Proceeding of the IEEE 28th
Canadian Conference on Electrical and Computer Engineering
Halifax, Canada, May 3-6, 2015

Vision Based Autonomous Gap-Flying-Through Using the Micro Unmanned Aerial Vehicle

Erli Lyu, Yuan Lin, Wei Liu, Max Q. –H. Meng

***Abstract*—a vision-based enhanced PID control algorithm for the gap-flying-through problem using a low-cost multirotor platform is the main topic of this paper. We designed an image processing program to detect the target rectangle gap in each video frame captured by onboard camera of the drone. During the process, some manipulation such as dilate and erode are implemented to reduce the noise. To control the motion of the drone, we applied a PID control algorithm with several enhancement like segmented coefficients, restricted integral saturation and integral separation. The current version of our system is based on an AR.Drone quadrotor. Our program is executed on a laptop, which cooperates with the drone through Wi-Fi connection and Robot Operation System (ROS). Our system have been evaluated with a series of indoor experiments under a relatively harsh circumstances, with success rate of 81.67%, the effectiveness of our system has been demonstrated.**

I. Introduction

RECENTLY, micro UAVs have gained much attention in a wide range of research. For instance, both indoor [1, 2] and outdoor [3-5] UAV navigation technology have been well implemented, plus [6] provided a denoising and drift-control method for drawing the trajectory. Meanwhile, there are lots of projects focusing on UAV landing problem. T.K. Venugopalan and Tawfiq Taher provided a good method for brightness correction of images in their research about autonomous UAV landing on a marine vehicle [7]. An approach of obtaining target information by using HSV picture was presented by Chi-Tinh Dang and Hoang-The Pham [8]. In [9], with simulation based on ROS Gazebo and TUM AR.Drone Simulators, researchers performed landing maneuvers on a mobile base using fuzzy logic. However, those landing problems are quite straightforward, since the absolute distance between the drone and the target can be easily perceived. Jesus Pestana and Jose Luis Sanchez-Lopez used an AR.Drone quadrotor to follow user assigned objects in relatively spacious unknown outdoors environments [10]. They discussed some useful geometrical characteristics of target displayed in video frames. Rather than tracking objects on the ground, [10] focuses on tracking objects in the air. In our paper we name a group of problem as Gap-Flying-Through problem, which is similar to the basic landing problem, but more challenging in both control and vision perception.

Gaps, like building windows, could be found almost everywhere. It is a channel that links two separated spaces. In some cases, the user may attempt to obtain information from the other side of a gap, or just want the UAV to fly through the gap (e.g. considering the UAV based express delivery system). Thus, a system that can autonomously fly through the gap is quite valuable.

In [11], Kumar and Mellinger controlled the drone to achieve a series of aggressive maneuvers of high difficulty, one of which was flying through a perpendicular gap. They accomplished this task by implementing several segmented trajectories with iteratively updated controller. But in that paper the position of the drone was obtained by the Vicon system, which could not be pre-installed in the real world. In our work, we would like to use a low cost drone with onboard camera and sensors to accomplish an easier mission.

The Gap-Flying-Through problem considered in this paper has been simplified. First of all, we only consider a rectangular red gap in this work. Secondly, the procedure merely include taking off, flying through the gap, hovering for a while and posteriori landing. During the design process of our control system, we referred to the ideas described in [10], but we've made some improvement to meet the specific requirement of actual environment. Also, we implemented a rectangular detection program based on OpenCV to find the gap and return features as the input of the control program.

The remainder of the paper is organized as follows. The hardware platform would be briefly described at Section II. The description of whole system would be introduced at Section III. In Section IV, we will report the experiment result. Finally, Section V draw a conclusion.

II. Platform

In this research, we use the AR.Drone 2.0 quadrotor as our hardware platform. Produced by the Parrot Company, AR.Drone 2.0 is designed to be an autonomous UAV with inner control system to conduct command sent by the user, and maintain position while being disturbed. It can communicate with smart phone and computer via Wi-Fi. In addition, all the navigation data and video could be obtained through given API. The main processor of AR.Drone 2.0 is ARM Cortex-A8 with 1 GHz basic frequency. The AR.Drone 2.0 has two cameras, one is positioning to the ground, and the other one is oriented horizontally, positioning forwards. The front camera has 93° view angle, and can deliver images at high resolution as

This project is supported by RGC GRF# CUHK-14205914 awarded to Professor Max Q.-H. Meng.

