to align itself with respect to the landing platform. However, the vehicle was capable to land in all tests, validating the effectiveness of our method while landing with unideal conditions.

VI. EXPERIMENTAL RESULTS

This section presents the results obtained in real-world tests using a DJI F450 in an autonomous landing sequence.

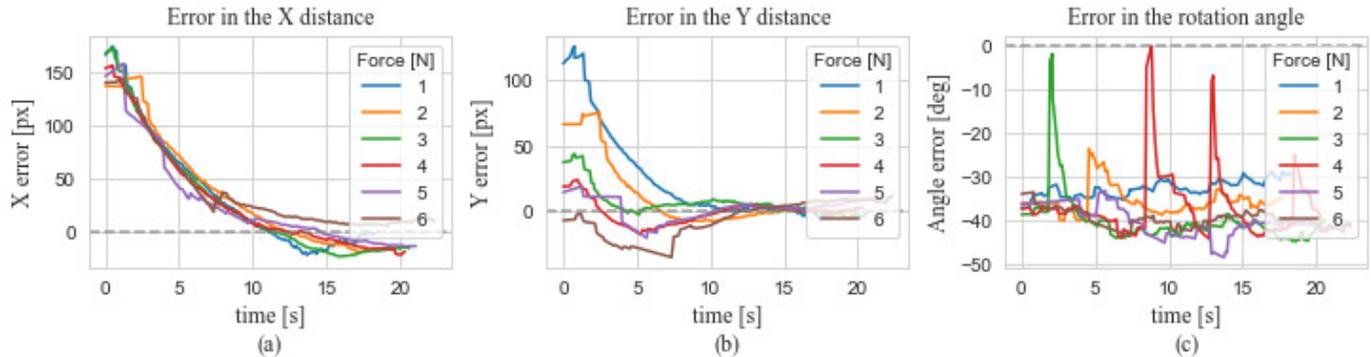**Controller Errors with Low Illumination and Wind Disturbances**

Fig. 11: Error in the PD controller for $X_t^{(i=1:3)}$ with low illumination and different wind disturbances: (a) X error for the centroid; (b) Y error for the centroid; (c) error in the heading θ.

To thoroughly assess the performance of the autonomous landing system, a custom DJI F450 with an Odroid XU4 on-board computer and Pixhawk 1, as shown in Fig. 6 (b), was used to test the developed framework. Due to the limited computational resources of the Odroid, the PD controller was employed, as this was the fastest method of landing the vehicle, and the result of three landing trials were averaged to evaluate the system. The size of the image was also reduced to 320 x 240 pixels to obtain a frame rate of 15 FPS and to ensure system convergence. The SIFT feature detector-descriptor was used (based on the simulation results) to carry out these tests.

To bridge the algorithms developed during the simulation phase with the real-world, it was necessary to unplug the SITL component. This was achieved by connecting the Pixhawk FCU to the on-board computer and launching all the nodes developed in ROS. This process guaranteed that the system was connected with the physical FCU, bypassing the need for the SITL component. The detection and control pipeline will therefore operate directly in the custom rotor-craft, enabling it to carry out autonomous landing maneuvers. All the parameters used during the simulation where transferred to the aircraft without *finetuning* to demonstrate that the use of simple vision and control models allow for zero-shot domain transfer.

