An educational simulation platform for GPS-denied Unmanned Aerial Vehicles aimed to the detection and tracking of moving objects

Giuseppe Silano and Luigi Iannelli

***Abstract*— The main motivation of this work is to show, for educational purposes, that the visual based object tracking problem can be illustrated through the simulation-in-the-loop approach: by using the MathWorks™ Virtual Reality Toolbox together with Matlab®, it is possible to simulate the behavior of a drone in a 3D environment when detection and control algorithms are run. Matlab VR is used due to the familiarity that students have with. In this way the attention can be moved to the classifier, the references generator and the trajectory tracking control. Each block is decoupled and independent, so it can be easily replaced with others thus simplifying the development phase.**

The virtual environment allows to test quickly the flight control system, comparing and evaluating different plans, both for indoor and outdoor scenarios. The system acquires frames from the virtual world, searches for one or more target on which it has been trained using machine learning techniques, and it extracts information about the pose in order to apply a trajectory control.

A simple case study has been presented in order to show the effectiveness of the approach.

***Index Terms*— Visual servoing, UAV, vision based control, GPS-denied, simulation-in-the-loop, educational.**

I. Introduction

During the last ten years, much effort has been put into the research field of (semi-)autonomous unmanned aerial vehicles (UAVs). Indeed, even though many algorithms for autonomous control and navigation exist, they are not usually suitable for making UAVs work autonomously in constrained and unknown environments (e.g., by using GPS, which is not available indoors). Thus, by considering the strong increase of the use of UAVs for inspection and surveillance purposes (e.g., in suburban scenarios) and for detecting and tracking arbitrary moving objects (e.g., for military operations), it follows the need for tools that allow to understand what it happens when some new applications are going to be developed. The problem is that UAV systems are expensive, not so easy to manage with (consider, for instance, the outdoor scenario) and, sometimes, they can be also dangerous in some way. For such reasons a complete software platform that makes possible to test different algorithms for UAV moving in a simulated 3D environment is more and more important for the whole design process, as well as for educational purposes. In this paper the objective will be to describe a software, based on the Matlab® computing environment, where virtual reality rendering and detection and control algorithms can be verified and validated. Due to the simple implementation and the limited possibility of interfacing with dedicated tools, the proposed platform has been meant as an educational tool, even though it can be considered also as a starting point for the development of a simulation-in-the-loop platform (see [1]) in the UAV field.

The specific domain of interest regards the autonomous behavior of UAV that act according to image based visual servoing [2]. Indeed, most of commercial applications using UAVs as devices for data and video streaming towards a central station, require the vehicles being remotely commanded by a supervisor (a human) that decides the next action to do by looking at the camera images. The camera extends the aircraft sensory capacity, and implements the feedback information that is closed in the loop through the (human) manual controller. An automatic control loop, instead, is implemented when the UAV is directly driven by an algorithm implementing the image-based visual servoing [3], [4]. According to the camera configuration, we can have two possibilities: the eye-in-hand (the camera is rigidly attached to the UAV), and the eye-to-hand (the camera has a fixed orientation in the workspace) [5]. For instance, many commercial Micro Aerial Vehicles (MAVs), such as the ARDrone 2.0 [6], adopt the eye-in-hand configuration. In this paper we will consider a similar case study with an eye-in-hand configuration.

We consider the UAV able to detect and track a specific object moving on the ground. Designing such kind of aircraft/ground robot system is not a simple task since many feedback control loops interact among them in order to control a highly nonlinear dynamic system. Thus, it helps a lot having a simple software platform able to perform simulation-in-the-loop of the overall system where the computer vision algorithms and the control laws are implemented in the same environment that will be used for the virtual world simulation. That allows the testing of visual servoing under many different scenarios easily defined, so to have an immediate and intuitive representation (3D animations) of what can be achieved with the real application of interest.

