IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 22, 2021, accepted February 16, 2021, date of publication March 9, 2021, date of current version March 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3064758

MAT-Fly: An Educational Platform for Simulating Unmanned Aerial Vehicles Aimed to Detect and Track Moving Objects

GIUSEPPE SILANO[1,2], (Student Member, IEEE), AND LUIGI IANNELLI[2], (Senior Member, IEEE)
[1]Faculty of Electrical Engineering, Czech Technical University in Prague, 16636 Prague, Czech Republic
[2]Department of Engineering, University of Sannio in Benevento, 82100 Benevento, Italy

Corresponding author: Giuseppe Silano (giuseppe.silano@fel.cvut.cz)

This work was supported in part by the ECSEL Joint Undertaking Research and Innovation Program COMP4DRONES under Grant 826610, and in part by the European Union's Horizon 2020 Research and Innovation Program AERIAL-CORE under Grant 871479.

ABSTRACT The main motivation of this work is to propose a simulation approach for a specific task within the Unmanned Aerial Vehicle (UAV) field, i.e., the visual detection and tracking of arbitrary moving objects. In particular, it is described MAT-Fly, a numerical simulation platform for multi-rotor aircraft characterized by the ease of use and control development. The platform is based on Matlab® and the MathWorks™ Virtual Reality (VR) and Computer Vision System (CVS) toolboxes that work together to simulate the behavior of a quad-rotor while tracking a car that moves along a nontrivial path. The VR toolbox has been chosen due to the familiarity that students have with Matlab and because it does not require a notable effort by the user for the learning and development phase thanks to its simple structure. The overall architecture is quite modular so that each block can be easily replaced with others simplifying the code reuse and the platform customization.

Some simple testbeds are presented to show the validity of the approach and how the platform works. The simulator is released as open-source, making it possible to go through any part of the system, and available for educational purposes.

INDEX TERMS Educational, Matlab/Simulink, image-based visual servoing, trajectory control, vision detection and tracking, software-in-the-loop, unmanned aerial vehicles, multi-rotor.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), although originally designed and developed for defense and military purposes (e.g., aerial attacks or military air covering), in the recent years gained an increasing interest and attention related to civilian use. Nowadays, UAVs are employed for several tasks and services like surveying and mapping [1], for spatial information acquisition and buildings inspection [2], data collection from inaccessible areas [3], agricultural crops and monitoring [4], manipulation and transportation or navigation purposes [5].

Many existing algorithms for the autonomous control [6] and navigation [7] are provided in the literature, but it is particularly difficult to make the UAVs able to work autonomously in constrained and cluttered environments or also indoors. Thus, it follows the need for tools that allow to understand what it happens when some new applications are going to be developed in unknown or critical situations. Simulation is one of such helpful tools, widely used in robotics [8]–[12], whose main benefits are costs and time savings, enabling not only to create various scenarios, but also to carry out and to study complex missions that might be time consuming and risky in real world applications. Moreover, bugs and mistakes cost virtually nothing: it is possible to crash a vehicle several times and thereby getting a better understanding of implemented methods under various conditions. Thus, simulation environments are very important for fast prototyping and educational purposes, although they may have some drawbacks and limitations, such as the lack of

The associate editor coordinating the review of this manuscript and approving it for publication was Halil Ersin Soken.

noisy real data or the fact that simulated models are usually incomplete or inaccurate. Despite the limitations, the advantages that the simulation provides are more, as like as to manage the complexity and heterogeneity of the hardware, to promote the integration of new technologies, to simplify the software design, to hide the complexity of low-level communication [13].

Different solutions, typically based on external robotic simulators such as Gazebo [14], V-REP [15], AirSim [16], MORSE [17], are available. They employ recent advances in computation and computer graphics (e.g., AirSim is a photorealistic environment [7]) in order to simulate physical phenomena (e.g., gravity, magnetism, atmospheric conditions) and perception (e.g., providing sensor models) in such a way that the environment realistically reflects the actual world. In some cases, those solutions do not have enough features that could allow to create large scale complex environments close to reality. On the other hand, when the tools provide such possibilities, they are difficult to use or they require high computational capabilities [16]. Definitely, it comes out that simulating the real world is a nontrivial task, not only due to multiple phenomena that need to be modeled, but also because their complex interactions ask the user a notable effort for the learning and development phase. For all such reasons, having a complete software platform that makes possible to test different algorithms and control strategies for UAVs moving in a simulated 3D environment is increasingly important both for the whole design process and for educational purposes.

In this paper, it is presented a software platform in which detection, tracking and control algorithms can be evaluated and tested all together in a 3D graphical tool. Due to the simple implementation and the limited possibilities of interfacing it with dedicated middlewares (e.g., ROS [18], YARP [19], GenoM [20]), the proposed platform should be meant with an educational purpose. However, that does not imply a loss of generality nor makes the platform less important. Indeed, as highlighted in [21], the use of interactive learning approaches allows students to improve their technical knowledge and communication skills, giving them the experience of what they will encounter in a real world environment. Therefore, the platform can be appreciated for its potentialities thanks to the advantages coming from the use of a Software-in-the-loop (SIL) approach [6], [22], [23]. In other words, the functionalities provided by the simulator can be easily expanded by students, researchers, and developers modifying or integrating new vehicles dynamics (e.g., hexarotor [24], fully actuated platform [25]), control algorithms (e.g., geometric control laws [26], flatness-based control methods [27]) or detection and tracking techniques (e.g., YOLO [28], [29], Fast R-CNN [29], [30]) for their purposes.

Compared to the commercial and open-source platforms available in the literature [31]–[35], the proposed framework runs on a built-in environment (i.e., Matlab and its toolboxes) and has no constraints in terms of hardware (e.g., memory, unit processor, etc.). Moreover, the simulator is self-contained (i.e., everything is in one place) and can also be used by people without programming skills (i.e., algorithms are typically written in the most common programming languages). Matlab and the Computer Vision System (CVS)[1] and Virtual Reality (VR)[2] toolboxes are the only tools the user needs to work with.

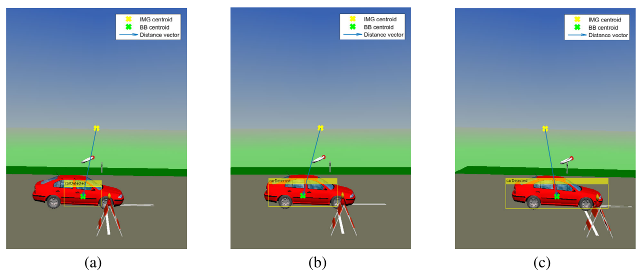


FIGURE 1. Three consecutive frames produces as output by the CAMShift algorithm while tracking the target. The image and the bounding box centroids as well as the distance vector among centroids are depicted (see Sec. IV).

The specific domain of interest regards the behavior of multi-rotor aircraft acting in accordance with the Image-Based Visual Servoing (IBVS) approach [36], [37]. The *eye-in-hand* camera configuration [38] along with the pinhole camera model is considered for the aerial vehicle. Compared to other approaches [39], the camera is rigidly attached to the UAV frame and moves according to the aircraft motion.

