Automatic Navigation and Landing of an Indoor AR. Drone Quadrotor Using ArUco Marker and Inertial Sensors

Mohammad Fattahi Sani, Ghader Karimian
Faculty of Electrical and Computer Engineering
University of Tabriz
Tabriz, Iran
m.fattahi93@ms.tabrizu.ac.ir, karimian@tabrizu.ac.ir

***Abstract*—Within the next few years, unmanned quadrotors are likely to become an important vehicle in humans' daily life. However, their automatic navigation and landing in indoor environments are among the commonly discussed topics in this regard. In fact, the quadrotor should be able to automatically find the landing point from the nearby position, navigate toward it, and finally, land on it accurately and smoothly. In this paper, we proposed a low-cost and thorough solution to this problem by using both bottom-facing and front-facing cameras of the drone. In addition, in the case that vision data were unavailable, inertial measurements alongside a Kalman filter were used to navigate the drone to achieve the promising continuity and reliability. An AR.Drone 2.0 quadrotor, as well as an ArUco marker, were employed in order to test the proposed method experimentally. The results indicated that the drone successfully landed on the predefined position with an acceptable time and accuracy.**

Keywords— Auto Landing; Unmanned Arial Vehicle; Low Cost; Monocular Vision; Pose Estimation; Visual Servoing; Quadrotor

I. INTRODUCTION

For the past ten years, there has been a rapid rise in the use of drones. Meanwhile, demands for automatic landing of drone on a defined position, safely, accurately, and without human's intervention, increase every day. Several papers have been published in the history of the indoor navigation and landing. Many attempts have been made with the purpose of navigation of the quadrotor using both Inertial Measurement Unit (IMU) and Vision sensors [1-3]. The major difference among the different approaches is on their control algorithm, feedback system, or/and selected marker. While some researchers prefer to use PID controller [4-6], others use more complicated controllers. Authors studied landing of a quadrotor using Dynamic Image-Based Visual Servoing [7]. Moreover, scientists simulated automatic landing of a quadrotor using sliding mode control [8], and nonlinear control methods [9].

Furthermore, some researchers utilized infrared camera [10], stereo camera [11], RGB-D sensors [12] or other methods [13-15] to provide the position feedback for the quadrotor. However, we just focused on a low-cost method in order to obtain the position feedback: utilization of the specific markers. It is worth mentioning that selecting a proper marker plays a crucial role in the experiment of the automatic landing. Markers used in [6, 16-20], have an advantage of simplicity, but their symmetrical shape deprives us of estimating drone's 3D position. However, markers used in [21, 22] can estimate the 3D position, but their shapes are unique, i.e. we cannot have several distinctive markers with this configuration. In contrast, ArUco marker [23] has 1024 different arrangements. The marker used in [24] has the same advantages in addition to being multi-level, which allows us to detect it from various distances. The only disadvantage of this marker could be the possibility of the wrong detection of the black areas. AprilTag is also among the most widely used types of markers by researchers [25, 26]. Authors in [27] used markers stuck on the wall for position estimation. On the other hand, some studies have been carried out on specific marker detection process and methods [22, 28], and different landmark situations [29]. Authors in [30, 31] presented a great summary of the relative advantages of different landmark designs and landmark recognition methods.

Landing the quadrotor on a moving target is also an important constituent of the recent research trends [7, 16, 25, 26, 32-34]. Authors in [6, 19] developed a system, in which an AR. Drone Unmanned Arial Vehicle (UAV) lands on a ground vehicle. Similarly, researchers implemented automatic landing of a quadrotor UAV on a car moving at 50 km/h [35]. Gazebo simulation environment has been used by some researchers for the purpose of simulation of automatic landing procedure [24, 36, 37]. Other researchers tried to land a quadrotor on the defined place for recharging purpose either by wireless charger [38] or connecting it to the specific pads [39, 40].