Fig. 12 presents the results obtained with the PD controller for each state $X_t^{(i=1:5)}$. As expected, the system is capable of landing the rotor-craft on the landing platform within approximately 35 seconds. The real-world system displays more spiky behavior than the simulated vehicle (Fig. 8
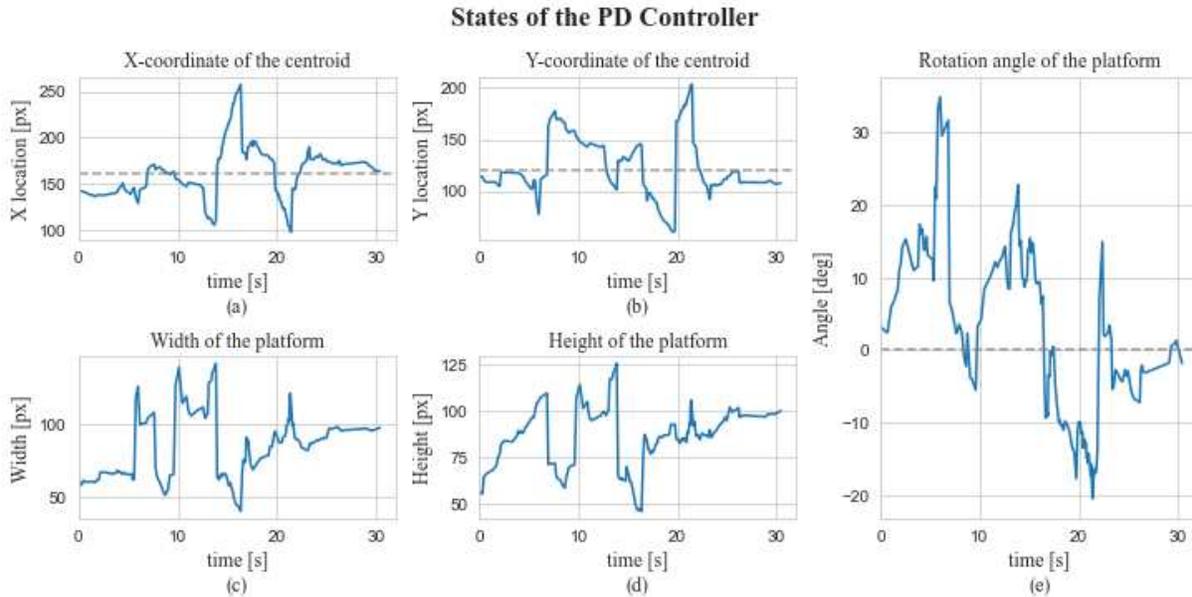
States of the PD Controller

Fig. 12: Output of the PD controller for each state in $\boldsymbol{X}_t^{(i=1:5)}$, as tested with a DJI F450: (a) X-coordinate of the centroid; (b) Y-coordinate of the centroid; (c) width of the platform O_w; (d) height of the platform O_h; (e) heading θ.

TABLE IV
Controller errors in experimental landing process

Controller		Centroid X [px]	Centroid Y [px]	Angle [deg]
PD	RMSE	39.7460	36.4111	8.7761
	Standard deviation	39.7583	31.5632	8.7724

(orange)); however, as the test advances, the response of the controller stabilizes, guaranteeing the appropriate landing of the UAV.

Comparably, the RMSE of the controller during the landing procedure was also assessed and presented in Table IV. This error was computed over the three landing trials conducted with the real-world rotor-craft while the vehicle was trying landing. Altogether, it is possible to appraise that the vehicle maintains strikingly similar values of RMSE for the variables X, Y, θ when compared with the RMSE presented for the simulation in Table II. In fact, the RMSE is slightly reduced within the real-world landing trials. The plots of these errors are unshown as their dynamic behavior is similar as the ones presented in Fig. 9 (orange).

Finally, to complete the assessment process, Table V [33] presents the position error between

TABLE V

Experimental landing errors measured from the center of the pad to the center of the vehicle

Controller		Centroid X [m]	Centroid Y [m]
PD	Average	0.1314	0.1592
	Standard deviation	0.1041	0.1344

the vehicle and the landing platform. This error was computed as the distance from the center of the pad to the center of the rotor-craft once it had landed. It can be seen that the average value is less than 16 centimeters. Compared to the results in Table III, the error in the real-world implementation of the PD controller is around five times that of the simulation. Although these results seem undesirable, the rotor-craft is capable of precisely landing on the desired platform and accomplishing the autonomous landing task, as expected from the simulation results.

VII. CONCLUSION

This paper presents a SITL approach to developing a monocular image-based autonomous landing system for quadcopter drones. The proposed method and system, which integrates ROS, Gazebo and PX4's SITL tools, enables users to not only endow quadcopters with low-cost vision-based autonomous landing capabilities, but also to fine-tune all the parameters of a potentially dangerous device in a safe simulated environment. With minimal modifications, both the vision and control modules developed in our simulated environment, were successfully validated in a physical DJI F450 with an Odroid XU4 on-board computer and a Pixhawk 1 flight controller.

