Vision Based Fuzzy Control Autonomous Landing with UAVs: From V-REP to Real Experiments

Miguel A. Olivares-Mendez* and Somasundar Kannan* and Holger Voos*

***Abstract*— This paper is focused on the design of a vision based control approach for the autonomous landing task of Vertical Take-off and Landing (VTOL) Unmanned Aerial Vehicles (UAVs). Here is presented the setup of a simulated environment developed in V-REP connected to ROS, and its uses for tuning a vision based control approach. In this work, a Fuzzy control approach was proposed to command the UAV's vertical, longitudinal, lateral and orientation velocities. The UAV's pose estimation was done based on a vision algorithm and the knowledge of the landing target. Real experiments with a quadrotor landing in a moving platform are also presented.**

***Index Terms*— Unmanned aerial vehicles, fuzzy control, vision based control, autonomous navigation.**

I. INTRODUCTION

Simulation environments are always an important design tool in research, specially in the specific field of robotics. First we will discuss a few relevant tools tools with the advantages and disadvantages. The Webots [1] is one of the most sought about, but free versions are unavailable. The Matlab based Robotics Toolbox by Peter Corke [2] is recommendable to understand many robotics control problems and its vision based on outer control loop. The disadvantage of this approach is that there is no direct way to use what is implemented in the toolbox with a real robot. This is something that was solved with ROS [3]. The Robot Operative System provides an open source framework to develop packages to interactive with real sensors, actuators, robots, etc, using a publisher/subscriber system for the communication between them. The main advantage of this pseudo-operative system, is that is easy to use, to develop new packages and to communicate with existing packages. This framework comes with a 3D simulation environment called Gazebo 3D [4]. The main advantage of this software is that all the packages (algorithms, control, etc) used in the virtual world of Gazebo, can be used with minor changes in the real version of the simulated robot. This fact implies an enormous reduction of time in the software implementation part of any research. The disadvantage of the Gazebo simulator is that it requires heavy computational capabilities. An alternative environment for 3D simulation is the software developed by Coppelia Robotics, the Virtual Robotics Experimentation Platform (V-REP) [5]. Compared to Gazebo, this software can be installed and run without a powerful graphic card and does not required a powerful CPU. The V-REP comes with a large number of robots, sensors and actuators models, and several structures to create a virtual world. It also allows us to interact with the virtual environment during the simulation running time. An important advantage of this software is the bridge with ROS [6], allowing to use everything developed in the previously mentioned framework. All these characteristics make the V-REP and the connection with ROS the ideal platform to learn, teach, perform research and developed with robots.

This work focuses on the vision based control system of a quadrotor in the simulated environment to be used later with a real aircraft. A specific task of autonomous landing on a moving target with a quadrotor has been defined to test the V-REP and ROS connection. There are many visual servoing applications present in the literature. Different vision-based algorithms have been used to follow a car from a UAV [7], [8],[9]. Visual terrain following (TF) methods have been developed for a Vertical Take Of and Landing (VTOL) UAV [10]. In [11] a description of a vision-based algorithm to follow and land on a moving platform and other related tasks are proposed. A cooperative strategy has been presented in [12] for multiple UAVs to pursuit a moving target in an adversarial environment. The low-altitude road following problem for UAV using computer vision technology was addressed in [13]. People following method with Parallel Tracking and Mapping (PTAM) algorithm has been developed in [14]. Contrary to the above discussed research, the autonomous landing approach presented in this work is based on the control of the lateral, longitudinal,, vertical, and heading velocities of the quadrotor to modify its position to land on a predefined platform.

The work presented in this paper is based on the authors' previous works [15], [16]. In the first of these previous works the authors shows the tuning process of the fuzzy control system and the tests done in the simulation environment of the autonomous landing on a static and moving platform. Here in this paper the authors extend this work with real experiments. In the second of these works the authors present a tuning process of the fuzzy control system, simulation tests and real experiments of a tracking process of different static objects on a wall. Here in this work the authors extend this work by the tracking of a moving object, as well as the autonomous landing on a static and moving platforms.

The outline of this paper is structured in the following way. Section II presents the simulation environment. Section III explains the vision algorithm and fuzzy control approach. In Section IV we discuss the V-REP simulation and then the experiments done with a quadrotor and a ground vehicle. Section V will discuss the conclusions and the future work.

*Automation Research Group. Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. 6, Rue Coudenhove-Kalergi, L-1359, Luxembourg miguel.olivaresmendez@uni.lu

II. Simulation Environment

Here we briefly discuss the simulation environment of V-REP. The V-REP presents an easy and intuitive environment to create your own virtual system and include any of the robots that are provided, as well as objects, structures, actuators and sensors. It also allows to create your own robot by adding actuators, joints, sensors and basic forms. An example of a V-REP environment is shown in Figure 1. On the left side of the environment there is a list of robots, sensors, actuators, structures, etc that could be easily include in the simulation scene by drag and drop (model browser). In the next column, there is the scene hierarchy, where all the robots, sensors, graphs and structures of the current scene are represented. A script based on *LUA* script could be associated to each sensor and robot to interact with them, inside the V-REP environment of from the outside (e.g. C++ code or from a ROS package). The central window could be divided in one or more views, we divided it in four views, two external cameras (top and back), and the representation of the velocities and the position of the UAV. More detailed information about this robotics experimental platform is found in the V-REP official site. The V-REP comes with a quadrotor model which will be used for the simulation purpose here.