Erli Lyu is with the School of Mechanical Engineering and Automation, Harbin Institute of Technology Shenzhen Graduate School, Guangdong, China (email: erlilyu@outlook.com)

Yuan Lin is with the School of Mechanical Engineering and Automation, Harbin Institute of Technology Shenzhen Graduate School, Guangdong, China (email: yuan_lin@hotmail.com)

Wei Liu is with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong (email:weiliu4@gmail.com)

M. Q.-H. Meng is with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong (email: qhmeng@ee.cuhk.edu.hk)

978-1-4799-5829-0/15/$31.00 ©2015 IEEE

720p at 30 fps. The AR.Drone 2.0 contains a low-cost IMU chip that consists of a 3-axis accelerometer and a 2-axis gyroscope. A more accurate 1-axis vertical gyroscope is considered for the vertical axis. Plus, 2 ultrasonic sensors, that can measure distance as large as 6m at 25 Hz is embedded at the button of the drone. [12]

The inner communication between every components in our system is under the Robot Operation System (ROS) framework [13], and its distribution is Hydro. The ardrone_autonomy ROS package [14] is used in our work as a ROS driver for the drone. In our work we take the image and the navigation data as message and using subscriber to receive the message.

In our work we employ a relatively narrow rectangular gap, whose dimension is 60cm × 25cm. Compared to the size of AR.Drone 2.0, which is about 52cm×52cm×11cm with the indoor hull, the gap is only 14cm longer in the height (the distance would be even smaller when the drone flying through the gap) and 8cm longer in the width. We made this gap to make sure our control algorithm is sufficiently precise.

III. System Description

A. General Procedure

Our procedure for dealing with such problem under experimental condition could be described as follow. Initially, the UAV is placed at 3-6 meters in front of a gap with random orientation. After taking off, the UAV would fly up-and-down along a spiral trajectory, while transferring video to the computer via Wi-Fi. At the same time, an OpenCV based algorithm would keep analyzing the video frames so as to find the gap. Once the gap is properly recognized, several geometric features derived from the gap's four vertex coordinates would become the feedback of the controllers, which would command the UAV to approach a pre-defined position. If the UAV could maintain its stability at the pre-defined position for about 0.3s, it would move to the next phase: flying through the gap with a pre-designed trajectory. After successfully flying through the gap the UAV would hover for about a second and land on the ground.

This system is mainly comprised of two parts: Image processing algorithm and an enhanced PID control algorithm.

B. Angle-Associated Error Correction

In order to obtain the correct image parameter, we must compensate the coordinate error caused by non-zero pitch or roll angle every time a new image arrives. In this process we get the pitch and roll angle from the latest navigation data.

For the pitch angle compensation, we implement a linear model considering the pitch angle is relatively small in our experiment. For 1 radians we compensate 12.5 pixel in the image plane.

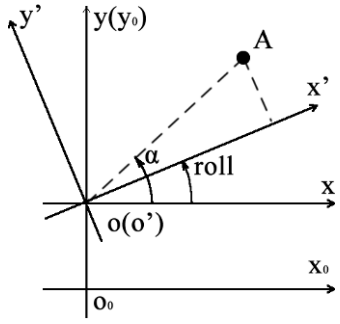


Fig. 1. Coordinate Compensation

For the roll angle compensation, we calculate the real position $(A_{x'}, A_{y'})$ by the following equations.

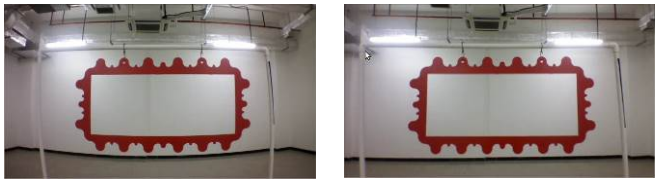
$$\alpha = a\tan 2(A_y, A_x)$$
$$A_{x'} = dis(\mathrm{AO}) \cdot \cos(\alpha - roll) \quad (1)$$
$$A_{y'} = dis(\mathrm{AO}) \cdot \sin(\alpha - roll)$$

C. Image Processing

The Gap finding part is the most challenging part in our work. It is achieved by image processing program, which is based on OpenCV and written in C++. The main modules are listed as follow:

1) Distortion Correction

As the drone approaching the gap, the distortion become more obvious. When the drone fly close to the gap, it is almost impossible for the program to identify the rectangle without distortion correction.