Previous work has mainly focused on the landing problem only when the marker is in the field of view of the vertical camera even though some researchers utilized gimbaled camera [25, 26], or fisheye lens [16] to cover wider area of the land. Our contribution in this article is suggesting a low cost, effective, and thorough solution for landing of the quadrotor using both vision and IMU sensors considering that the quadrotor is far from the target in an indoor area. In contrast to the previous research, we employed IMU data to navigate the drone toward the marker whenever the vision data becomes unavailable. As shown in Fig. 1, first, the quadrotor finds the

978-1-5386-0765-7/17/$31.00 ©2017 IEEE

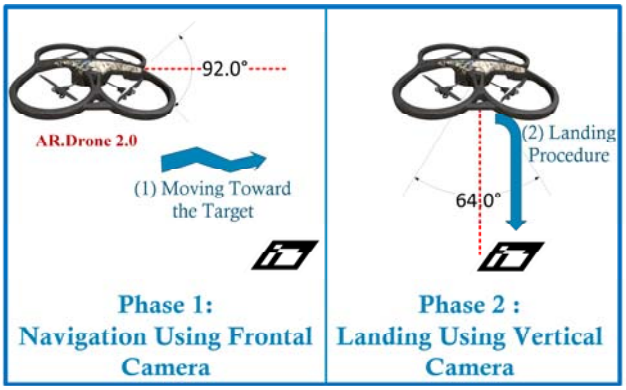

Fig. 1. Various stages of the drone's landing procedure

position of the marker using the frontal camera. Next, it approaches the marker using the vision feedback, IMU sensors, and Kalman filter. Then, the drones' position with respect to (w.r.t) the marker will be estimated as soon as the marker becomes visible in the drone's vertical camera. Finally, it will be aligned with the marker and land on it precisely. The position estimation procedure is discussed in the second section. The third section looks at the control loop and the proposed algorithm for landing using both IMU and vision sensors. Finally, the success of the proposed method is evaluated by the experimental results in section 4.

II. Position Estimation

Every feedback system has its own shortcomings. IMU sensor's data are going to deviate from the real values after a while because of the build-up of the previous errors [41]. On the other hand, vision data suffer from the low data-rate problem or even sometime become unavailable. Therefore, in this research, we gained the benefit of both systems. Deviation errors of the IMU system were compensated by exact data of the vision feedback. Instead, we could navigate using IMU data where the vision feedback was not available.

We needed two reference systems for describing the equations. The first one is the ground coordinate system, which is shown as X_G, Y_G, and Z_G. For the sake of simplicity, we assumed the center of this frame as the center of the marker. The second coordinate system is the coordinate system of the drone's body which is shown as X_c, Y_c, and Z_c. The optical center of the camera was considered as the center of this coordinate frame since the camera was fixed to the drone's body. Finally, the image coordinate system is shown with u and v. All of the three coordinate systems are shown in Fig. 2 (a).

A. Vision Based Position Estimation

1) Marker Detection Procedure and Process

In this research, the well-known "ArUco" marker [23] was used. First, the original image (Fig. 3(a)) was converted to the grayscale image, and then to the binary one using the adaptive threshold method in order to achieve the shortest processing time. The next step was finding the image's contours. The neighborhood border/contour tracing algorithm of Suzuki [42]

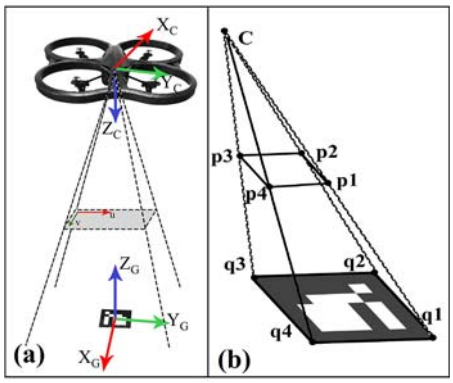

Fig. 2. **a)** all of the three frames: drone's camera frame, image frame, and reference or ground frame, **b)** projection of n points from the real world into the image plan (C is the camera center).

in the OpenCV library was used for this purpose. The returned value of this method was a list of quadrilaterals and each quadrilateral described a contour. Next, the contours which contained the area less than one-fifth of our image columns, as well as the ones containing more or less than four corners, were waived since they probably have no valid markers, and they would not be detectable [43]. After that, threshold of the detected area was calculated again using Otsu's algorithm [44], a more accurate method, to obtain a better quality.