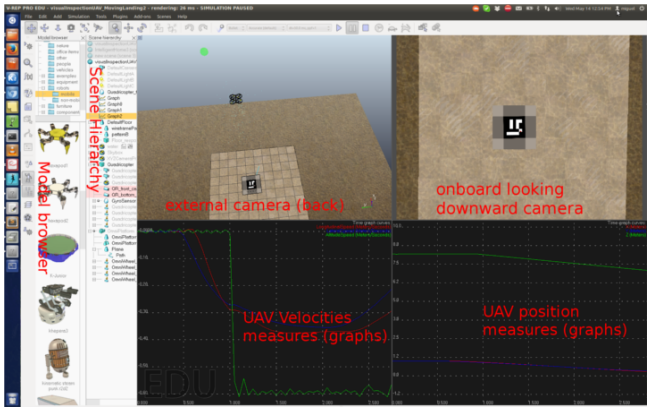

Fig. 1. Capture frame of the V-REP environment.

III. Vision Based Fuzzy Control approach

A. Vision Algorithm

The purpose of the vision algorithm is to obtain the position of the landing platform. With this information, the control system approach has to be able to command the UAV to center landing platform in the image, orientate it, and approximate it to the landing platform. The vision algorithm is not the principal purpose of this work, for this reason we use a visual algorithm based on the detection, recognition, and processing of augmented reality (AR) markers or codes. The idea is to have a moving landing platform with a code printed on it. Based on a markers database, the camera calibration parameters, and the size of the code, the algorithm is able to estimate the pose of the camera (quadrotor) respect to the marker, that is the orientation of the camera (the quadrotor) in the three axis, the distance between the camera (the quadrotor) and the code, as well as the lateral and vertical displacement versus the center of the marker. The developed ROS package for the visual algorithm is the adaptation to ROS of the ArUco software [17], a developed C++ library of augmented reality that uses OpenCV. It is an improved Hamming Code based algorithm with an error detection.

The ArUco-ROS package, called *aruco_eye* is subscribed (that is the ROS method to get information shared by other packages) to the specific image streaming publisher from the camera (real or virtual). The current frame is processed and the extracted information is sent by a publisher defined for this purpose.

To use this package with the virtual camera and a real one it is needed to include the camera calibration and the specific topic's name in the location where the camera published the images. This could be done by the command line or configuring a roslaunch file. To do the tests in the virtual environment a panel with one of the ArUco codes added as a texture is included, as it is shown in the Figure 2. In the real world the codes where printed and paste into a wall.

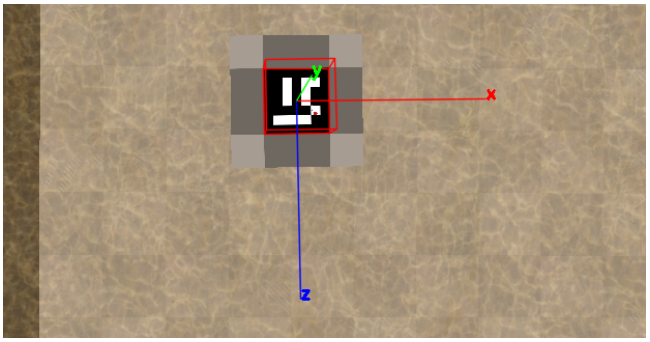

Fig. 2. Virtual image captured by the virtual camera on the UAV and processed using the ArUco ROS package.

The image processing algorithm consists of estimation of the distance between the ArUco target code and the UAV in the three axis (longitudinal, lateral and vertical distances), as it is shown in the Figure 3.

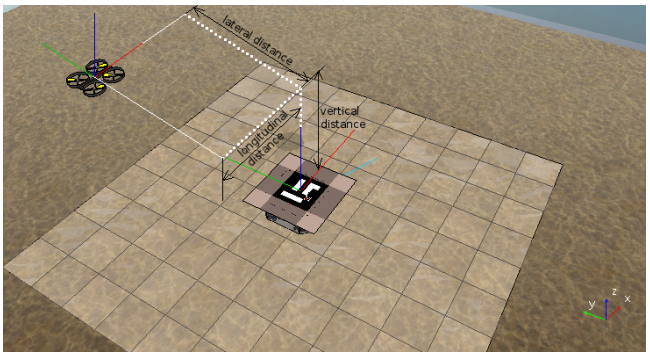

Fig. 3. Explanation of the image processing algorithm.

The developed ROS package presented in this section and some examples of how to use it, are available online [18].