The application that is considered is an extension of the authors' previous work [40], that has been revised for making the aircraft able to detect and track a specific object (a car) moving along a nontrivial path. This simple scenario is used as a testbed to show: (i) how the platform works, (ii) the elements that make up the software architecture, and (iii) the adaptability of the platform to the different needs of the user. Compared to the previous work, a tracking algorithm has been added into the loop: the classifier is used to detect the target only at the first step or in case of partial occlusions. Apart from such scenarios, a Continuously Adaptive Mean-Shift (CAMShift) tracking algorithm [41] is employed to follow the car along the path, thus reducing the computational burden and the possibility to lose the target during the tracking. Moreover, in this paper it is proposed a novel procedure based on *ad hoc* Matlab scripts that automatically select the bounding box area of the target (see, Fig. 1) avoiding to use specific Matlab tools, such as *Training Image Labeler*. These scripts also allow comparing various classifier configurations to help select the most suitable for the case study among different features types (e.g., Haar, HOG, LBP) [42] and number of training stages. Finally, the software platform is published as open-source[3] with the aim to share results with other researchers, students, and developers that might use the platform for testing their algorithms and understanding how

[1]https://www.mathworks.com/products/computer-vision.html
[2]https://www.mathworks.com/products/3d-animation.html
[3]https://github.com/gsilano/MAT-Fly

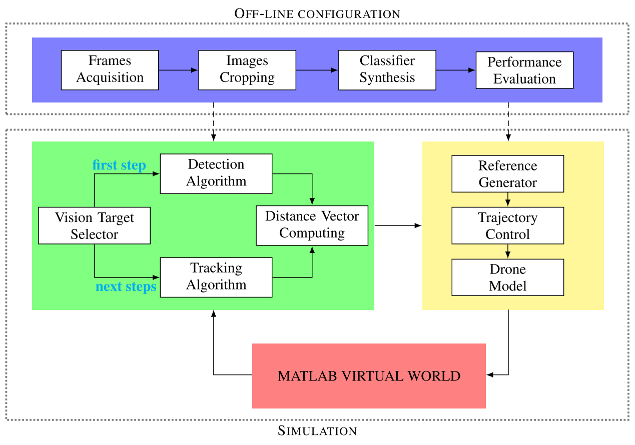


FIGURE 2. The proposed software platform architecture. Arrows represent the data exchanged among blocks and how they interact with each other. Colors point out the four parts making up the system: the classifier training phase (in blue), the vision-based target detection and tracking (in green), the flight control system (in yellow), and the Matlab VR toolbox (in red).

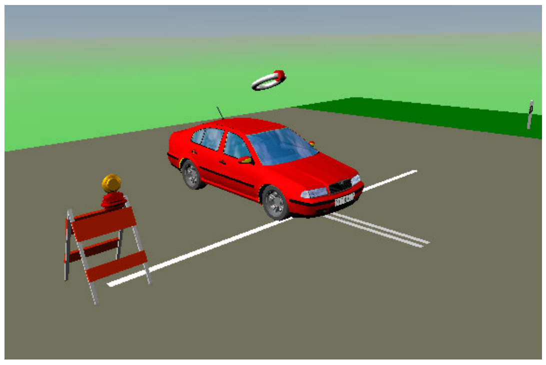


FIGURE 3. Initial frame extracted from the *object tracking example*. The steering angle visualizer allows monitoring the car movements along the path.

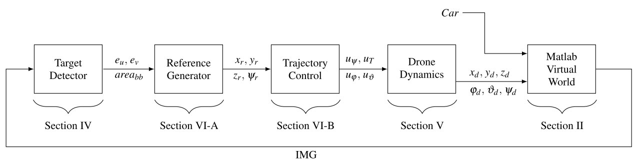


FIGURE 4. The control scheme. Subscript *d* indicates the drone variables, while *r* represents the references to the controller. Each block is mapped with the section that describes its content to help matching the blocks with the corresponding description within the paper.

different approaches can improve the performance and affect the system stability.

The paper is organized as follows. Section II explains the simulation scenario and its functionalities. The classifier training phase and the vision-based target detection and tracking algorithms are presented in Sec. III and IV, respectively. Section V briefly describes the quad-rotor model while numerical results and the control algorithm are reported in Sec. VI. Finally, Section VII concludes the paper.

II. SYSTEM DESCRIPTION

This section aims to describe MAT-Fly and how it works together with the Matlab VR and CVS toolboxes. An illustrative application, i.e., the *object tracking example*, where a drone tracks a car moving along a nontrivial path is considered. An overview of the main elements that make up the system is depicted in Fig. 2.

The software platform is mainly divided into four parts: the classifier training phase (see, Sec. III), the vision-based target detection and tracking (see, Sec. IV), the flight control system (see, Sec. VI), and the Matlab VR toolbox. To facilitate the development of various control and computer vision strategies and the reuse of existing software components, the system was set up using a modular approach splitting each functionality into interchangeable modules. In other words, each part of the system (e.g., the vision-based target detection and tracking, the flight control system) was developed by isolating every feature (e.g., the detection algorithm, the reference generator) in such a way they can be easily replaced with others by facilitating the test and evaluation process.

The Matlab VR toolbox allows to simulate a scenario as much similar as to the real world accounting for the interaction between complex dynamic systems with the surrounding scenario. Moreover, thanks to animation recording functionalities, frames and videos from the scene can be acquired and used to implement an IBVS problem. Also, the tool makes it easy to add external viewpoints to monitor any moving object in the 3D environment from different positions and orientations.

One of the available examples[4] (specifically the *vr_octavia_2cars* example) that describes a quite detailed dynamical model of a car moving along a nontrivial path was used as a starting point (see, Fig. 3). The example represents a standard double-lane-change maneuver [43] conducted in two-vehicles configuration, where one engages the Electronic Stability Program (ESP) control while the other switches off such control unit when changing the lane. From this perspective, a simpler scenario was considered by removing one of the two vehicle configurations, i.e., the car without the ESP controller.

Then, an external viewpoint was added to the scheme for simulating the behavior of a quad-rotor that flies by observing the car moving along the path. In Matlab VR a viewpoint has six Degrees of Freedoms (DoFs): the spatial coordinates x, y, and z, and the angles *yaw* (ψ), *pitch* (ϑ), and *roll* (φ). The whole process is the following: images are updated according to the position and the orientation of the quad-rotor w.r.t. the car; such images are acquired and elaborated for getting the necessary information to detect and track the target, and to run the control strategy designed for the tracking problem. The outputs of the control algorithm consists of the commands u_φ, u_ϑ, u_ψ, and u_T that should be given to the drone in order to update its position (x_d, y_d, and z_d) and orientation (φ_d, ϑ_d, and ψ_d), see Fig. 4.

[4]The list of the ready-to-use scenarios is accessible at https://goo.gl/rtEx3S.