A standard ArUco marker was seen as a 7*7 binary matrix. Each row of inner 5*5 matrix consisted of 2 bits of data and 3 bits of fault detection. This matrix produced a 10-bit code, which enabled us to detect 1024 different markers [37]. Finally, the errors of the codes were calculated in order to find the correct rotation among the 4 possible rotations, since the right rotation should hold a zero error [43]. After that, corners of the markers were found with sub-pixel accuracy with the *cornerSubpixel* function of OpenCV library. However, it is worth mentioning that we could not use this method for all of the contours at the first stage due to its huge process load. Fig. 3 (b) shows the founded marker. The average frequency of processing of the images is 24 Hz here.

2) 3D Position Estimation

Projection of n points in the world coordinate system on the image plane is defined as below:

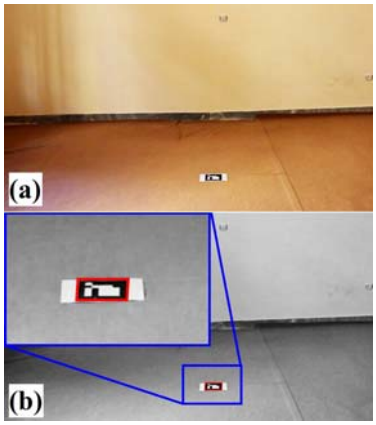

Fig. 3. The original image captured by AR.drone's frontal camera (upper picture), **b)** detected marker (lower picture).

$$sp_i = A[R_G \mid T_G]q_i \tag{1}$$

Where s is the desired scale factor, $q_i(i = 0 \dots n)$ is a point in the 3D coordinates, and p_i is the projection of q_i on the image plan. R_G and T_G are rigid body rotation and translation parameters from the camera frame into the ground frame, respectively. Matrix $[R_G|T_G]$ is called extrinsic parameters matrix. A is the camera matrix or the matrix of intrinsic parameters which is found by the camera calibration procedure [45, 46]. We can rewrite (1) as below:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{pmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{13} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_G \\ Y_G \\ Z_G \\ 1 \end{bmatrix} \tag{2}$$

Where X_G, Y_G and Z_G are the coordinates of the 3D points in the world coordinate system, $(u, v)^T$ are the coordinates of the pixel point projected in the image plane, $(c_x, c_y)^T$ are the coordinates of the image center, r_{ij} is an element of the rotation matrix (R_G), and t_i is an element of the translation matrix (T_G). f_x and f_y are the focal lengths in the pixel scale where they are equal in the ideal conditions. Finally, γ describes how skewed the two image axes are.

To find rotation and translation using the four points found in the image (four corners of the square), four corresponding 3D positions of these points are needed. As can be seen in Fig. 2 (b), the marker is located on the ground plane (X_G Y_G) (the Z_G element is zero) and center of the marker is assumed as *(0, 0, 0)*. $q1 - q4$ are points in the real world and $p1 - p4$, which are coordinates of the corners of the marker in the image plane, were previously found using image processing. So we can find the desired R_G and T_G using (1).

The *solvePnP* function of OpenCV library was used to find the rotation and translation vectors. This process is usually known as the *n* point projection, and the method proposed in [47] was used here. Furthermore, Rodrigues equation was used to obtain a 3*3 rotation matrix from the rotation vector [48].