II. System Description

In order to simulate a scenario as much similar as to a real world, the Matlab Virtual Reality Toolbox has been used. The toolbox simulates a 3D world to observe the interaction between complex dynamic systems and the surrounding scenario. From such perspective the tool has been employed for simulating the interaction of a drone following a car. Given the particular scenario, we started from one of the

Giuseppe Silano and Luigi Iannelli are with the Department of Engineering, University of Sannio in Benevento, Piazza Roma, 21 - 82100 Benevento, Italy, email: {giuseppe.silano, luigi.iannelli}@unisannio.it

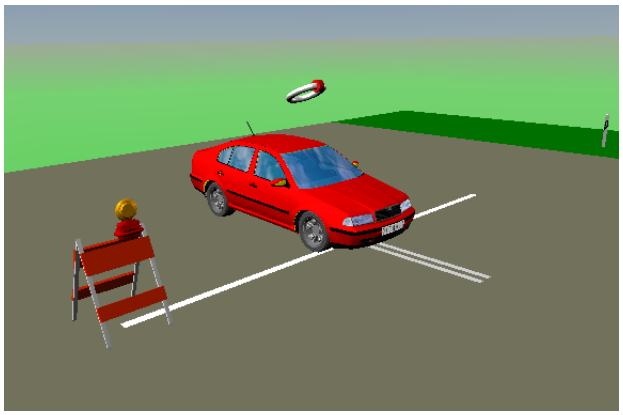

Fig. 1. Initial frame of the scenario considered in the paper.

examples available on the MathWorks platform (specifically the *vr_octavia_2cars* example) describing a quite detailed model of the car dynamics where the car moves along a non trivial path (see, Fig. 1).

We used an example already available from the standard installation only because our interest was in the UAV dynamics and control. We exploited the object oriented approach so to minimize the development time for real applications.

Then, it has been added an external observer that represented the UAV. In Matlab VR the external observer has six degrees of freedom: the spatial coordinates x, y, z, and the angles *yaw*, *pitch* and *roll*. In this way it is possible to simulate the behavior of a drone that moves in the virtual environment and observes the car moving along the path. The whole process is the following: images are updated according to the position and the orientation of the Matlab VR external observer (the UAV) with respect to the car; such images are acquired and elaborated for getting information necessary in order to detect the object (the car) and to run the control strategy designed for tracking the moving object. The output of the control algorithm consists into the actuation commands that should be given to the observer (the drone) in order to update its position and orientation. Here a further low level feedback control loop is used. The overall scheme is depicted in Fig. 2 with references to sections describing the specific parts. The Simulink scheme in Fig. 3 models the virtual world behavior. The car dynamics are simulated to follow a given path as given in the block *esp_on* in the scheme (the orginal Matlab example was aimed to show the ESP effects on the car dynamics). At each time step

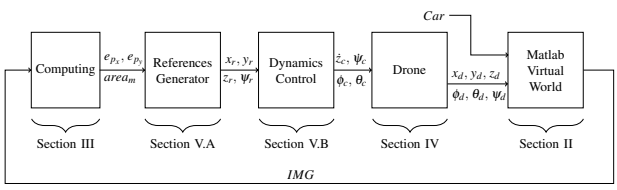

Fig. 2. The control scheme. Subscript c indicates the commands, r indicate references and d indicate the drone.

the steering angle, the brake system intensity, the car linear velocity and position are computed. The data processed are sent to the *VR Visualization* block that takes care of the car movements in the virtual environment. The *observer_position* block represents the aircraft position (x_d, y_d and z_d) in the scenario. The *rotation_matrix* block represents the direction cosine matrix [15] used to obtain the necessary quaternion to define the drone attitude in the virtual reality world (here we used the Rodrigues's formula [7]). Both blocks are updated by the regulator at each frame (time step) during the entire simulation.

Note that Matlab VR adopts a reference system slightly different from the classic fixed reference frame O_{FI} (see Sect. IV). Figure 4 illustrates such difference: in particular, axes are differently oriented and, furthermore, the virtual world reference system is centered in the car center of gravity, although the axes orientation is fixed. Those differences are taken into account in all elaborations.

Finally, the scheme saves the current car position (x_{car}, y_{car} and z_{car}), later on used for comparing the drone and car trajectories, and frames of the virtual scenario observed from the drone point of view. Those frames will be used, as described in next sections, for pattern recognition.

III. Vision Based Target Detection

The Viola & Jones algorithm [8] has been used for recognizing the car along the path in order to obtain a fast detection. Such algorithm, despite requiring more computational time than other algorithms [9] (e.g., SIFT and SURF algorithms [10], [11]), it ensured the best performance for the considered case study. During the classifier phase, much time has been dedicated to the learning process. In this way we were able to detect the target when it was in different positions, too.

A. Classifier learning phase

A high number of images are needed in order to train the classifier. The images are divided into two groups: positive (that contain the target) and negative images. By following as suggested in [12], we used a minimum number of 1000 positive images and at least 2000 negative images.