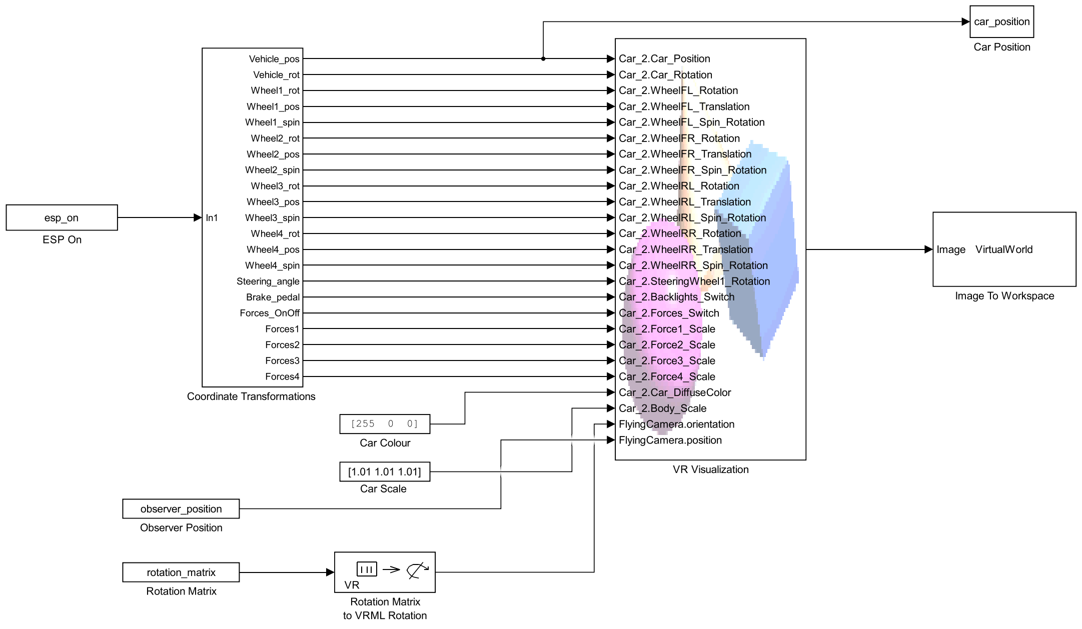


FIGURE 5. Simulink scheme employed for simulating the drone and car dynamics in the 3D simulation environment.

It is worth noticing that ground truth data are used by the tracking controller (see, Sec. VI). Therefore, any analysis can be conducted on the correctness of the data and how this affects the navigation. However, this does not constitute a limitation for the proposed framework thanks to the modular interface exhibited by the platform [44].

In Figure 5 the Simulink scheme employed for simulating the drone and the car dynamics is reported. The *esp_on* and the *coordinates_transformation* blocks compute the steering angle, the linear velocity and the position of the car, and all forces needed to follow a given path. Instead, the *observer_position* and *rotation_matrix* blocks represent the aircraft position and orientation (it is expressed by using the direction cosine matrix [45] and the Rodrigues's formula [46]), respectively. The processed data are sent to the *VR Visualization* block that takes care of the drone and car movements in the simulated scenario.

Note that Matlab VR adopts a reference system (O_{FVR}) [47] slightly different from the classic fixed reference frame O_{FI} (see, Sec. V), thus simple references transformations have been taken into account in all elaborations.

Finally, the Simulink scheme saves the current car position (x_{car}, y_{car}, and z_{car}), used for comparing the drone and the car trajectories (see, Sec. VI-A), and frames of the virtual scenario observed from the drone point of view. Those frames are used, as described in next sections, for pattern recognition.

III. CLASSIFIER TRAINING PHASE

The classifier training phase is the most important part of the system: the object detection and tracking depend on it. Matlab scripts have been developed to automate the entire procedure, from the *frames acquisition* to the *classifier synthesis* and *performance evaluation*. To this aim, the training process has been divided into four parts, as depicted in Fig. 2: the frames acquisition, the bounding box selection, the classifier synthesis and the performance evaluation.

A. FRAMES ACQUISITION

When going to train a classifier, a high number of images is needed. The images are divided into two groups: *positive* (that contain the target) and *negative images*. Following what described in [46], 2626 positive and 10504 negative images were used achieving a $1:4$ ratio in accordance to the Pareto's principle (aka the $80/20$ rule).

For the frames acquisition, a simulation was performed with the quad-rotor moving along a spiral trajectory around the car parked in its initial state (see, Fig. 6). The aircraft attitude and position have been computed for each frame so as described by the sphere surface equations,

$$\begin{cases} y = r\cos\beta \\ x = r\sin\beta\sin\alpha \\ z = r\sin\beta\cos\alpha, \end{cases} \tag{1}$$

where r, the sphere radius, is the distance between the car and the drone (assumed to be fixed and equal to 15 meters), and together with $\alpha \in [0, 2\pi]$ and $\beta \in [0, \pi/2]$ angles, identifies the drone position in the 3D space, as depicted in Fig. 6. A video showing the quad-rotor camera point of view while observing the car parked in its initial state while following the spiral trajectory is available in [48].

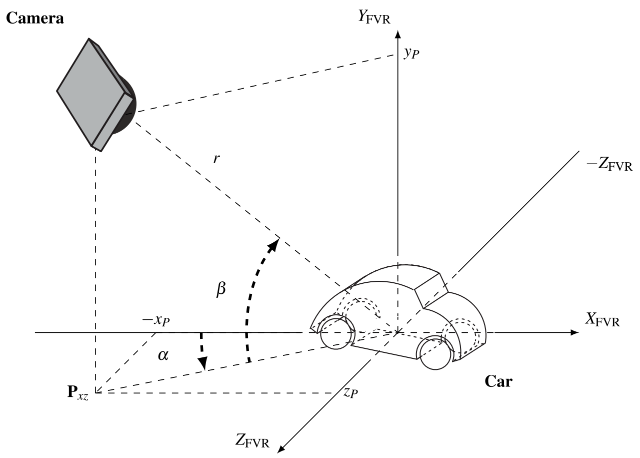


FIGURE 6. Drone trajectory around the car parked in its initial state during the frames acquisition phase.

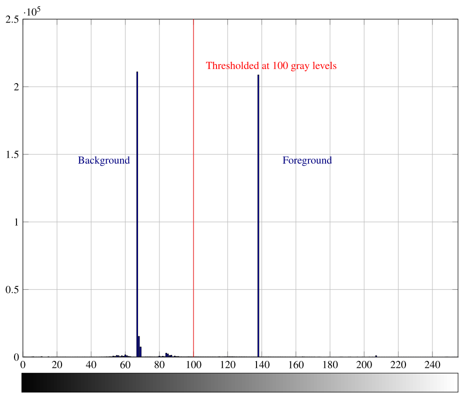


FIGURE 8. Histogram of the image data. The red line, i.e., the grayscale threshold, divides the graph into two parts: the *background* and the *foreground*. The gray gradient bars indicate the associated color to each x-value, from 0 to 255.

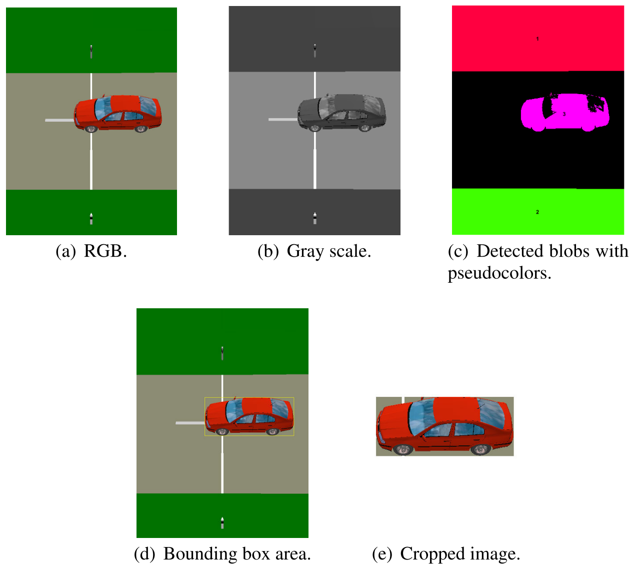


FIGURE 7. Frames obtained by the images segmentation process. From the RGB file format (a) to the cropped image (e) the steps are shown, sequentially.

B. BOUNDING BOX SELECTION

To train the classifier, the Region of Interest (ROI) of the target needs to be computed. Due to the high number of images, manual labeling tools, such as the MathWorks *Training Image Labeler*, cannot be used. Thus a Matlab script was developed to automatically select the bounding box area surrounding the target. The image segmentation process was used to simplify and to change the image representation: from RGB to grayscale (Figs. 7(a) and 7(b), respectively). The result is a set of contours that make the image meaningful and easier to analyze: each group of pixels in a region is similar w.r.t. some characteristics or computed properties, intensity, or texture, while adjacent regions are significantly different w.r.t. the same properties.