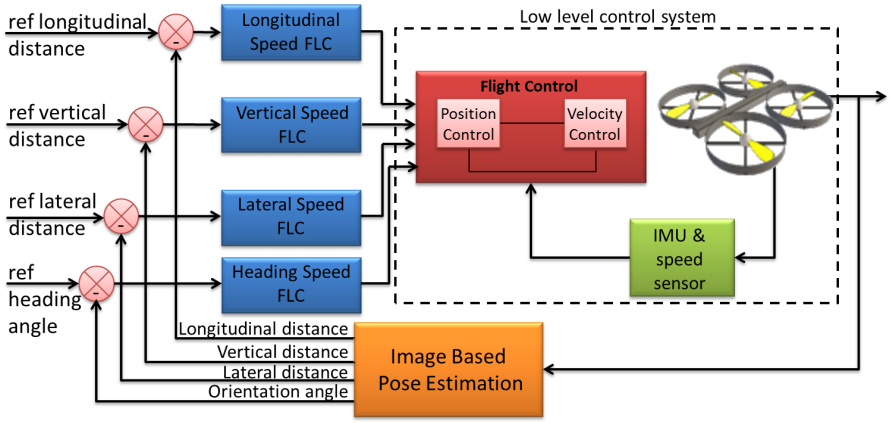

Fig. 4. Control loop of the Fuzzy control system approach for moving object tracking and autonomous landing.

B. Fuzzy Control

The autonomous landing task defined in this work is based on the capability of the quadrotor to change its position to land over the moving landing platform. The UAV must center itself to the landing platform, and once centered it must start to descend. To solve this task a control system approach was designed using four fuzzy controllers working in parallel. The longitudinal and lateral speed controllers keep the UAV positioning to have the moving landing platform in the center of the image. The vertical speed controller approximates the UAV to the landing platform. The heading controller modifies the heading of the UAV to have the lading platform well oriented. The longitudinal, lateral, and heading velocity controllers have been designed as a fuzzy PID-like controller, with three inputs and one output. The vertical speed controller is just a simple fuzzy PD-like controller. The longitudinal, lateral, and vertical speed controllers have as outputs velocity commands in meters per seconds, and for the heading velocity controller is degrees per seconds. Figure 4 shows the control loop of the control system approach implemented for this work.

All the controllers were defined in a simply way, with just three sets per each input, and five sets for the output, defined using triangular functions, that means that the rules' base is composed just with 27 rules for the longitudinal, lateral and heading velocity controllers and 9 rules for the vertical speed controller. The defuzzification method used is the height weight and the inference motor is the product. The rule base was defined based on the heuristic information of the relation of the three inputs. In the longitudinal and lateral speed controllers were taken into account the current pitch and roll angles of the UAV in the information obtained by the vision algorithm for the estimation of the translation in x and y axis respected to the moving landing platform.

The Tables I, II, III show the initial definition of the rule base.

To implement the fuzzy controllers in the ROS environment a new ROS package was developed called MOFS-ROS. This ROS package is the adaptation to ROS of the own developed C++ library MOFS (Miguel Olivares' Fuzzy Software) [19]. As well as the C++ library, this new ROS package (MOFS-ROS) allows to implement fuzzy controllers in an easy way, loading the specific characteristics of the controller and the rule base from two different txt files. This package interacts with the roscore and other packages by a service, that provides a new output each time that new inputs were received by the common subscriber/publisher ROS' communication policy. The specification of the subscriber and the publisher is done by a parameter in the *rosrun* command line or by a *roslaunch* file.

This ROS package allows to create new fuzzy controllers using triangular or trapezoidal membership functions, the product or the maximum inference motor and the height weight defuzzification method. Extended possibilities will be included in the next versions of this ROS package. More detailed information of this software is found in [20] in where

Dot/error	Left	Zero	Right
Negative	Left	Zero	Right
Zero	Zero	Right	Right
Positive	Right	Right	Big Right

TABLE I

BASE OF RULES WITH VALUE FOR THE THIRD INPUT (INTEGRAL OF THE ERROR) EQUAL TO **NEGATIVE**, BEFORE THE MANUAL TUNNING PROCESS

Dot/error	Left	Zero	Right
Negative	Left	Zero	Zero
Zero	Left	Zero	Right
Positive	Zero	Zero	Right

TABLE II

BASE OF RULES WITH VALUE FOR THE THIRD INPUT (INTEGRAL OF THE ERROR) EQUAL TO **ZERO**, BEFORE THE MANUAL TUNNING PROCESS

Dot/error	Left	Zero	Right
Negative	Big Left	Left	Left
Zero	Left	Left	Zero
Positive	Left	Zero	Right

TABLE III

BASE OF RULES WITH VALUE FOR THE THIRD INPUT (INTEGRAL OF THE ERROR) EQUAL TO **POSITIVE**, BEFORE THE MANUAL TUNNING PROCESS

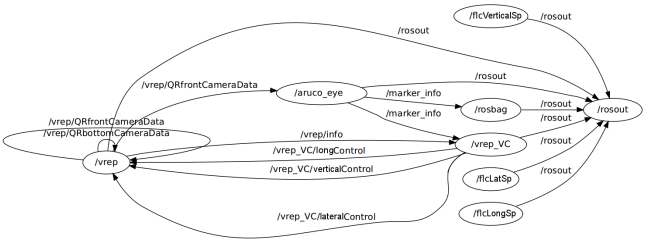

Fig. 5. Interaction between all the actives processes from V-REP and ROS during the simulator tests.

is explained the C++ library version.