We wrote a Matlab script to minimize and automate the frames acquisition and the computing phase. To this aim we simulated the drone moving along a spiral trajectory around the car parked in its initial state. The aircraft attitude and position have been computed at each frame so that the observer point described a trajectory along the sphere surface as shown in Fig. 5.

Finally, the area bounding the car, also known as region of interest (ROI), was selected using the Matlab tool *Training Image Labeler*.

Figure 6 shows the detection results obtained using the Haar cascade [9] and histogram of oriented gradients (HOG) [13] features type. The car is only partially detected in spite of the high images number used in the learning process. Although there are no revelation errors, different bounding boxes have been detected in the image. That is

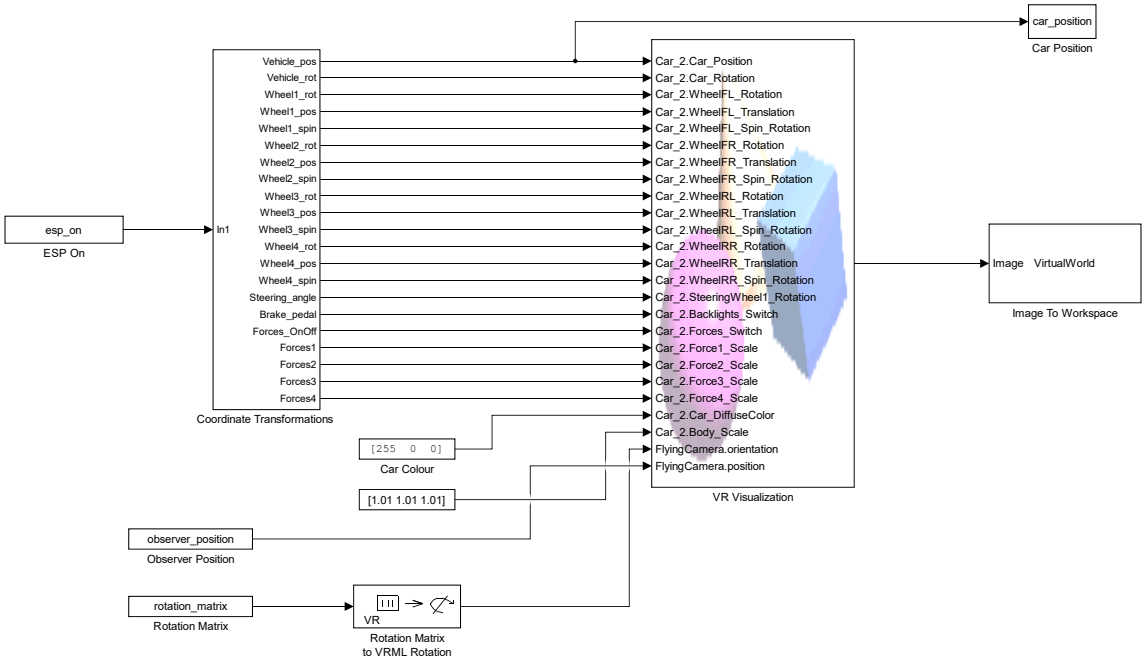

Fig. 3. Virtual world Simulink scheme.

probably caused by the several image view points used in the learning process. On the other hand, they introduce enough "useful noise" to help the detection. Many tests have been conducted in order to assess the true performance of the algorithm. A video showing the results has been made available [14].

B. Bounding box selection

The recognition of different bounding boxes requires an algorithm in order to obtain a unique box surrounding the target. A Matlab script computes the maximum and the average bounding box, as shown in Fig. 7. The maximum approach put more trust in the detection results while the average approach tries to filter out the detection results. The 'good' choice depends on the particular used classifier. In our case study the maximum bounding box has been chosen.

After that, the image (x_{img}, y_{img}) and the bounding box (x_{bb}, y_{bb}) centroids are computed, as well as the distance vector between the centroids, so as described in [6] and illustrated in Fig. 8.

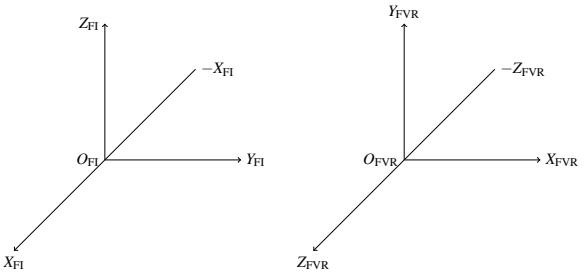

Fig. 4. The picture illustrates the classic fixed frame O_{FI} (left) and the corresponding virtual fixed O_{FVR} (right) reference system.