To automatically select each group of pixels, the Balanced Histogram Thresholding (BTH) method [49] was used. Such a method allows to separate the background from the foreground image by dividing the data into two main classes (see, Fig. 8) and by searching for the optimum threshold level.

Starting from the foreground grayscale image (see, Fig. 7(b)), the script deals with labeling the individual blobs by using the connected-component labeling algorithm [46] with a fixed heuristic (8-connected, in the considered case). In Figure 7(c) the obtained blobs are depicted with different colors and numbers for visual convenience. Then, the script selects the blob that meets the criteria in terms of size and intensity (chosen to match the target properties) in order to obtain a unique bounding box surrounding the target (see, Figs. 7(d) and 7(e)). Finally, the script provides as output a MAT-file containing, for each positive image, the suitable ROI components, i.e., the bounding box centroid, its width and height. This file is used for the classifier synthesis in the target detection design process (see, Fig. 2).

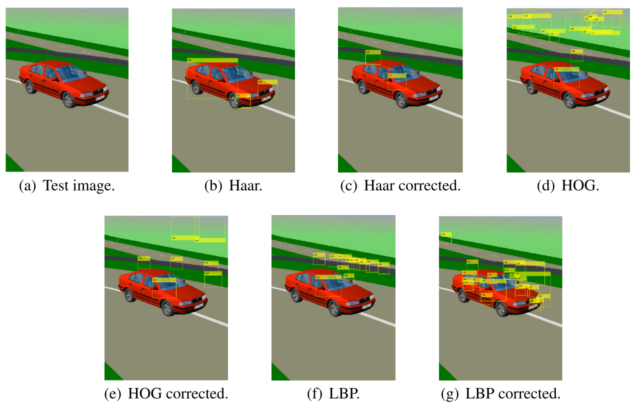
The proposed approach allows to automatically label the target (the car) from the positive images, thus decreasing the time spent for the training phase. In Figure 7, for a single sample frame, all elaboration steps are reported.

On the considered data set, the script was able to automatically detect the ROIs with an error of 8.18 %: the target was not recognized only in 215 frames out of 2626 positive images, and the first ROI loss appeared at the 1791th frame.

C. CLASSIFIER SYNTHESIS

The Viola & Jones algorithm [50] was chosen as object detection framework to recognize the car along the path. The algorithm was originally designed and developed for face detection problem, but it can be easily trained to detect any object [51] by using different features types (e.g., Haar, HOG, LBP) [42] and training stages.[5] Although even more complicated and performing object detection frameworks (e.g., YOLO [28], [29], Fast R-CNN [29], [30]) are available in the literature, historical reasons motivated this choice: the Viola

[5]This is common in cascade classifiers where each stage is an ensemble of weak learners, i.e., simple classifiers called decision stumps.



(a) Test image. (b) Haar. (c) Haar corrected. (d) HOG. (e) HOG corrected. (f) LBP. (g) LBP corrected.

FIGURE 9. Detection results obtained by using Haar, HOG and LBP feature types. The false alarm and true positive rates are 0.001 and 0.995, respectively, while the number of training stages is 4. Two different target models are considered: the "corrected" and "uncorrected" version.

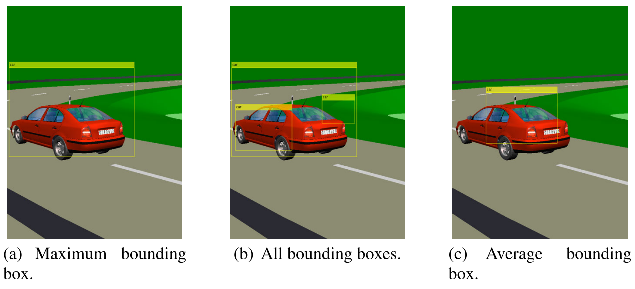
& Jones classifier was the first object detection framework in real-time. Thus, it is of interest, at least for educational purposes, to have a simulation platform that performs object detection with such a solution.

For the considered testbed, the Haar features were used to design the classifier. These features along with LBP are often used to detect faces due to their fine-scale textures while HOG features are often employed to detect objects. However, the obtained results suggested to choose Haar features which appeared more useful for capturing the overall shape of the target (see, Fig. 9) even if longer time was needed during the training phase.

D. PERFORMANCE COMPARISON

When designing a cascade object detector, the number of training stages, the false alarm and true positive rates, need to be tuned in accordance to the required performance (e.g., accuracy) and constraints (e.g., time response). To facilitate the analysis as well as to find the most suitable set of parameters that fit the problem, a Matlab script was developed to evaluate the performance of the classifier. This script is part of the proposed software platform (see, Fig. 2) and allows to compare in a few steps various configurations and models getting a general overview of how the object detector behaves.

In Fig. 9 the results obtained for a single sample frame are reported. Two different models were considered to prove the validity of the proposed approach: the *uncorrected* and *corrected* models. The first one uses the ROIs automatically extracted from the algorithm presented in Sec. III-B, while the second one employs those obtained using the Matlab tool *Training Image Labeler*. In all revelations, the car is only partially detected despite the large number of images employed to train the classifier. Except for some cases, there are no revelation errors: different bounding boxes are detected in the image. This is probably due to the absence of photorealism in the collected frames. As described in [52], the reality gap (i.e., realistic geometry, textures, lighting conditions, camera noise, and distortion) affects the performance of computer



(a) Maximum bounding box. (b) All bounding boxes. (c) Average bounding box.

FIGURE 10. Bounding box selection algorithm. The detection results are obtained by using the Haar cascade features type. The maximum (left) and average (right) bounding boxes are computed by using the result obtained from detection (center).

vision algorithms. On the other hand, they introduce enough "useful noise" to help the detection (the presence of several image view points).

Many tests have been conducted in order to assess the true performance of the classifier. As shown in Figs. 9(b) and 9(c), the detection results are very similar for both models. Thus, it is a good approximation to consider the "uncorrected" ROIs instead of the "corrected" version in the classifier design process. Such approximation allows to save time during the training phase thus avoiding to use specific tools, such as the Matlab tool *Training Image Labeler*, when the ROI detection fails. Moreover, it proves the validity and the effectiveness of the automatic tool procedure for bounding box selection. Of course, further tests may be carried out considering more valuable evaluation criteria, such as confusion matrix, accuracy, precision, recall, specificity [29].

IV. VISION-BASED TARGET DETECTION

The vision-based target detection phase sets up the IBVS problem using the classifier and tracking algorithms as feedback from the environment (see, Fig. 2). There are four components that constitute this part: the *vision target selector*, the *detection* and *tracking algorithm*, and the *distance vector computing*.

The vision target selector takes care of switching between the detection and the tracking algorithms based on the recognition results: the detector is used only at the first step or in case of partial occlusion, otherwise a CAMShift tracking algorithm [46] is employed to follow the car along the path. This algorithm performs target tracking by searching for its probability distribution pattern in a local adaptive size window. Although it does not guarantee the best performances, the algorithm supplies reliable and robust results [53].