An extra ROS package was developed to put across the information from the visual algorithm, to the fuzzy control system. This package is also in charge of sending the controllers' output to the virtual or real quadrotor. This package receives two different names, when it is used with the virtual quadrotor is called *VREP_VC* (VREP visual control), and *(*visual_control_system). This package is the only one that has to be modified depending the UAV to be used, the number of controllers to use, and the error's signals to control. In this way the ArUco-ROS package and the MOFS-ROS package is kept without significant changes to be used with the virtual or the real environment, or either for future control approaches. This package also contains some extra process when a real AR.Drone quadrotor is in use. This process is called *emergency_ardrone_control*, and it allows to send basic commands to take off, land, and to do an emergency stop to the AR.Drone. Through this process the user can also select the specific ArUco code to set as a target.

All the developed ROS packages presented in this section and some examples of how to use them are available online [18].

IV. EXPERIMENTS

A. V-REP Tests

1) Tuning Process: The simulation environment was set by a quadrotor with a looking forward camera, and a panel with one ArUCo code, as it is shown in Figure 6. The location of the quadrotor is set to have a two meters step signal when it starts to work.

A complete scheme of all the processes involved in the simulation environment is shown in Figure 5. Where *vrep* is the V-REP simulator environment, explained in section II, *aruco_eye* is the process of the vision algorithm explained in section III, the *flcLatSp*, *flcLongSp*, and *flcVerticalSp* are the fuzzy controllers for the lateral, longitudinal and vertical speed respectively, and *flcOrientSp* (explained in section III). The *vrep_VC* is the process that shares the visual information to the control system and send the control commands to the virtual quadrotor in the V-REP. The *rosbag* is an internal ROS package to store some data of each test. Finally, the *rosout* is the core of the ROS system.

During the manual tuning process, the range of the variables are first adapted and then the output of some specific rules. The V-REP simulation environment and in conjunction

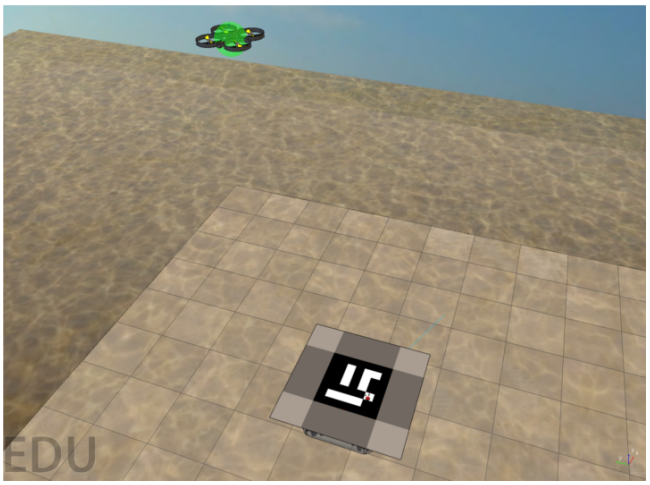

Fig. 6. V-REP environment design with a quadrotor and the ArUco code.

with ROS and the developed ROS' packages allow to test the controllers and to modify the different characteristics of the controllers easily. Because of the big similarities between the lateral, longitudinal, and heading controllers, the tuning process is applied to one of the three controllers, and then the results are used in the others. The tunning process is defined by the response of the lateral speed controller responding against a 2 meters step signal. First, the inputs' range have to be adjusted. The Figure 7 shows the response of some of the different configuration tested for the range of the inputs. It is started with the initial controller explained in section III, represented as *MF1* with the blue line in Figure 7. Then several modifications are tested on the range of the inputs and the output, some of them are shown in the mentioned Figure as *MF2*, *MF3*, *MF4*, being the last one (represented by the green line) the one that gets a better response. The variables definition of the three controllers is shown in Figure 8.

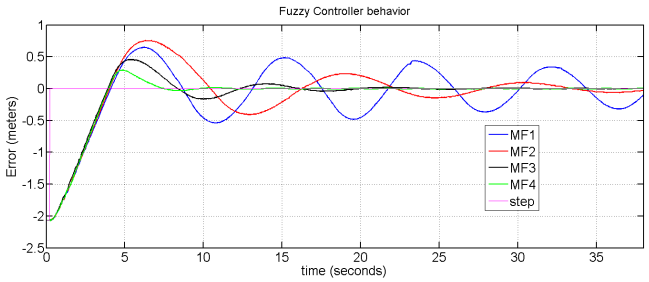

Fig. 7. Response of the different lateral speed controllers during the tunning phase of the inputs' range adjustment.