IV. Model of a hex-rotor drone

In our study case we considered a drone with six rotors. The development of a suitable attitude and position controller for the hex-copter required an accurate dynamical model to be derived. Classical Newtonian and Lagrange modeling methods [15] can be applied. In our case we used the Newtonian approach, the extensively used choice for modeling traditional helicopters [16]. We introduced two reference systems: the fixed-frame O_{FI} (where FI stand for fixed inertial), also called inertial frame, and the body-frame O_{ABC} (where ABC stands for Aircraft Body Center) that is fixed in the aircraft center of gravity and oriented according to the aircraft attitude, see Fig. 9.

The resulting model consists of six equations for the system dynamics (eqs. (1)) and four equations describing the inputs to the system (eqs. (3)) with $c.$ and $s.$ denoting $\cos(\cdot)$ and $\sin(\cdot)$ functions, respectively.

$$
\begin{cases}
J_{xx}\,\ddot{\phi}_d = \dot{\theta}_d\,\dot{\psi}_d\,(J_{yy}-J_{zz}) - J_r\,\dot{\theta}_d\,\Omega_r - K_{fax}\,\dot{\phi}_d^2 + U_2 \\
J_{yy}\,\ddot{\theta}_d = \dot{\phi}_d\,\dot{\psi}_d\,(J_{zz}-J_{xx}) + J_r\,\dot{\phi}_d\,\Omega_r - K_{fay}\,\dot{\theta}_d^2 + U_3 \\
J_{zz}\,\ddot{\psi}_d = \dot{\theta}_d\,\dot{\phi}_d\,(J_{xx}-J_{yy}) - K_{faz}\,\dot{\psi}_d^2 + U_4 \\
m\ddot{x}_d = -k_{ftx}\dot{x}_d + \left(c_{\phi_d}\,c_{\psi_d}\,s_{\theta_d} + s_{\phi_d}\,s_{\psi_d}\right)U_1 \\
m\ddot{y}_d = -k_{fty}\dot{y}_d + \left(c_{\phi_d}\,s_{\theta_d}\,s_{\psi_d} - s_{\phi_d}\,c_{\psi_d}\right)U_1 \\
m\ddot{z}_d = -k_{ftz}\dot{z}_d - g + c_{\phi_d}\,c_{\theta_d}\,U_1
\end{cases}
\tag{1}
$$

The first three equations describe the angular accelerations of the aircraft while the remaining three equations describe the UAV linear acceleration in the direction of x, y and z, respectively. The parameter J_r is the inertia for each rotor while Ω_r (eq. (2)) is the overall propeller speed. J_{xx}, J_{yy}, J_{zz}, K_{fax}, K_{fay}, K_{faz}, K_{ftx}, K_{fty} and K_{ftz} are the inertial components, the propeller drag coefficients and the aerodynamic force constants about the x-axis, y-axis and z-axis, respectively. The total mass of the hex-copter is m and

$$\Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 - \Omega_5 + \Omega_6. \tag{2}$$

The system inputs are reported in eqs. (3), where U_1 represents the throttle command (that modifies $\dot{z}_c$), U_2 represents the roll angle command (that modifies $\dot{\phi}_c$), U_3 is the pitch angle command (that modifies $\dot{\theta}_c$), while U_4 represents the yaw rate of change command (that modifies $\dot{\psi}_c$).

$$U_1 = b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2 + \Omega_5^2 + \Omega_6^2\right) \tag{3a}$$

$$U_2 = bl\left[-\Omega_2^2 + \Omega_5^2 + \frac{1}{2}\left(-\Omega_1^2 - \Omega_3^2 + \Omega_4^2 + \Omega_6^2\right)\right] \tag{3b}$$

$$U_3 = \frac{bl\sqrt{3}}{2}\left(-\Omega_1^2 + \Omega_3^2 + \Omega_4^2 - \Omega_6^2\right) \tag{3c}$$

$$U_4 = d\left(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 - \Omega_5^2 + \Omega_6^2\right) \tag{3d}$$