Due to multiple target revelations (see, Sec. III-D), a Matlab script was used to obtain a unique bounding box surrounding the target (the car). The script computes the maximum (Fig. 10(a)) and the average (Fig. 10(c)) bounding boxes, as shown in Fig. 10. The maximum approach puts more trust in the detection results, while the average approach tries to filter out the revelation errors. The "good" choice depends on the particular employed classifier and on the amount of frames used during the training phase. For the considered

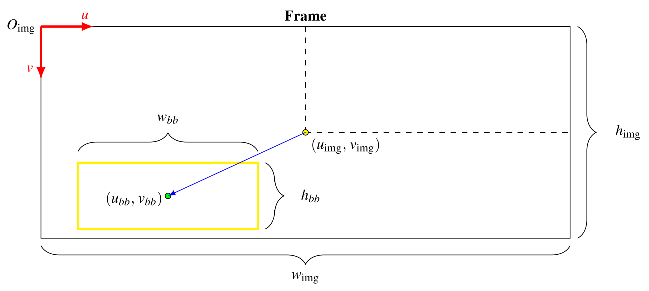


FIGURE 11. The diagram shows how frames are processed once the target is detected. The distance vector and the bounding box are represented in blue and yellow, respectively. The image (u_{img}, v_{img}) and bounding box (u_{bb}, v_{bb}) centroids are also reported.

testbed, the maximum bounding box was chosen to figure out the IBVS problem.

Once the target has been recognized, the distance vector computing block generates the references for the drone trajectory control (see, Sec. VI and Fig. 4) measuring the distance between the image (u_{img}, v_{img}) and bounding box (u_{bb}, v_{bb}) centroids, as depicted in Fig. 11. The vector aims to provide the reference signals (e_u and e_v) to move the drone so that the center of the bounding box surrounding the car overlaps the centroid of the image.

V. DRONE DYNAMICAL MODEL

For the specific case study, a quad-rotor in a plus configuration has been considered. The design of a high performance attitude and position controller requires often an accurate model of the system. It is here recalled the commonly used dynamical model of a quad-rotor [54] and, by following usual approaches, two orthonormal frames are introduced: the fixed-frame O_{FI} (where FI stands for Fixed Inertial), also called inertial (or reference) frame, and the body-frame O_{ABC} (where ABC stands for Aircraft Body Center) that is fixed in the aircraft center of mass and is oriented according to the aircraft orientation, see Fig. 12.

The translational dynamic equations of the aircraft can be expressed in the inertial frame as follows:

$$m\ddot{\boldsymbol{\xi}} = -mg\mathbf{E}_z + u_T\mathbf{R}(\varphi, \vartheta, \psi)\mathbf{E}_z, \tag{2}$$

where g denotes the gravity acceleration, m the mass, u_T the total thrust produced by the rotors, $\boldsymbol{\xi} = (x, y, z)^\top \in \mathbb{R}^3$ the drone position expressed in the inertial frame, $\mathbf{E}_z = (0, 0, 1)^\top$ is the unit vector along the Z-axis, while $\mathbf{R}(\varphi, \vartheta, \psi) \in \mathbb{R}^{3\times3}$ is the rotation matrix from the body to the inertial frame and it depends on the attitude $\boldsymbol{\eta} = (\varphi, \vartheta, \psi)^\top \in \mathbb{R}^3$ (i.e., Euler angles roll, pitch, and yaw, respectively) that describes the body-frame orientation according to the ZYX convention [55]. Furthermore, the rotational dynamics can be expressed as

$$\mathbf{I}\dot{\boldsymbol{\omega}}_B = -\boldsymbol{\omega}_B \times \mathbf{I}\boldsymbol{\omega}_B + \boldsymbol{\tau}, \tag{3}$$

where '$\times$' denotes the vector product, $\boldsymbol{\omega}_B = (\omega_x, \omega_y, \omega_z)^\top \in \mathbb{R}^3$ is the angular velocity vector expressed in the body-frame, $\mathbf{I} = \text{diag}(I_x, I_y, I_z) \in \mathbb{R}^{3\times3}$ is the inertia matrix of the vehicle

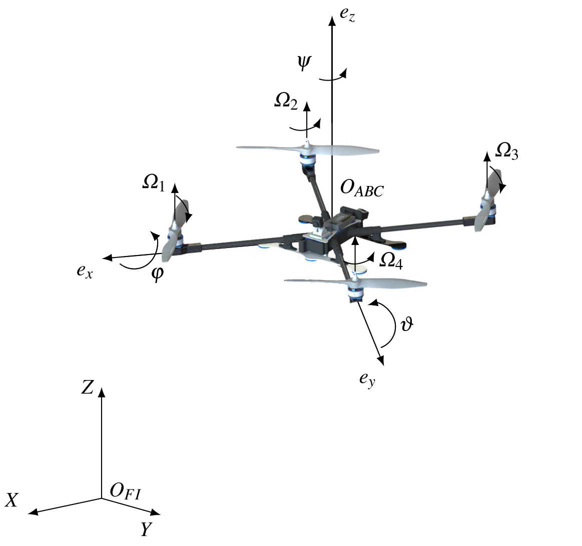


FIGURE 12. Drone in the body-frame (O_{ABC}) and the fixed-frame (O_{FI}) reference systems. Forces produced by each rotor, spin directions and propeller velocities, Ω_i, are also reported.

w.r.t. its principal axes, and $\boldsymbol{\tau} = (u_\varphi, u_\vartheta, u_\psi)^\top \in \mathbb{R}^3$ is the control torque vector obtained by actuating the rotors speeds according to the rotors configuration and the vehicle shape.

At low speeds and around the hovering state, the simplified dynamic model consists of six second order differential equations obtained from balancing forces and momenta acting on the drone, where $c_\bullet$ and $s_\bullet$ denote the cos($\bullet$) and sin($\bullet$) functions, respectively:

$$I_x\ddot{\varphi} = \dot{\vartheta}\dot{\psi}\left(I_y - I_z\right) + u_\varphi, \tag{4a}$$
$$I_y\ddot{\vartheta} = \dot{\varphi}\dot{\psi}\left(I_z - I_x\right) + u_\vartheta, \tag{4b}$$
$$I_z\ddot{\psi} = \dot{\vartheta}\dot{\varphi}\left(I_x - I_y\right) + u_\psi, \tag{4c}$$
$$m\ddot{x} = u_T\left(c_\varphi s_\vartheta c_\psi + s_\varphi s_\psi\right), \tag{5a}$$
$$m\ddot{y} = u_T\left(c_\varphi s_\vartheta s_\psi - s_\varphi c_\psi\right), \tag{5b}$$
$$m\ddot{z} = u_T c_\vartheta c_\varphi - mg, \tag{5c}$$

with

$$u_T = b_f\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right), \tag{6}$$

and

$$\begin{pmatrix} u_\varphi \\ u_\vartheta \\ u_\psi \end{pmatrix} = \frac{b_f}{b_m}\begin{pmatrix} b_m l\left(\Omega_4^2 - \Omega_2^2\right) \\ b_m l\left(\Omega_3^2 - \Omega_1^2\right) \\ -\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 \end{pmatrix}, \tag{7}$$

where Ω_i, $i \in \{1, 2, 3, 4\}$, are the actual rotors angular velocities expressed in rad s^{-1}, l is the distance from the propellers to the center of mass, while b_f and b_m are the thrust and drag factors, respectively. Further details can be found in [45], [54], [55]. Table 1 reports the parameters values of the drone for the considered case study (see, Sec. VI-C).

VI. FLIGHT CONTROL SYSTEM

Various state-of-the-art solutions investigate the trajectory tracking problem with quad-rotors. However, not all of them

TABLE 1. Drone parameter values for the considered case study.