Based on this controller, tuning process it is continued with the adaptation of the rules' base. In this phase the output of few rules were modified to reduce the overshoot presented in the response of the best controller obtained in the previous phase (the *MF4*). Also for this phase the test is the same than in the previous phase, a step signal of 2 meters. The Figure 9 shows the behavior of some of the controllers tested. The behavior of the different controllers tested are quite similar

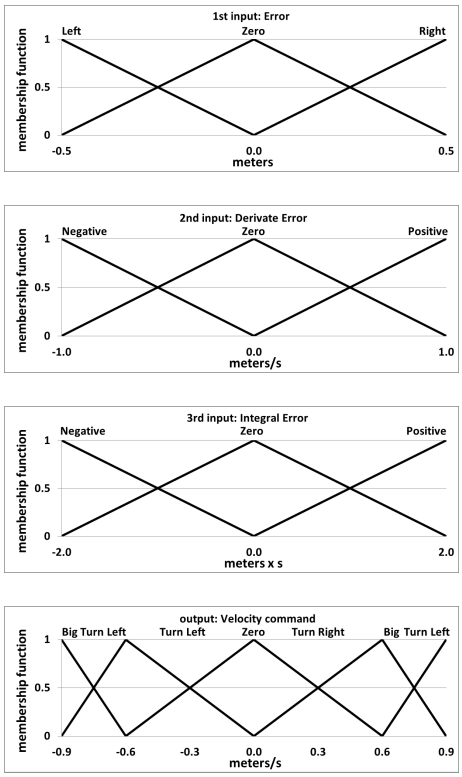

Fig. 8. Final design of the variables of the fuzzy controller after the manual tuning process.

unless one of them, that reduces completely the overshoot presented in the others (*Rules4*). The Tables IV, V, VI show the final state of the rule base after the manual tuning process. Fromm the initial base of rules presented in section III-B, 10 output's rules were changed.

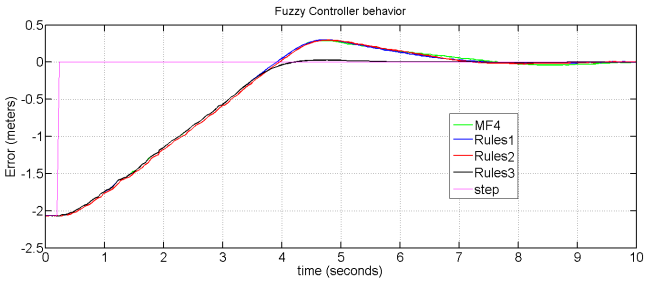

Fig. 9. Response of the different lateral speed controllers during the tunning phase of the rules' base adaptation.

Dot/error	Left	Zero	Right
Negative	Big Left	Left	Right
Zero	Left	Zero	Right
Positive	Right	Right	Big Right

TABLE IV

BASE OF RULES WITH VALUE FOR THE THIRD INPUT (INTEGRAL OF THE ERROR) EQUAL TO **NEGATIVE**, AFTER THE MANUAL TUNNING PROCESS

Once the lateral speed controller is tuned, this information is used to set the other controllers, the longitudinal, vertical,

Dot/error	Left	Zero	Right
Negative	Left	Left	Zero
Zero	Left	Zero	Right
Positive	Zero	Right	Right

TABLE V

BASE OF RULES WITH VALUE FOR THE THIRD INPUT (INTEGRAL OF THE ERROR) EQUAL TO **ZERO**, AFTER THE MANUAL TUNNING PROCESS

Dot/error	Left	Zero	Right
Negative	Big Left	Left	Left
Zero	Left	Zero	Right
Positive	Left	Right	Big Right

TABLE VI

BASE OF RULES WITH VALUE FOR THE THIRD INPUT (INTEGRAL OF THE ERROR) EQUAL TO **POSITIVE**, AFTER THE MANUAL TUNNING PROCESS

and heading velocities. An adaptation to degrees (inputs), and degrees per seconds (output) was done for the heading controller.

2) Autonomous Landing Tests: Once the four controllers are obtained, they can be used all together to control the virtual quadrotor in the specific task of autonomous landing. Two different test are defined. In the first one, the landing platform is static, and in the second one it change its position in the plane x, y and turn over the z axis. In both cases the code is located over a ground robot, to be easy to configure the movements of the landing platform. The initial position of the quadrotor is the same in both cases, 8.0 meters of altitude, and a displacement of 1.6 and 0.9 meters respect to the landing platform for the x, y axis of the quadrotor respectively. The vertical velocity controller has the constraint of not to act until the error in the x and y axis of the QR are reduced under 0.5 meters. This is an strategy to reduce the potential disturbances created by the action of the descend of the Quadrotor.

The behavior of the control system approach was evaluated using the root mean square error (RMSE) during both tests. The Table VII shows the RMSE values for the longitudinal, lateral and heading velocities controllers.

Controller type	Static Platform (RMSE value)	Moving Platform (RMSE value)	units
Longitudinal	0.5346	0.4615	meters
Lateral	0.3661	0.3777	meters
Heading	3.9029	4.3728	degrees

TABLE VII

ROOT MEAN SQUARE ERROR FOR THE LONGITUDINAL, LATERAL AND HEADING VELOCITIES CONTROLLERS IN THE TWO TESTS PRESENTED.