Original Image Image After Correction

Fig. 2. Distortion Correction

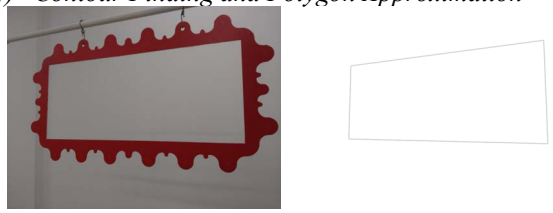
2) Channel Extraction

In this project, we use a red gap to simplify the Gap-finding process. At the beginning, we extract each color channel from the raw image, sum up the green and the blue channel with 0.5 weight factor and then subtract this weighted summation from the red channel, as a result, we got a single-channel image.

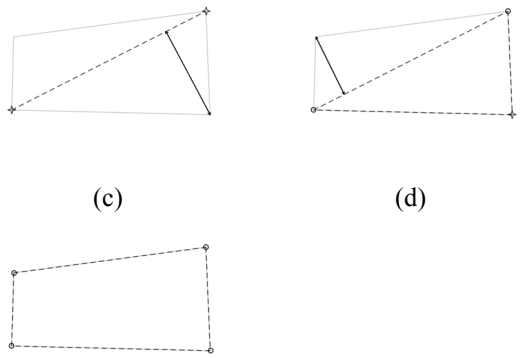
3) Canny Edge Detection

At this modules we first implement the morphological operation to the extracted image and then use the built-in canny function to detect the edge.

4) Contour Finding and Polygon Approximation



(a) (b)



(c) (d)

(e)

Fig. 3. Contour Finding and Polygon Approximation

Followed by the canny detection step we implement an OpenCV built-in function to find the contours. For every contour we found, the cvApproxPoly function is implemented to figure out whether it's a rectangular or not. As shown in Fig. 3, the cvApproxPoly would act in this way. It would start from two points, which have the farthest distance. Then the function would draw a line between these points, and find next point which has the longest distance to this line. The function would then add this point to the result. This point-finding procedure would execute iteratively until the precision of approximation is acceptable [15].At last, we implement several conditions to determine whether the contour we approximate is the wanted contour or not. After successfully finding the rectangular, our algorithm would return coordinates of its vertexes.

D. Error Calculation Algorithm

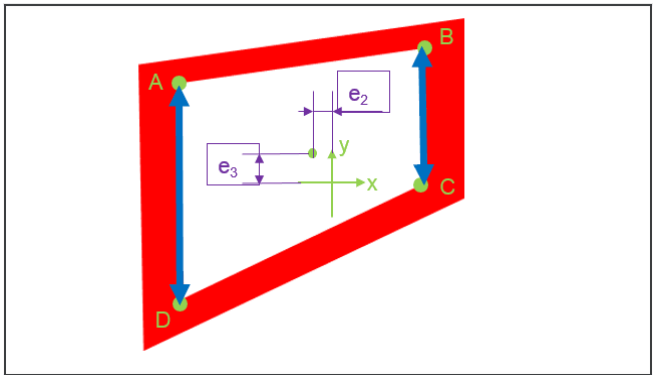


Fig. 4. Error Calculation Algorithm

The error parameter is derived from the previous gained vertexes.

$$e_0 = r_{\exp} - \frac{S_{ABCD}}{S_{view}}$$
$$e_1 = 1 - \frac{|\overrightarrow{AD}|}{|\overrightarrow{BC}|} \quad (2)$$

e_2 and e_3 is calculate as shown in Fig. 4 The corresponding relation is listed at TABLE I.

TABLE I
CORRESPONDING RELATION

Error Parameter	Controlled Variables
e_0	Pitch
e_1	Roll
e_2	Yaw
e_3	Gaz

A. Control Algorithm

The main focus of the whole control algorithm is to ensure the proper execution of function Searching(). The fundamental components of Searching() are SearchGap() and GotoFront(). SearchGap(), just as its name implies, is a phase when host PC commands the AR.Drone to find the target gap by flying gently up and down while rotating about z axis(The coordinate is defined as Fig. 5). Once a valid target frame is detected, the program will switch to execute function GotoFront() so the drone will move to the desired location that is close to the gap. The GotoFront() function will be executed iteratively until the target is lost or the isBalance() condition is satisfied. Consequently, the GotoFront() function is the most important one in control algorithm. And, it is what we will mainly discuss in this section.