	Sym.	Value	Unit
Mass	m	0.65	kg
Distance to center of gravity	l	0.23	m
Thrust factor	b_f	$7.5 \cdot 10^{-7}$	kg
Drag factor	b_m	$3.13 \cdot 10^{-5}$	kg m
Inertia component along e_x-axis	I_x	$7.5 \cdot 10^{-3}$	kg m^2
Inertia component along e_y-axis	I_y	$7.5 \cdot 10^{-3}$	kg m^2
Inertia component along e_z-axis	I_z	$1.3 \cdot 10^{-3}$	kg m^2

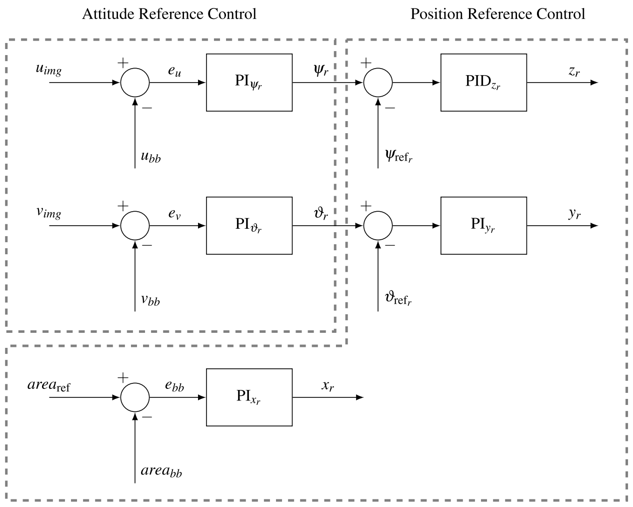


FIGURE 13. The reference generator scheme referred to virtual reference system (O_{FVR}). The obtained heuristic PID gains are: $K_{P_{\psi_r}} = 1 \cdot 10^{-5}$, $K_{I_{\psi_r}} = 1 \cdot 10^{-3}$, $K_{P_{\vartheta_r}} = 1 \cdot 10^{-5}$, $K_{I_{\vartheta_r}} = 1 \cdot 10^{-3}$, $K_{P_{x_r}} = 1 \cdot 10^{-6}$, $K_{I_{x_r}} = 6 \cdot 10^{-6}$, $K_{P_{y_r}} = 1 \cdot 10^{-2}$, $K_{I_{y_r}} = 1 \cdot 10^{-2}$, $K_{P_{z_r}} = 15$, $K_{I_{z_r}} = 57.5$ and $K_{D_{z_r}} = 3.75$.

are suitable for the specific case of application [56]. Therefore, with the aim of illustrating a control design methodology exploiting the IBVS approach, it has been considered the flight control system described in [45] and [57] that uses a *reference generator* and an *integral backstepping controller* to figure out the drone trajectory tracking problem. The reference generator extracts the information from the images to generate the path to follow, while the Integral Backstepping (IB) controller uses those references to compute the needed drone command signals. Figures 13 and 15 describe the overall control scheme.

A. REFERENCE GENERATOR

The reference generator is decomposed into two parts: the attitude and the position controller, both illustrated in Fig. 13. The attitude controller tunes the yaw (ψ_r) and the pitch (ϑ_r) angles trying to overlap the image (u_{img}, v_{img}) and the bounding box (u_{bb}, v_{bb}) centroids (see, Fig. 11), while the roll (φ_r) angle is computed by the IB controller.[6] These values are later used by the position controller to vary the drone reference position z_r and y_r, while x_r is computed comparing

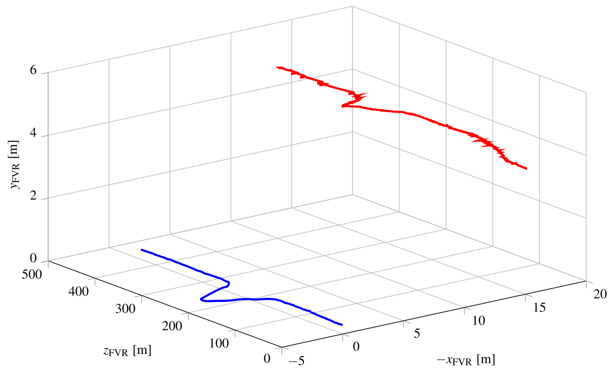


FIGURE 14. The car (in blue) and the output of the reference generator (in red) while tracking the target.

the detected area $area_{\mathrm{bb}}$ with those obtained when training the classifier $area_{\mathrm{ref}}$[7]

The proposed control architecture is based on control loops that are nothing but Proportional-Integral-Derivative (PID) controllers. These are a standard solution in the literature for quad-rotor control design [58]. For the considered case study, the vehicle starts flying 4 meters over the ground (Z-axis) with a distance of 15 meters from the car along the X-axis in the O_{FVR} reference system.

Figure 14 reports the trajectories followed by the car and the drone when running the simulation, while a further video has been made available at [59].

B. INTEGRAL BACKSTEPPING CONTROLLER

The integral backstepping of [45], [54] has been used as trajectory controller for the path tracking. It performs robustness against external disturbances (offered by backstepping) and sturdiness w.r.t. model uncertainties (given by the integral action). Starting from the outputs of the reference generator, the IB controller computes the orientation ($\varphi_{\mathrm{ref_{IB}}}$ and $\vartheta_{\mathrm{ref_{IB}}}$) that the drone should assume to follow the reference path (x_r and z_r). The $\varphi_{\mathrm{ref_{IB}}}$ and $\vartheta_{\mathrm{ref_{IB}}}$ reference angles are computed as:

$$\vartheta_{\mathrm{ref_{IB}}} = \frac{m}{u_T}\Bigg[\left(1 - c_1^2 + \lambda_1\right) e_x + (c_1 + c_2)\, e_{x\mathrm{IB}} + -c_1\lambda_1 \int_0^t e_x(\tau)\mathrm{d}\tau\Bigg], \tag{8a}$$

$$\varphi_{\mathrm{ref_{IB}}} = -\frac{m}{u_T}\Bigg[\left(1 - c_3^2 + \lambda_2\right) e_z + (c_3 + c_4)\, e_{z\mathrm{IB}} + -c_3\lambda_2 \int_0^t e_z(\tau)\mathrm{d}\tau\Bigg], \tag{9a}$$

with

$$e_{x\mathrm{IB}}(t) = \lambda_1 \int_0^t e_x(\tau)\mathrm{d}\tau + c_1\, e_x(t) + \dot{e}_x(t), \tag{10a}$$

$$e_{z\mathrm{IB}}(t) = \lambda_2 \int_0^t e_z(\tau)\mathrm{d}\tau + c_3 e_z(t) + \dot{e}_z(t), \tag{10b}$$

[6]All elaborations are expressed in the O_{FVR} reference system.

[7]This values is obtained as sample of mean of the collected ROIs while training the classifier (see, Sec. III-A).