The videos related to the tests presented in this section are available online [21].

B. Real Tests

For the evaluation of the behavior of the control system approach an experimental environment was set using a real quadrotor and a ground vehicle. The quadrotor used was the AR.Drone parrot [22]. The UAV was modified to have the

looking forward camera in downward direction. The ground vehicle used was a youbot-kuka [23]. The youbot's robotic arm was removed to install a helipad as is shown in Figure 10. It was composed by three different ARuCo codes with different sizes, to be seen by the UAV at different distances. The ground vehicle was commanded with a smart phone based on movements measured by the internal gyroscopes. The omnidirectional wheels of this ground robot allow to move the platform in all the direction possibles in the ground plane, as well as change the orientation at any time.

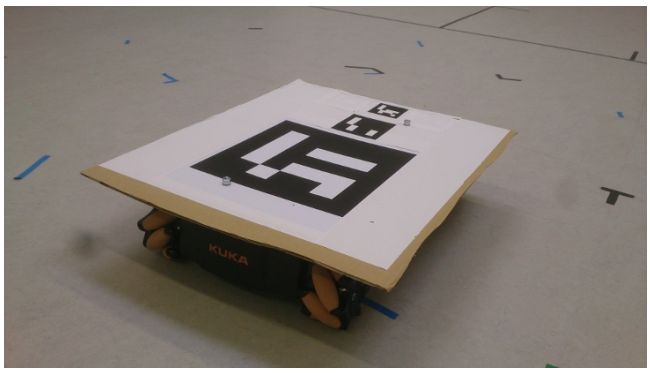

Fig. 10. Omnidirectional moving landing platform based on a Kuka youbot robot with three ARuCo's codes.

Two different type of tests were defined. The first one was defined to test the behavior of the control system to track the moving platform form a fixed altitude of 4.0 meters. The second one was defined to evaluate the control system to do an autonomous landing from an altitude of 3.5 meters.

In the case of the tracking the helipad, the vertical speed controller was set to keep the UAV at a predefined altitude from the helipad. The other controllers work in parallel to keep the platform in the center of the the image, and with the desire orientation. The Figure 11 shows the evolution of the control system for the vertical, lateral and longitudinal controllers. The Figure 12 shows the performance of the heading controller during this test. In this Figure is also shown a big orientation change in the moving platform in the first half of the test.

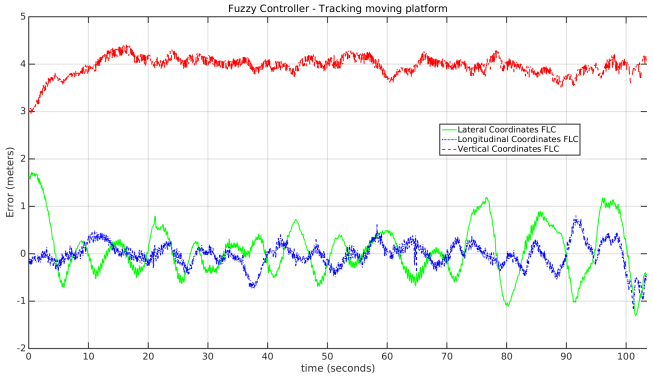

Fig. 11. Evolution of the error of the lateral, longitudinal and vertical controllers during one of the experiments of tracking the moving helipad.

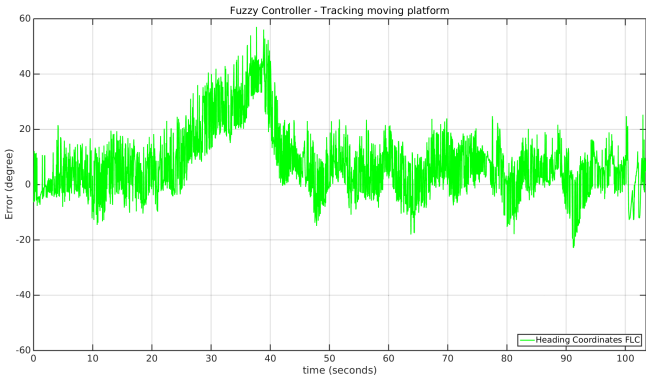

Fig. 12. Evolution of the error of the heading controller during one of the experiments of tracking the moving helipad.

The Table VIII shows the root means squared error (RMSE) of the different control inputs of the control system approach for two different tracking a moving platform experiments. The Experiment #1 corresponds to Figures 11, 12. In both experiments the helipad platform was in continuous movements during the almost 2 minutes of the tests. The RMSE values of the mentioned table shown the good behavior of the control system approach for the tracking experiments, with a average value in the two experiments of 0.51915 meters for the lateral velocity controller, 0.17815 meters for the longitudinal velocity controller, 8.18005 degrees for the heading controller, and 0.3945 meters for the vertical velocity controller.