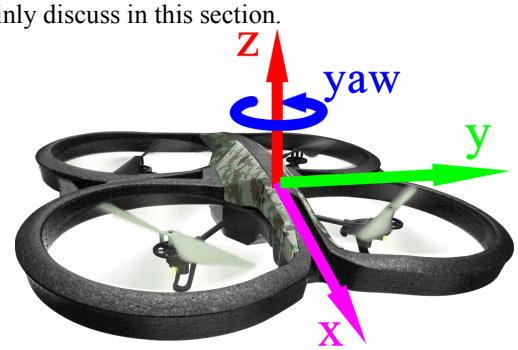


Fig. 5. The Coordinate Definition of AR.Drone

1) General Control Structure

As shown in Fig. 6, the system generally consists of two control loops.

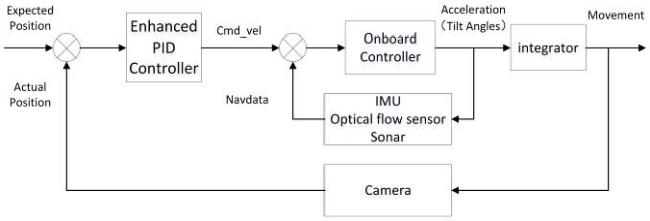


Fig. 6. The General Structure of the Control System

The inner loop autonomously runs on the guest control board embedded on the AR.Drone quadrotor. During every control period, the host PC sends an expected pose to the quadrotor through Wi-Fi connection. Meanwhile, the quadrotor frequently measures its actual pose by fusing the data obtained from each onboard sensor. Then, the onboard controller will compare the actual pose with expected pose and adjust the rotation rate of each motor accordingly, so as to reduce the difference between these two poses.

$$G = F_{\text{Throttle}} \times \cos(\theta)$$
$$F_h = F_{\text{Throttle}} \times \sin(\theta)$$
$$\ddot{x}_h = \frac{F_h}{m} \quad (3)$$
$$\theta = \theta(roll, pitch)$$

We know when AR.Drone holds its height, a part of the thrust must neutralize the force of gravity. By the principle

of force decomposition, we can draw a conclusion that the acceleration in horizontal direction is a function of the tilt angles like pitch and roll. Therefore, as explained in(3), sending a pose expectation (same as tilt angle expectation) is equivalent to demand corresponding acceleration.

The execution of outer loop relies on cooperation of the host PC and the quadrotor. The AR.Drone keeps sending video data to the host PC frame by frame. Once a valid rectangle is detected in a video frame, the fore-mentioned error calculation algorithm will be executed. Afterwards, the PID controller will take the position error as input and send expected values of all four DOF to the inner control loop.

The combination of these two control loops will help AR.Drone to approach the expected position and prepare for the execution of GoThrough().

2) Enhanced PID Controller

The classic proportional integral derivative (PID) control algorithm has been widely implemented in the UAV control problem [7, 8, 10]. In this project, we used a PID control algorithm with several enhancements, such as segmented coefficients, restricted integral saturation and integral separation.

The basic formula form is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \quad (4)$$

As we know, all the movements of AR.Drone are achieved by assigning particular rotation rate to each motor. That is to say, motions in different degree of freedom (DOF), are seriously coupled. Therefore, we must use four PID controllers to adjust all four errors simultaneously.

a) Segmented Coefficients

According to [10], the area of a certain object in view is proportional to the distance between the object and the camera. So the line segment length changing rate is non-linear about the distance change. In order to achieve a better performance in this non-linear condition, we used PID controllers with segmented coefficients. Different distance between the quadrotor and the target will activate different set of PID coefficients.