ACKNOWLEDGMENT

REFERENCES

[1] S. Bayraktar and E. Feron, "Experiments with Small Unmanned Helicopter Nose-Up Landings," Journal of Guidance, Control, and Dynamics, vol. 32, no. 1, pp. 332–337, 2009, doi: 10.2514/1.36470.

[2] D. H. Ye, J. Li, Q. Chen, J. Wachs, and C. Bouman, "Deep Learning for Moving Object Detection and Tracking from a Single Camera in Unmanned Aerial Vehicles (UAVs)," Electronic Imaging, vol. 2018, no. 10, 2018, doi: 10.2352/ISSN.2470-1173.2018.10.IMAWM-466.

[3] R. Jin, H. M. Owais, D. Lin, T. Song, and Y. Yuan, "Ellipse proposal and convolutional neural network discriminant for autonomous landing marker detection," Journal of Field Robotics, vol. 36, no. 1, pp. 6–16, 2018, doi: 10.1002/rob.21814.

[4] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue," IEEE Robotics & Automation Magazine, vol. 19, no. 3, pp. 46–56, 2012, doi: 10.1109/MRA.2012.2206473.

[5] A. Gautam, P. B. Sujit and S. Saripalli, "A survey of autonomous landing techniques for UAVs," 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, 2014, pp. 1210-1218, doi: 10.1109/ICUAS.2014.6842377.

[6] S. Yang, S. A. Scherer, K. Schauwecker, and A. Zell, "Autonomous Landing of MAVs on an Arbitrarily Textured Landing Site Using Onboard Monocular Vision," Journal of Intelligent & Robotic Systems, vol. 74, no. 1-2, pp. 27–43, 2013, doi: 10.1007/s10846-013-9906-7.

[7] J. Thomas, J. Welde, G. Loianno, K. Daniilidis and V. Kumar, "Autonomous Flight for Detection, Localization, and Tracking of Moving Targets With a Small Quadrotor," in IEEE Robotics and Automation Letters, vol. 2, no. 3, pp. 1762-1769, July 2017, doi: 10.1109/LRA.2017.2702198.

[8] Yu L, Luo C, Yu X, et al. Deep learning for vision-based micro aerial vehicle autonomous landing. International Journal of Micro Air Vehicles. June 2018:171-185. doi:10.1177/1756829318757470

[9] Mittal, Payal et al. Deep learning-based object detection in low-altitude UAV datasets: A survey. Image Vis. Comput. 104 (2020): 104046.

[10] O. Araar, N. Aouf, and I. Vitanov, "Vision Based Autonomous Landing of Multirotor UAV on Moving Platform," Journal of Intelligent &; Robotic Systems, vol. 85, no. 2, pp. 369–384, 2016, doi: 10.1007/s10846-016-0399-z.

[11] Yang, T.; Li, P.; Zhang, H.; Li, J.; Li, Z. Monocular Vision SLAM-Based UAV Autonomous Landing in Emergencies and Unknown Environments. Electronics 2018, 7, 73.

[12] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.

[13] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti and S. Longhi, "A single-camera feature-based vision system for helicopter autonomous landing," 2009 International Conference on Advanced Robotics, Munich, 2009, pp. 1-6.

[14] B. Herissé, T. Hamel, R. Mahony and F. Russotto, "Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow," in IEEE Transactions on Robotics, vol. 28, no. 1, pp. 77-89, Feb. 2012, doi: 10.1109/TRO.2011.2163435.

[15] D. Lee, T. Ryan and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, 2012, pp. 971-976, doi: 10.1109/ICRA.2012.6224828.

[16] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization fortransferring deep neural networks from simulation to the real world. InIntelligent Robots andSystems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017.

[17] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS; A Modular Gazebo MAV Simulator Framework", "Robot Operating System (ROS): The Complete Reference (Volume 1)", pp. 595-625", 2016, isbn: "978-3-319-26054-9", doi="10.1007/978-3-319-26054-9. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS; A Modular Gazebo MAV Simulator Framework", "Robot Operating System (ROS): The Complete Reference (Volume 1)", pp. 595-625", 2016, isbn: "978-3-319-26054-9", doi="10.1007/978-3-319-26054-9_23", url="http://dx.doi.org/10.1007/978-3-319-26054-9_23"

[18] C. Patruno, M. Nitti, A. Petitti, E. Stella, and T. D'Orazio, "A Vision-Based Approach for Unmanned Aerial Vehicle Landing," Journal of Intelligent &; Robotic Systems, vol. 95, no. 2, pp. 645–664, 2018, doi: 10.1007/s10846-018-0933-2.