Controller type	Experiment #1 (RMSE value)	Experiment #2 (RMSE value)	units
Longitudinal	0.1876	0.1687	meters
Lateral	0.6664	0.3719	meters
Heading	7.8555	8.5046	degrees
Vertical	0.3726	0.4164	meters

TABLE VIII

ROOT MEAN SQUARE ERROR FOR THE LONGITUDINAL, LATERAL, VERTICAL AND HEADING VELOCITIES CONTROLLERS IN THE TWO EXPERIMENTS DONE DURING THE TRACKING OF A MOVING PLATFORM.

For the autonomous landing on a moving platform experiments a constraint for the vertical velocity controller was included. This controller is not going to act until the error's absolute value for the lateral and longitudinal controller is smaller than 0.2 meters. This constraint is included to compensate the reduction of the field caused by the reduction in the altitude. The finalization of the landing task is done when the measured altitude between the UAV and moving helipad is smaller than 0.4 meters. In this moment, and when the previous mentioned constraint happens, the UAV reduce gradually the speed of the motors (trust torque) and lands. A small tracking phase is also included at the beginning of the test. The landing phase is activated by a keyboard through one of the developed ROS packages previously mentioned. Figures 13, 14 show the evolution of the control system of the lateral, longitudinal and vertical velocities for the first mentioned Figure, and the heading controller for the

second mentioned Figure. In this last Figure is shown how the initial orientation of the UAV is turned more than 60 degrees respect to the desire orientation. In the Figure 13 is shown the tracking phase and the landing phase, as well as the exactly moment when the control system stop to work at the altitude of 0.4 meters.

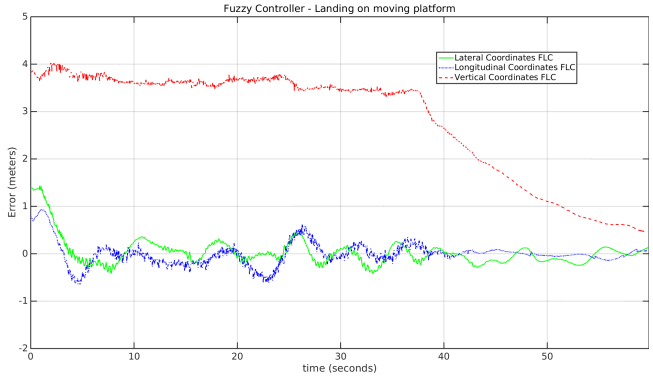

Fig. 13. Evolution of the error of the lateral, longitudinal and vertical controllers during one of the experiments of autonomous landing on the moving helipad.

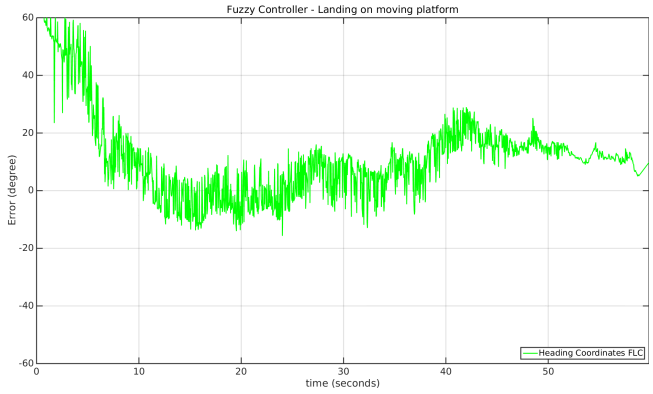

Fig. 14. Evolution of the error of the heading controller during one of the experiments of autonomous landing on the moving helipad.

The Table VIII shows the root means squared error (RMSE) of the different control inputs of the control system approach for the tracking a moving platform experiments. The performance of the vertical controller is not shown because, the measure of the error between the current altitude of the UAV and the desire altitude position, that means 0 meters, is not going to be 0 until the end of the experiment, when the UAV is landed. The Figures 13, 14 represent the behavior of the experiment #3. It is the most representative because of the big difference in the heading desire position. The RMSE values of the mentioned table shown the good behavior of the control system approach for the autonomous landing experiments, with a average value in the three experiments of 0.51915 meters for the lateral velocity controller, 0.3091 meters for the longitudinal velocity controller, and 14.2572 degrees for the heading controller. It has to be taken into account that the big difference between the average value of the heading controller in this kind of test and the previous one is because of the previously mentioned error at the starting point of the test.

Controller type	Experiment #1 (RMSE value)	Experiment #2 (RMSE value)	Experiment #3 (RMSE value)	units
Longitudinal	0.2199	0.3708	0.3368	meters
Lateral	0.6984	0.3989	0.4524	meters
Heading	7.1984	7.7271	27.8461	degrees

TABLE IX

ROOT MEAN SQUARE ERROR FOR THE LONGITUDINAL, LATERAL AND HEADING VELOCITIES CONTROLLERS IN THE TWO EXPERIMENTS DONE DURING THE LANDING ON A MOVING PLATFORM.

All the videos related to these experiments are available online [24] [21].