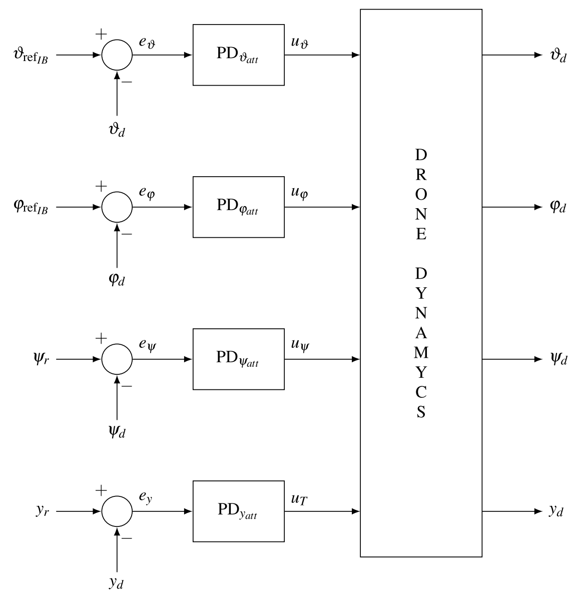


FIGURE 15. The drone trajectory controller. All variables are expressed in the virtual reference system (O_{FVR}). Here we recall the heuristic control gains employed into the simulation scenario: $K_{P_{y_{att}}} = 1000$, $K_{D_{y_{att}}} = 200$, $K_{P_{\varphi_{att}}} = 8$, $K_{D_{\varphi_{att}}} = 4$, $K_{P_{\vartheta_{att}}} = 12$, $K_{D_{\vartheta_{att}}} = 4$, $K_{P_{\psi_{att}}} = 10$, and $K_{D_{\psi_{att}}} = 4$.

and

$$e_x = x_r - x_d, \tag{11a}$$

$$e_z = z_r - z_d. \tag{11b}$$

For the considered motivating examples, the following values have been chosen: $\lambda_1 = 0.025, \lambda_2 = 0.025, c_1 = 2, c_2 = 0.5$, $c_3 = 2$ and $c_4 = 0.5$. Figure 15 shows the overall control system architecture.

C. NUMERICAL RESULTS

To prove the validity and effectiveness of the proposed framework, numerical simulations have been carried out by using the 2015b release of Matlab equipped with CVS and VR toolboxes.[8] The video available at [60] illustrates in a direct way how the system works, i.e., the ability of the quad-rotor to follow the car that moves along the nontrivial path. In addition, the video shows the behavior of the detection and tracking algorithms that never lose the target while tracking the target. Moreover, the video shows the capabilities of the control system in reacting to changes in the car's dynamics: during the double lane change maneuver the vehicle suddenly increases its speed and the aircraft tilts around the X-axis (the car seems to climb a hill[9]) to capture the shift in the dynamics.

A further scenario (the video is available at [61]) was considered to show how the simulation can be easily customized without the need to redesign the entire system. The numerical example aims to show how the performance of the detection and tracking algorithms can be easily evaluated while running alongside the drone tracking controller. Two cars are considered for the case of interest. One (red car) engages the ESP control while the other (yellow) switches off such control unit when changing the lane. As it can be seen from the video, the search window adapts its sizes in response to partial occlusions of the target.

Finally, the video at [62] shows the advantages of using a modular architecture for the platform. The video shows how the whole system architecture can be tested under various conditions simply changing the scenario.[10] As shown in the video, halfway through the simulation (26 s) increasing the speed of the car causes the drone to tilt excessively moving to instability.

The proposed scenarios demonstrate as the software platform allows to test the complex system while interacting with the surrounding environment and computer vision and control algorithms are in the loop.

VII. CONCLUSION

In this paper, a numerical simulation platform for multi-rotor aircraft based on Matlab and the MathWorks Virtual Reality and Computer Vision System toolboxes has been described. The platform makes easy to implement and to simulate complex scenarios where computer vision algorithms can be run and tested together with drone tracking controllers. The simulator provides a ready-to-use environment allowing students, researchers, and developers to easily test and evaluate their own algorithms. The platform also constitutes the first step towards the development of a more structured software tool where exploiting the advantages of software-in-the-loop simulations. The software has been released as open-source[3] making it possible to go through any part of the system. Future work includes the integration of the platform with more advanced robotics middleware and the creation of the interface with the hardware moving toward hardware-in-the-loop tests.

REFERENCES

[1] D. Scaramuzza *et al.*, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, pp. 26–40, 2014.

[2] S. Choi and E. Kim, "Image acquisition system for construction inspection based on small unmanned aerial vehicle," in *Advanced Multimedia and Ubiquitous Engineering* (Lecture Notes in Electrical Engineering), vol. 352. Springer, 2015, pp. 273–280. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-47487-7_40

[3] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4557–4564.

[4] D. Anthony, S. Elbaum, A. Lorenz, and C. Detweiler, "On crop height estimation with UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4805–4812.

[5] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 21–28.

[6] D. F. D. Castro and D. A. D. Santos, "A Software-in-the-Loop simulation scheme for position formation flight of multicopters," *J. Aerosp. Technol. Manage.*, vol. 8, no. 4, pp. 431–440, Oct. 2016.

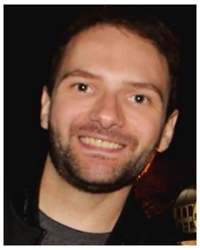
[8] The simulator is fully compatible with each further release of Matlab.

[9] The drone flies in an eye-in-hand configuration, i.e., tilts around the axis direct affects the camera orientation.

[10] The *vr_octavia* scenario was considered.