Finally, l is the distance to the center of gravity while b and d are the thrust and drag factor in hovering, respectively. Increasing or decreasing the speed of the propellers together will determine the altitude change in position and velocity, while varying the speed of two propellers (Ω_1 and Ω_6 or Ω_3 and Ω_4) will cause the aircraft to tilt about the y-axis which is denoted as pitch angle θ. Similarly varying the speed of the three propellers (Ω_1, Ω_2 and Ω_3, or Ω_4, Ω_5 and Ω_6) will cause the aircraft to tilt about the x-axis which is denoted as roll angle ϕ. Finally, the vector sum of the reaction moment produced by the rotation of Ω_1, Ω_3 and Ω_5 and the reaction moment produced by the rotation of Ω_2, Ω_4 and Ω_6 will cause the hex-rotor to spin about its axis (z-axis) which is denoted as yaw angle ψ. These are the six system DOFs consisting of the position x_d, y_d and z_d and the orientation ψ_d, ϕ_d and θ_d. Further details on the model are reported in [15], together with the parameters values, as well.

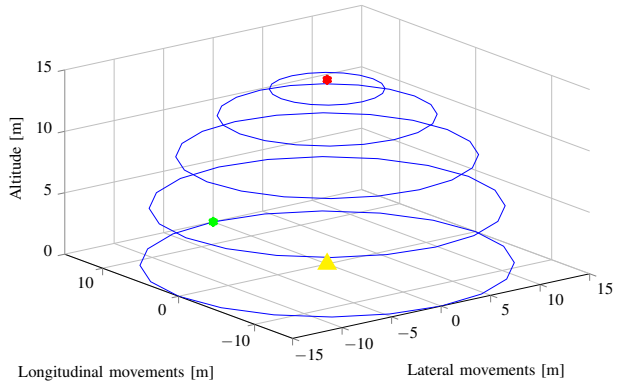

Fig. 5. Drone trajectory around the car during frame acquisition. In red the drone initial position and in green its final position. The car is indicated by the yellow point.

V. Flight Control System

The flight control system has been split into parts: a references generator, that uses the information extracted from the images to generate the path to follow, and the integral backstepping (IB) controller described in [17]. Figure 2 describes the whole control scheme.

A. Reference generator

The references generator is decomposed into two parts: the attitude and the position controller, both illustrated in Fig. 10. In order to minimize the distance vector, the attitude controller tunes the yaw (ψ_r) and pitch (θ_r) angles, while roll (ϕ_r) remain equal to zero, trying to overlap the image (x_{img}, y_{img}) and the bounding box (x_{bb}, y_{bb}) centroids. The angles are later used to tune the reference position x_r, y_r, z_r with respect to the virtual world reference (see, Fig. 4).

The references generator takes into account the camera initial position and its movements in the virtual environment.

The PI_{ψ_r} and PI_{θ_r} outputs are the variations $\Delta\psi_r$ and $\Delta\theta_r$ of ψ_r and θ_r that are used by the IB controller as path reference (see Fig. 12) and to compute x_r (see Fig. 10), respectively. Similarly it happens for the variables y_r and x_r, that need the aircraft initial positions y_{init} and x_{init}. The values ψ_{ref_r} and θ_{ref_r}, both equal to zero, are the attitude references that the UAV assumes during the target tracking. Finally, the error signal e_{surface} is given by the difference between the bounding box area ($w_{bb} \cdot h_{bb}$, aka $area_{\text{mes}}$) and the reference area ($area_{\text{ref}}$) given by the sample mean of the images uses during the learning process. It is used in order to tune the distance z_r.

Figure 11 reports the trajectories followed by the car and the UAV while a further video [19] has been made available for showing the results of the proposed approach.

B. Integral Backstepping controller

The integral backstepping of [18] has been used as the controller for the trajectory the path tracking. It combines the

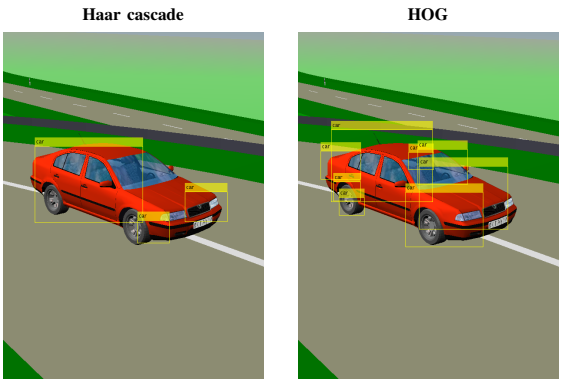

Fig. 6. Detection results obtained using the Haar cascade (left) and histogram of oriented gradients (right) features type.