Another reason for using segmented coefficients is different requirements for each DOF error adjustment. For instance, when the gap is small in view, the roll adjustment is unnecessary, so the corresponding coefficient could be zeros. Likewise, when the drone is too close to the gap, we need a larger K_p for pitch adjustment to drive it backwards in time.

b) Positional or Incremental

In a discrete control system, the output of a positional PID controller for the Kth period must be derived as(5).

$$u(k) = K_p e(k) + K_i \sum_{j=0}^{k} e(j) + K_d \left[e(k) - e(k-1) \right] \quad (5)$$

Similarly, the value for $(K-1)th$ period is

$$u(k-1) = K_p e(k-1) + K_i \sum_{j=0}^{k-1} e(j) + K_d \left[e(k-1) - e(k-2) \right] \quad (6)$$

Subtract (5) from(6), we can obtain the formula of incremental PID algorithm as

$$\begin{aligned} \Delta u(k) &= u(k) - u(k-1) \\ &= K_p \left[e(k) - e(k-1) \right] + K_i e(k) \\ &\quad + K_d \left[e(k) - 2e(k-1) + e(k-2) \right] \end{aligned} \quad (7)$$

Since the incremental form no longer needs error accumulation, it is often preferred in most implementation.

Generally these two forms of PID control algorithm seem equivalent, however, the same coefficient may play different role. For instance, the effect of K_p is directly related to current error $e(k)$ in positional PID controller. However, by the first term of(7), we noticed that the K_p in incremental PID algorithm have closer relation to the changing rate of error, which is quite similar to the function of K_d in positional PID algorithm (but not necessarily the same). Thus, we must have a second thought about the possible outcome when tuning the coefficients of incremental PID controller.

Otherwise, the relation between the controller output and the variable we want to control is also a determining factor of which form we shall use. When the system is almost stable-in other words, the errors are nearly zero- the output of positional PID controller would definitely be quite small, but that of incremental PID controller might just have small changing rate rather than actually being small. Therefore, the incremental PID controller probably fits well in motor rotation control when the control value is the driving electric current. As for our system, the positional PID controller shall be better. Because the PID controller in our system ought to send a zero velocity expectation when the AR.Drone is almost in position.

Considering the convenience in coefficient tuning process and the relation between velocity command and desired position, we decided to use the positional PID controller.

c) Restricted integral saturation

As we can see in(5), the discrete positional PID controller features an error accumulation item, which originate from the integral item in(4). So long as we use such item with proper parameter it will effectively eliminate the static error. Otherwise, the accumulated error would be troublesome.

We use the area ratio of target rectangle and whole image frame to indicate the distance between the AR.Drone and the target. As displayed in Fig. 7, the error for pitch adjustment would be positive for a long time since the AR.Drone approaches to the target at a low speed. If this error accumulated for the whole process, the AR.Drone may rush towards the target despite of our distance expectation.

Fig. 7. Distance Expectation (Top View)

For this consideration, we've set limits to each error accumulation item.

d) Integral separation

The advantage of using integral item in PID controller is that we can get rid of static error, while the tradeoff is lower response speed. Hence, we let the integral item be available only if the error is small.

3) Error Prediction

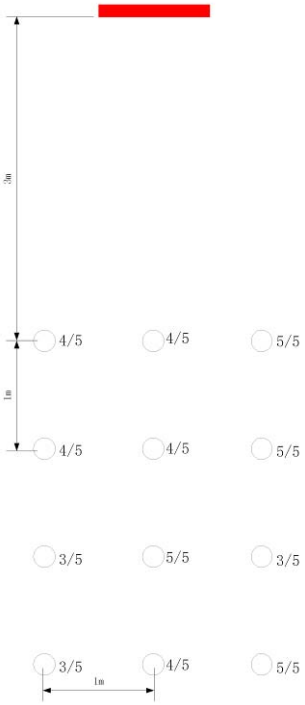
During the test, we realize that occasionally missing target is inevitable. Thus, we designed an error prediction algorithm to smoothen the process. Once the target is lost, we will apply least square approximation on latest errors to generate an estimation of current error.

When the occurrence of error prediction has reached the limit we set, the AR.Drone will switch to execute the function SearchGap().

IV. Experiment and Result

(a) Frame holder

(b) Smooth floor with no marks



(c) Test Result (Top View)

Fig. 8. Experiment Result

In order to examine the reliability of our system, we've built a frame holder as shown in (a) of Fig. 8, the height of frame center is around 132.5cm. The size of inner rectangle of the red frame is 60cm×25cm, while the size of AR.Drone quadrotor is approximately 52cm×52cm×11cm.

At the commence of each test, we place the AR.Drone quadrotor on one of the initial points displayed in (c) of Fig. 8, then start our algorithm and record whether the AR.Drone quadrotor has successfully flown through the gap. The success rate on each initial point is demonstrated in (c) of Fig. 8 as well. The average success rate we finally get is 81.67%.