[19] M. F. Sani and G. Karimian, "Automatic navigation and landing of an indoor AR. drone quadrotor using ArUco marker

and inertial sensors," 2017 International Conference on Computer and Drone Applications (IConDA), Kuching, 2017, pp. 102-107, doi: 10.1109/ICONDA.2017.8270408.

[20] J. Hermansson, A. Gising, M. Skoglund and T. B. Schön, "Autonomous Landing of an Unmanned Aerial Vehicle," 2010 Reglermöte (Swedish Control Conference), Lund, Sweden, June 2010.[Online]. Available: https://www.researchgate.net/publication/229027579_Autonomous_Landing_of_an_Unmanned_Aerial_Vehicle

[21] M. S. Ruiz, A. M. P. Vargas and V. Romero-Cano, "Detection and tracking of a landing platform for aerial robotics applications," 2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA), Barranquilla, 2018, pp. 1-6, doi: 10.1109/CCRA.2018.8588112.

[22] A. Chavez, D. L'heureux, N. Prabhakar, M. Clark, W.-L. Law, and R. J. Prazenica, "Homography-Based State Estimation for Autonomous UAV Landing," AIAA Information Systems-AIAA Infotech @ Aerospace, 2017, doi: 10.2514/6.2017-0673.

[23] V. Sudevan, A. Shukla and H. Karki, "Vision based autonomous landing of an Unmanned Aerial Vehicle on a stationary target," 2017 17th International Conference on Control, Automation and Systems (ICCAS), Jeju, 2017, pp. 362-367, doi: 10.23919/ICCAS.2017.8204466.

[24] M. Bloesch, S. Omari, M. Hutter and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 298-304, doi: 10.1109/IROS.2015.7353389.

[25] F. Mercado-Rivera, A. Rojas-Arciniegas and V. Romero-Cano, "Probabilistic Motion Inference for Fused Filament Fabrication," 2020 Printing for Fabrication: materials, applications, and processes, 2020, pp. 85-91, doi: 10.2352/ISSN.2169-4451.2020.36.85.

[26] O. L. Rojas-Perez, R. Munguia-Silva, and J. Martinez-Carranza, "Real-time landing zone detection for uavs using single aerial images," 10th international micro air vehicle competition and conference, Melbourne, Australia, 2018, p. 243–248. [Online]. Available: http://www.imavs.org/papers/2018/IMAV_2018_paper_45.pdf

[27] Y. Yang, H. Gong, X. Wang, and P. Sun "Aerial Target Tracking Algorithm Based on Faster R-CNN Combined with Frame Differencing," Aerospace, vol. 4, no. 2, p. 32, 2017, doi: 10.3390/aerospace4020032.

[28] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. D. L. Puente, and P. Campoy, "A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform," Journal of Intelligent & Robotic Systems, vol. 93, no. 1-2, pp. 351–366, 2018, doi: 10.1007/s10846-018-0891-8.

[29] Y. Uchida, "Local feature detectors descriptors and image representations: A survey", 2016, arXiv:1607.08368.

[30] J. Perafan-Villota, F. Leno-da-Silva, R. de-Souza-Jacomini, and A. Reali-Costa, "Pairwise registration in indoor environments using adaptive combination of 2D and 3D cues," Image and Vision Computing, vol. 69, pp. 113-124, 2018, doi: 10.1016/j.imavis.2017.08.008.

[31] T. D. Barfoot, "State Estimation for Robotics", 2019, Cambridge University Press, Toronto.

[32] Simulation, PX4 User Guide, PX4, 2021. [Online]. Available: https://docs.px4.io/master/en/simulation/.

[33] M. Saavedra Ruiz and A. Pinto Vargas, "Desarrollo de un sistema de aterrizaje autónomo para un vehículo aéreo no tripulado sobre un vehículo terrestre", Red.uao.edu.co, 2019. [Online]. Available: http://hdl.handle.net/10614/10754