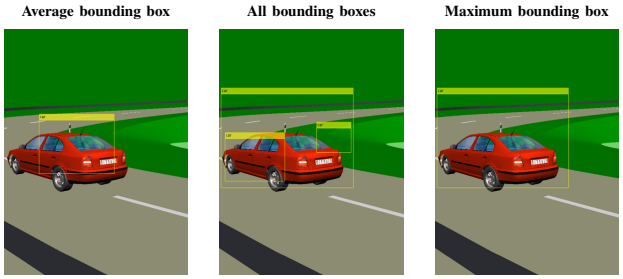

Fig. 7. Bounding box selection algorithm.

robustness against disturbances offered by backstepping and robustness against model uncertainties offered by the integral action. The scheme in Fig. 12 summaries as the controller system works.

The trajectory control strategy works for making the aircraft attitude $\left(\phi_{\text{ref}_{IB}}\right.$ and $\left.\theta_{\text{ref}_{IB}}\right)$ to follow the reference generator outputs (z_r and x_r). The reference signals $\phi_{\text{ref}_{IB}}$ and $\theta_{\text{ref}_{IB}}$ are computed (see Fig. 12) as:

$$\theta_{\text{ref}_{IB}} = \frac{m}{U_1}\Big[\left(1-c_1^2+\lambda_1\right)e_x + \left(c_1+c_2\right)e_{x_{IB}} - c_1\lambda_1\int_0^t e_x(\tau)\mathrm{d}\tau\Big] \tag{4a}$$

$$\phi_{\text{ref}_{IB}} = -\frac{m}{U_1}\Big[\left(1-c_3^2+\lambda_2\right)e_z + \left(c_3+c_4\right)e_{z_{IB}} - c_3\lambda_2\int_0^t e_z(\tau)\mathrm{d}\tau\Big], \tag{4b}$$

with

$$e_{x_{IB}}(t) = \lambda_1\int_0^t e_x(\tau)\mathrm{d}\tau + c_1 e_x(t) + \dot{e}_x(t) \tag{5a}$$

$$e_{z_{IB}}(t) = \lambda_2\int_0^t e_z(\tau)\mathrm{d}\tau + c_3 e_z(t) + \dot{e}_z(t), \tag{5b}$$

and

$$e_x = x_r - x_d \tag{6a}$$

$$e_z = z_r - z_d, \tag{6b}$$

where (c_1, c_2, c_3, c_4, λ_1 and λ_2) are positive constants.

The following numerical values have been chosen: $\lambda_1 = 1$, $\lambda_2 = 1$, $c_1 = 2$, $c_2 = 0.5$, $c_3 = 2$ and $c_4 = 0.5$.

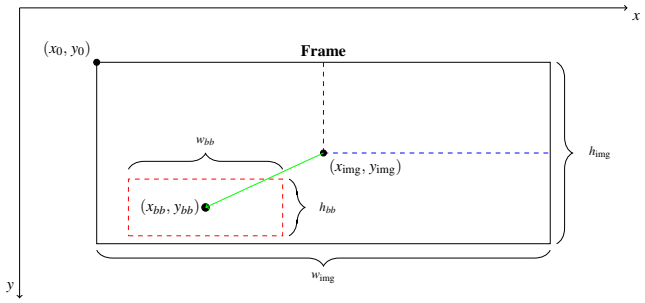

Fig. 8. The scheme illustrates the information extracted by the frames. Each frame is 800×600 pixels.

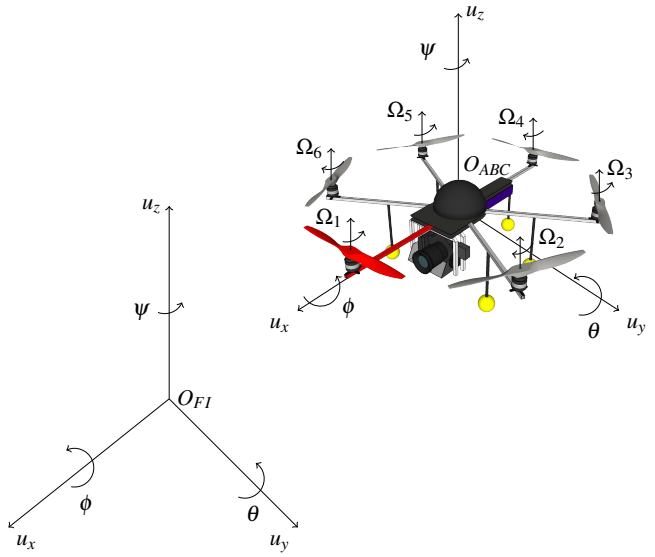

Fig. 9. Drone in the body-frame (O_{ABC}) and the fixed-frame (O_{FI}) reference system.