Though the success rate we obtained is not as high as we expected, we know the actual value could be higher. Since AR.Drone implement the optical flow and corner tracking algorithm to process the video obtained from the button camera for speed measurement [12, 16], the performance of the drone would be deteriorated when lacking enough reference marks on the floor.

In this test, we experimented the performance of drone above a smooth floor with no marks (see (b) of Fig. 8), which can be regard as the worst condition. It is reasonable to expect a better result in other environment.

V. Conclusion

Our paper aims at providing a solution for the Gap-Flying-Through problem using the low-cost AR.Drone 2.0. Based on Wi-Fi connection, the transfer of data and command is managed by our program with the support of ROS. An OpenCV based computer vision program is used to recognize the gap. Furthermore, an enhanced PID control algorithm is proposed in this paper to control the drone to reach the desired place.

In our work, we deliberately use a small gap, intending to improve the difficulty of the control problem. We conducted an indoor experiment with no additional mark on the ground and have success rate of 81.67%, which demonstrate the effectiveness of our system.

VI. Future Work

Valuable works could be done to improve the performance of our system. First of all, a more robust vision algorithm would reduce the blind zone and increase the success rate of the flying-through step. Moreover, in our work, the UAV could not autonomously decide whether the recognized gap is accessible or not. But if we relax the restriction of only using onboard camera, we could install a range sensor on the drone to obtain depth information, so as to make it capable of making such judgement.

References

[1] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, Nara, 2007, pp. 225-234.

[2] D. Forsyth, P. Torr, A. Zisserman, G. Klein, and D. Murray, "Improving the Agility of Keyframe-Based SLAM," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr and A. Zisserman, Eds.: Springer Berlin Heidelberg, 2008, pp. 802-815.

[3] A. Yol, B. Delabarre, A. Dame, J. E. Dartois, and E. Marchand, "Vision-based absolute localization for unmanned aerial vehicles," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Chicago, IL, 2014, pp. 3429-3434.

[4] G. Vasarhelyi, C. Viragh, G. Somorjai, N. Tarcai, X. Szo, T. Renyi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Chicago, IL, 2014, pp. 3866-3873.

[5] T. Nageli, C. Conte, A. Domahidi, M. Morari, and O. Hilliges, "Environment-independent formation flight for micro aerial vehicles," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Chicago, IL, 2014, pp. 1141-1146.

[6] C. Wang, W. Liu and M. Q. H. Meng, "A Denoising and Drift-control Approach for UAV Trajectory Tracking," in *Robotics and Biomimetics, IEEE International Conference on* Bali, Indonesia, 2014.

[7] T. K. Venugopalan, T. Taher and G. Barbastathis, "Autonomous landing of an Unmanned Aerial Vehicle on an autonomous marine vehicle," in *Oceans, 2012*, Hampton Roads, VA, 2012, pp. 1-9.

[8] D. Chi-Tinh, P. Hoang-The, P. Thanh-Binh, and T. Nguyen-Vu, "Vision based ground object tracking using AR.Drone quadrotor," in *Control, Automation and Information Sciences (ICCAIS), 2013 International Conference on*, Nha Trang, 2013, pp. 146-151.

[9] P. Benavidez, J. Lambert, A. Jaimes, and M. Jamshidi, "Landing of

an Ardrone 2.0 quadcopter on a mobile base using fuzzy logic," in *World Automation Congress (WAC), 2014*, Waikoloa, HI, 2014, pp. 803-812.
[10] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy, "Computer vision based general object following for GPS-denied multirotor unmanned vehicles," in *American Control Conference (ACC), 2014*, Portland, OR, 2014, pp. 1886-1891.
[11] D. Mellinger, N. Michael and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *INTERNATIONAL JOURNAL OF ROBOTICS RESEARCH,* vol. 31, pp. 664-674, 2012.
[12] P. Bristeau, F. Callou, D. Vissière, and N. Petit, "The Navigation and Control technology inside the AR.Drone micro UAV," in *18th IFAC World Congress* Milano, Italy, 2011, pp. 1477-1484.
[13] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.
[14] M. Monajjemi, "ardrone_autonomy : A ROS Driver for ARDrone 1.0 & 2.0,", 2012.
[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, 1st ed.: O'Reilly Media, 2008.
[16] J. Jimenez Lugo and A. Zell, "Framework for autonomous onboard navigation with the AR.Drone," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, Atlanta, GA, 2013, pp. 575-583.