V. CONCLUSIONS

In this paper is presented the use of the V-REP simulation environment in connection to the Robotics Operative System (ROS) for the design and tuning of a Fuzzy control approach for the complete control of a UAV based on visual information. The selected tasks to do the tuning and test this control approach were the object tracking and autonomous landing, both with moving targets. A vision algorithm based on augmented reality codes was used to estimated the pose of the UAV. The presented control approach was composed by four Fuzzy controllers working in parallel to manage the lateral, longitudinal, vertical and heading velocities of the UAV. The control approach was designed to keep the center of the moving platform in the center of the image, with a predefined orientation. The vertical controller was designed to keep the UAV at a predefined distance for the tracking experiments, and to land on the moving platform for the landing task. The control tuning process was implemented in a simulated environment using V-REP and ROS, as well as some simulated tests for the tracking and the autonomous landing tasks. The evaluation of the control approach was done in an indoor environment with a quadrotor and a ground vehicle equipped with omnidirectional wheels, as the moving landing platform. Two different kind of real experiments were presented, one for tracking the moving target from a fixed altitude, and another for the landing on it. In both cases the control system accomplished successfully a set of tests. The evaluation of the performance of the control system was presented by the RMSE values of a set of experiments.

The future work is focus in two different kind of test, on one hand to do a more precise evaluation of the control approach using a motion capture system, and on the other hand to test the system in outdoor environment. After that, the authors will focus their effort to estimate the pose of the UAV without using augmented reality codes, keeping in mind the assumptions of GPS-denied environments.

ACKNOWLEDGMENT

The authors would like to thanks Jan Dentler, Arun Annaiyan and Raphael Hinger from the Automation Research Group of the SnT - University of Luxembourg, for their support in the experimental tests.

REFERENCES

[1] Webots official site. http://www.cyberbotics.com, 2014.

[2] Peter corke's robotics toolbox for matlab. http://petercorke.com/Robotics_Toolbox.html, 2014.

[3] Robot operating system (ros). http://ros.org, 2014.

[4] Gazebo 3d simulator. http://ros.org/wiki/gazebo, 2014.

[5] Coppelia robotics. virtual robotics experimentation platform (v-rep). http://www.vrep.org, 2014.

[6] Wiki site of the ros bridge with v-rep. http://wiki.ros.org/vrep_ros_bridge, 2014.

[7] P. Campoy, J. F. Correa, I. Mondragón, C. Martínez, M. A. Olivares, L. Mejías, and J. Artieda. Computer vision onboard uavs for civilian tasks. *Journal of Intelligent and Robotic Systems*, pages 105–135, 2009.

[8] W. Ding, Z. Gong, S. Xie, and H. Zou. Real-time vision-based object tracking from a moving platform in the air. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 681–685, 2006.

[9] Celine Teuliere, Laurent Eck, and Eric Marchand. Chasing a moving target from a flying uav. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS2011*, pages 4929–4934, 2011.

[10] F. Ruffier and N. Franceschini. Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, pages 2339–2346, 2004.

[11] D. Lee, T. Ryan, and H.J. Kim. Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 971–976, 2012.

[12] U. Zengin and A. Dogan. Cooperative target pursuit by multiple uavs in an adversarial environment. *Robotics and Autonomous Systems*, pages 1049–1059, 2011.

[13] Joseph Egbert and Randal W. Beard. Low-altitude road following using strap-down cameras on miniature air vehicles. *Mechatronics*, pages 831–843, 2011.

[14] G. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald. A real-time method to detect and track moving objects (datmo) from unmanned aerial vehicles (uavs) using a single camera. *Remote Sensing*, pages 1090–1111, 2012.

[15] Miguel A. Olivares Mendez, S. Kannan, and H. Voos. V-rep & ros testbed for design, test, and tuning of a quadrotor vision based fuzzy control system for autonomous landing. In *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014*, August 2014.

[16] Miguel A. Olivares Mendez, S. Kannan, and H. Voos. Setting up a testbed for uav vision based control using v-rep amp; ros: A case study on aerial visual inspection. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 447–458, May 2014.

[17] S. Garrido-Jurado, R. Munoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marin-Jimenez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[18] Miguel angel olivares-mendez snt homepage, 2014.

[19] M.A. Olivares-Mendez, P. Campoy, C. Martinez, and I. Mondragon. A pan-tilt camera fuzzy vision controller on an unmanned aerial vehicle. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2879–2884, Oct.

[20] I. Mondragón, M. A. Olivares-Méndez, P. Campoy, C. Martínez, and L. Mejias. Unmanned aerial vehicles uavs attitude, height, motion estimation and control using visual systems. *Autonomous Robots*, 29:17–34, 2010. 10.1007/s10514-010-9183-2.

[21] Youtube channel of the automation research group at snt-university of luxembourg: Automation research group snt.uni.lu. https://www.youtube.com/channel/UCBkpapz06ViwK_cztjwqCAQ , 2014.

[22] Ar.drone parrot. http://ardrone.parrot.com, 2013.

[23] Kuka youbot robot official site, 2014.

[24] Isruav project homepage of the automation research group at snt - university of luxembourg. http://wwwen.uni.lu/snt/research/automation_research_group/projects/isruav, 2014.