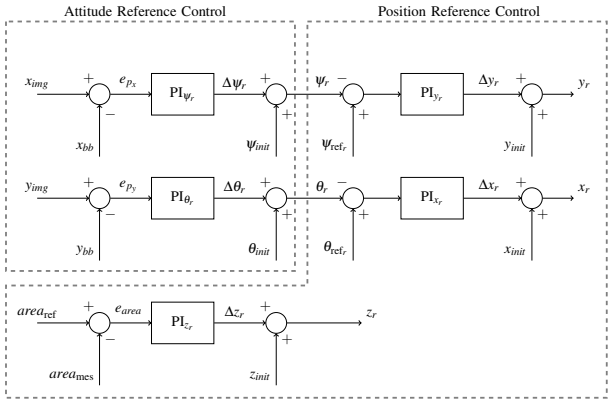

Fig. 10. The reference generator scheme. The obtained heuristic PD gains are: $K_{P,\psi_r} = 1\cdot10^{-5}$, $K_{I,\psi_r} = 1\cdot10^{-3}$, $K_{P,\theta_r} = 1\cdot10^{-5}$, $K_{I,\theta_r} = 1\cdot10^{-3}$, $K_{P,y_r} = 1$, $K_{I,y_r} = 2$, $K_{P,x_r} = 1.5$, $K_{I,x_r} = 40$, $K_{P,z_r} = 1\cdot10^{-6}$ and $K_{I,z_r} = 1\cdot10^{-5}$.

C. Numerical results

The overall system has been simulated in Matlab and the results illustrate in a direct way (you can find the video [20]) how the system performs. In the middle of the simulation (at 13.08 s) the car speed becomes much higher than the drone speed causing an excessive UAV rolling. Due to the eye-in-hand configuration, the target comes out of the camera view and it is lost.

Those results demonstrated as the system works and the limit of the eye-in-hand configuration, as well. Anyhow, the software platform allowed to test the complex system composed by computer vision and control algorithms interacting among them and with the moving objects dynamics.

VI. Conclusion

In this work a well-known computing environment (Matlab) has been used implement the simulation of a complex

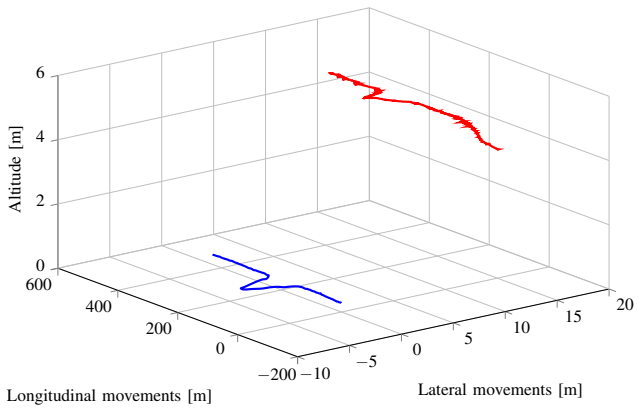

Fig. 11. The car (in blue) and the aircraft (in red) path described during the target following.