[7] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Toward domain independence for learning-based monocular depth estimation," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1778–1785, Jul. 2017.
[8] A. Tallavajhula and A. Kelly, "Construction and validation of a high fidelity simulator for a planar range sensor," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 6261–6266.
[9] R. Dianko and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robot. Inst., Pittsburgh, PA, USA, Tech. Rep. 79, 2008.
[10] G. Silano and L. Iannelli, "CrazyS: A software-in-the-loop simulation platform for the crazyflie 2.0 nano-Quadcopter," in *Robot Operating System (ROS), The Complete Reference*, vol. 4, A. Koubaa, Ed. Springer, 2020, pp. 81–115. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-20190-6_4
[11] S. Sinha, N. K. Goyal, and R. Mall, "Reliability and availability prediction of embedded systems based on environment modeling and simulation," *Simul. Model. Pract. Theory*, vol. 108, pp. 1–26, Dec. 2021.
[12] G. Silano, P. Oppido, and L. Iannelli, "Software-in-the-loop simulation for improving flight control system design: A quadrotor case study," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 466–471.
[13] A. Elkady and T. Sobh, "Robotics middleware: A comprehensive literature survey and attribute-based bibliography," *J. Robot.*, vol. 2012, pp. 1–15, May 2012.
[14] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2004, pp. 2149–2154.
[15] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1321–1326.
[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds. Springer, 2018, pp. 621–635. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-67361-5_40
[17] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: MORSE," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 46–51.
[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, May 2009, pp. 1–6.
[19] G. Metta, P. Fitzpatrick, and L. Natale, "Yarp: Yet another robot platform," *Int. J. Adv. Robotic Syst.*, vol. 3, no. 1, pp. 43–48, 2006.
[20] A. Mallet, S. Fleury, and H. Bruyninckx, "A specification of generic robotics software components: Future evolutions of G^{en}/sub o/M in the orocos context," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2002, pp. 2292–2297.
[21] S. Khan, M. H. Jaffery, A. Hanif, and M. R. Asif, "Teaching tool for a control systems laboratory using a quadrotor as a plant in MATLAB," *IEEE Trans. Educ.*, vol. 60, no. 4, pp. 249–256, Nov. 2017.
[22] M. A. Day, M. R. Clement, J. D. Russo, D. Davis, and T. H. Chung, "Multi-UAV software systems and simulation architecture," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2015, pp. 426–435.
[23] H. Shokry and M. Hinchey, "Model-based verification of embedded software," *Computer*, vol. 42, no. 4, pp. 53–59, Apr. 2009.
[24] J. Unicomb, L. Dantanarayana, J. Arukgoda, R. Ranasinghe, G. Dissanayake, and T. Furukawa, "Distance function based 6DOF localization for unmanned aerial vehicles in GPS denied environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5292–5297.
[25] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi, "6D physical interaction with a fully actuated aerial robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5190–5195.
[26] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 5420–5425.
[27] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.
[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[29] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019.
[30] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[31] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," 2020, *arXiv:2008.08050*. [Online]. Available: http://arxiv.org/abs/2008.08050
[32] E. P. de Freitas, L. A. L. F. da Costa, C. F. Emygdio de Melo, M. Basso, M. R. Vizzotto, M. S. C. Correa, and T. D. e Silva, "Design, implementation and validation of a multipurpose localization service for cooperative multi-UAV systems," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Sep. 2020, pp. 295–302.
[33] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. Suárez Fernández, and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *J. Intell. Robotic Syst.*, vol. 88, nos. 2–4, pp. 683–709, Dec. 2017.
[34] H. Lim, J. Park, D. Lee, and H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 33–45, Sep. 2012.
[35] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A modular gazebo MAV simulator framework," in *Robot Operating System (ROS): The Complete Reference*, vol. 1, A. Koubaa, Ed. Springer, 2016, pp. 595–625. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-26054-9_23
[36] F. Chaumette and S. Hutchinson, "Visual servo control. I. basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006.
[37] F. Chaumette and S. Hutchinson, "Visual servo control. II. Advanced approaches [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 109–118, Mar. 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4141039
[38] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.
[39] V. Lippiello, B. Siciliano, and L. Villani, "Eye-in-hand/eye-to-hand multi-camera visual servoing," in *Proc. 44th IEEE Conf. Decis. Control*, Dec. 2005, pp. 5354–5359.
[40] G. Silano and L. Iannelli, "An educational simulation platform for GPS-denied unmanned aerial vehicles aimed to the detection and tracking of moving objects," in *Proc. IEEE Conf. Control Appl. (CCA)*, Sep. 2016, pp. 1018–1023.
[41] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technol. J.*, to be published. [Online]. Available: http://www.cse.psu.edu/~rtc12/CSE598G/papers/camshift.pdf
[42] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. 9th Eur. Conf. Comput. Vis.*, L. Aleš, B. Horst, and A. Pinz, Eds. Berlin, Germany: Springer, 2006, pp. 430–443.
[43] S. Arefnezhad, A. Ghaffari, A. Khodayari, and S. Nosoudi, "Modeling of double lane change maneuver of vehicles," *Int. J. Automot. Technol.*, vol. 19, no. 2, pp. 271–279, Apr. 2018.
[44] MathWorks Inc. (2020). *IMU Sensor Fusion with Simulink*. [Online]. Available: https://www.mathworks.com/help/fusion/ug/imu-sensor-fusion-with-simulink.html
[45] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Towards autonomous indoor micro VTOL," *Auto. Robots*, vol. 18, no. 2, pp. 171–183, Mar. 2005.
[46] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011. [Online]. Available: https://link.springer.com/book/10.1007/978-3-642-20144-8
[47] The MathWorks Inc. *Virtual Reality Toolbox: For Use With MATLAB and Simulink*. MathWorks Official Website. Accessed: 2021. [Online]. Available: http://cda.psych.uiuc.edu/matlab_pdf/vr.pdf
[48] G. Silano. *Frames Acquisition Video*. YouTube. Accessed: 2021. [Online]. Available: https://youtu.be/A70zed84zv0
[49] A. dos Anjos and H. R. Shahbazkia, "Bi-level image thresholding—A fast method," in *Proc. 1st Int. Conf. Bio-Inspired Syst. Signal Process.*, vol. 2. Setúbal, Portugal: SciTePress, 2008, pp. 70–76.
[50] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. CVPR*, Dec. 2001, pp. 511–518.
[51] Y. Xu, G. Yu, X. Wu, Y. Wang, and Y. Ma, "An enhanced viola-jones vehicle detection method from unmanned aerial vehicles imagery," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1845–1856, Jul. 2017.
[52] P. Martinez-Gonzalez, S. Oprea, A. Garcia-Garcia, A. Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez, "UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation," *Virtual Reality*, vol. 24, no. 2, pp. 271–288, Jun. 2020.

[53] N. M. Artner and W. Burger, "A comparison of mean shift tracking methods," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, vol. 2008, pp. 197–204.
[54] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 2247–2252.
[55] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. Hoboken, NJ, USA: Wiley, 2015.
[56] T. P. Nascimento and M. Saska, "Position and attitude control of multirotor aerial vehicles: A survey," *Annu. Rev. Control*, vol. 48, pp. 129–146, Jan. 2019.
[57] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy, "Computer vision based general object following for GPS-denied multirotor unmanned vehicles," in *Proc. Amer. Control Conf.*, Jun. 2014, pp. 1886–1891.
[58] T. N. Dief and S. Yoshida, "Review: Modeling and classical controller of quadrotor," *Int. J. Comput. Sci. Inf. Technol. Secur.*, vol. 5, no. 4, pp. 314–319, 2015.
[59] G. Silano. *Vision-Based Target Tracking Scenario. The Drone Dynamics and the Control System are Neglected to Evaluate the Performance of the Reference Generator*. YouTube. Accessed: 2021. [Online]. Available: https://youtu.be/qAtndBIwdas
[60] G. Silano. *Vision-Based Target Tracking Scenario*. YouTube. Accessed: 2021. [Online]. Available: https://youtu.be/b8mTHRkRDmA
[61] G. Silano. *Vision-Based Target Tracking Scenario When the Target is Partially Covered*. YouTube. Accessed: 2021. [Online]. Available: https://youtu.be/RjXBtPqZZBc
[62] G. Silano. *Quad-Rotor Following a Car That Moves Along a Nontrivial Path in the Vr_octavia Scenario*. YouTube. Accessed: 2021. [Online]. Available: https://youtu.be/m43Zadq-6XM



GIUSEPPE SILANO (Student Member, IEEE) received the bachelor's degree in computer, the master's degrees in electronic engineering, and the Ph.D. degree in information engineering from the University of Sannio, Italy, in 2012, 2016, and 2020, respectively. From June 2020, he is with the Czech Technical University in Prague, where he holds a postdoctoral research fellowship position. From March to November 2019, he was a Visiting Student with LAAS–CNRS, Toulouse, France. His research interests include simulation and control, temporal logic, model predictive control, software-in-the-loop, and planning for micro aerial vehicles. He was among the finalists of the Aerial Robotics Control and Perception Challenge, the Industrial Challenge of the 26th Mediterranean Conference on Control and Automation (MED 2018), and among the participants of the LAAS Team selected as a finalist of the Mohammed Bin Zayed Robotics Competition (MBZIRC) 2020. He is a member of the IEEE Control System Society (CSS) and the IEEE Robotics and Automation Society (RAS).

LUIGI IANNELLI (Senior Member, IEEE) received the master's degree (Laurea) in computer engineering from the University of Sannio, Italy, in 1999, and the Ph.D. degree in information engineering from the University of Napoli Federico II, Italy, in 2003. After a postdoctoral position with the University of Napoli Federico II, he joined the University of Sannio as an Assistant Professor, and since 2016, he has been an Associate Professor of automatic control. He held a visiting researcher positions at the Royal Institute of Technology, Sweden, and the Johann Bernoulli Institute of Mathematics and Computer Science, University of Groningen, The Netherlands. His current research interests include analysis and control of switched and nonsmooth systems, stability analysis of piecewise-linear systems, smart grid control, and applications of control theory to power electronics and UAVs. He was a Co-Editor of the book *Dynamics and Control of Switched Electronic Systems* (Springer, 2012).