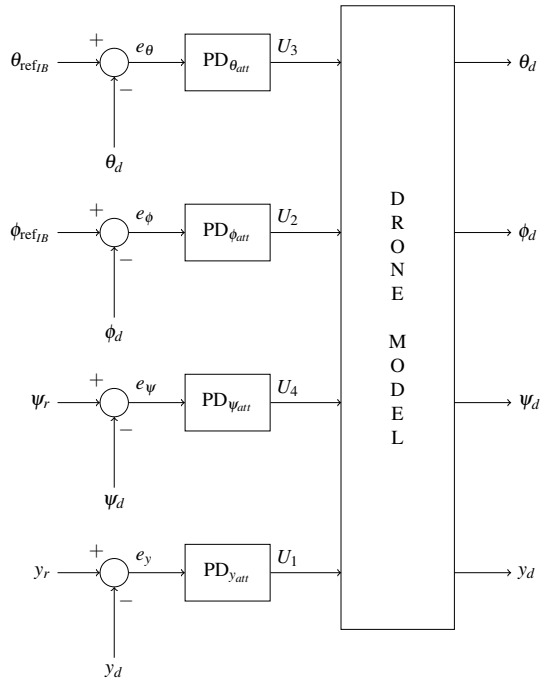

Fig. 12. Attitude control scheme using PID technique. U_1, U_2, U_3 and U_4 are the physical drone inputs: throttle ($\dot{z}_c$), roll (ϕ_c), pitch (θ_c), and yaw rate ($\dot{\psi}_c$), respectively. For the simulation we used the gains $K_{P_{y,att}} = 100$, $K_{D_{y,att}} = 20$, $K_{P_{\phi,att}} = 0.8$, $K_{D_{\phi,att}} = 0.4$, $K_{P_{\theta,att}} = 1.2$, $K_{D_{\theta,att}} = 0.4$, $K_{P_{\psi,att}} = 1$ e $K_{D_{\psi,att}} = 0.4$.

virtual world, so to show the effects of computer vision and control strategies aimed to detect and track moving objects. In this way it has been proven the effectiveness of the approach for educational purposes, so that interested students might work in a known environment by developing their own algorithms in a easy way. Nevertheless, in our opinion the work could constitute the first step towards the development of a more structured platform aimed for the simulation-in-the-loop of such kind of applications. The idea is to rely on more flexible and dedicated solution like V-REP [21] or Gazebo [22], [23]. In particular, the latter one will provide also the advantages of an open source solution.

REFERENCES

[1] M. A. Day, M. R. Clement, J. D. Russo, D. Davis, and T. H. Chung, Multi-UAV software systems and simulation architecture, *2015 International Conference on Unmanned Aircraft Systems*, pp. 426435, 2015.

[2] A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[3] K. Hashimoto M. Ishikawa, A. Namik. Hierarchical control architecture for high-speed visual servoing. *SAGE Journals*, vol. 2, no. 1, pp. 873-888, 10 2003.

[4] A.P. Del Pobil, M. Prats and P.J. Sanz. Model-based tracking and hybrid force vsion control for the uji librarian robot. In *IEEE Intelligent Robots and System 2005 (IROS 2005)*, pp. 1090-1095, 2005.

[5] A. Muis and K. Ohnishi, Eye-to-hand approach on eye-in-hand configuration within real-time visual servoing, In *IEEE/ASME Transactions on Mechatronics*, vol. 10, no. 4, pp. 404-410, 2005.

[6] J. Pestana and J. L. Sanchez-Lopez and S. Saripalli and P. Campoy, Computer vision based general object following for GPS-denied multirotor unmanned vehicles, In *American Control Conference 2014*, pp. 1886–1891, 2014.

[7] R. M. Murray, Z. Li and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.

[8] P. Viola M. Jones. Rapid object detection using boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition*, pp. I-511–I-518, 2001.

[9] Suwan Tongphu, Naddao Thongsak, and MatthewN. Dailey. Rapid detection of many object instances. In Jacques Blanc-Talon, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, volume 5807 of Lecture Notes in Computer Science, pp. 434-444. Springer Berlin Heidelberg, 2009.

[10] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, 2004.

[12] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In Bernd Michaelis and Gerald Krell, editors, *Pattern Recognition*, volume 2781 of Lecture Notes in Computer Science, pp. 297-304. Springer Berlin Heidelberg, 2003.

[13] S. J. D. *Prince. Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[14] G. Silano, Bounding box selection video, *YouTube*, `https://youtu.be/f_8ZHFlqhac`.

[15] Mostafa Moussid, Adil Sayouti, Hicham Medromi, Dynamic Modeling and Control of a HexaRotor using Linear and Nonlinear Methods, In *International Journal of Applied Information Systems (IJAIS)*, Volume 9, No. 5, 2015.

[16] K. P. Valavanis, Advances in Unmanned Aerial Vehicles vol. 33. Florida: Springer, 2007.

[17] S. Bouabdallah and R. Siegwart, Full control of a quadrotor, In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 153–158, 2007.

[18] S. Bouabdallah, Design and control of quadrotors with application to autonomous flying, PhD Thesis, EPFL, 2006.

[19] G. Silano, Reference generator video, *YouTube*, `https://youtu.be/p3XrPyXtXZk`.

[20] G. Silano, Final video, *YouTube*, `https://youtu.be/oVvUgkbLIto`.

[21] Singh S.P.N. Rohmer, E and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326. 2013.

[22] C.E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J.L. Rivero, J. Manzo, E. Krotkov, and G. Pratt. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494506, April 2015.

[23] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149-2154, Sendai, Japan, 9 2004.