Now we have translation and rotation of each marker w.r.t the camera. Using the property of rigid body transformation, the new rotation matrix is $R_C = R_G^T$, and the new translation matrix is $T_C = -R_G^T.T_G$, where R_C and T_C are the rigid body rotation and translation parameters from the ground frame to the camera one [37]. Therefore, the joint rotation and translation matrix are:

$$[R_C \mid T_C] = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} & t'_1 \\ r'_{21} & r'_{22} & r'_{23} & t'_2 \\ r'_{13} & r'_{32} & r'_{33} & t'_3 \end{bmatrix} \tag{3}$$

Assuming our marker is at the center of the ground frame, we would have the drone's position and rotation as below:

$$x_{G_{Vision}} = t'_1 \, , y_{G_{Vision}} = t'_2 \, , z_{G_{Vision}} = t'_3 \tag{4}$$

$$\varphi_{G_{Vision}} = \tan^{-1}(\frac{r'_{32}}{r'_{33}}) \, , \theta_{G_{Vision}} = \tan^{-1}(\frac{-r'_{31}}{\sqrt{r'^2_{32} + r'^2_{33}}})$$
$$\psi_{G_{Vision}} = \tan^{-1}(\frac{r'_{21}}{r'_{11}}) \tag{5}$$

Where $x_{G_{Vision}}, y_{G_{Vision}}$ and $z_{G_{Vision}}$ are the drone's position w.r.t the ground frame obtained by the vision, and $\varphi_{G_{Vision}}, \theta_{G_{Vision}}$ and $\psi_{G_{Vision}}$ are the roll, pitch and yaw angles of the drone w.r.t the ground frame.

B. Position estimation using the inertial sensors

Velocity, acceleration, and angle of the drone in all directions were provided by the IMU sensors. In addition, the drone's current position can be easily estimated from its former positions and velocities. Kalman Filter was employed in order to ameliorate noisy measurements of the sensors [49-51]. So, our state vector in step k is x_k as follows:

$$x_k = \begin{bmatrix} x_{G_{IMU}} & y_{G_{IMU}} & z_{G_{IMU}} & \dot{x}_{G_{IMU}} & \dot{y}_{G_{IMU}} & \dot{z}_{G_{IMU}} \end{bmatrix}^T_k \tag{6}$$

Where $x_{G_{IMU}}, y_{G_{IMU}}$ and $z_{G_{IMU}}$ are the drone's position w.r.t the ground frame and $\dot{x}_{G_{IMU}}, \dot{y}_{G_{IMU}}$ and $\dot{z}_{G_{IMU}}$ are the drone's velocity in three directions w.r.t the ground frame. According to the Kalman filter model, state of the system in step k could be obtained from state of the system in step *k-1*, with the equation below:

$$x_k = F_k x_{k-1} + B_k u_k + w_k \, , w_k = \mathrm{N}(0, Q_k) \tag{7}$$

Where F_k is the state transition matrix, B_k is the control-input matrix, and w_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k. The state transition matrix (F_k) is defined as below:

$$F_k = \begin{bmatrix} 1 & \cdots & dt\mathrm{I}_3 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \tag{8}$$

Where $dt = 0.041\ ms$ is the sampling time. A measurement of the state x_k at time k is performed according to the following equation:

$$m_k = H_k x_k + v_k \, , \; v_k = \mathrm{N}(0, R_k) \, , \; H_k = \begin{bmatrix} 0_{4,3} \mid \mathrm{I}_{4,4} \end{bmatrix} \tag{9}$$

Where H_k is the observation matrix and v_k is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_k. We assumed Q_k and R_k of the Kalman filter, which were obtained from the experimental tests, as below:

$$Q_k = \begin{bmatrix} (0.1)\mathrm{I}_3 & \mathrm{I}_3 \\ 0_3 & (0.3)\mathrm{I}_3 \end{bmatrix},\ R_k = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix} \quad (10)$$

The vector of the measured velocity of the drone (V_C) w.r.t the local frame is as below:

$$V_C = \begin{bmatrix} \dot{x}_{C_{IMU}} & \dot{y}_{C_{IMU}} & \dot{z}_{C_{IMU}} \end{bmatrix}^T \quad (11)$$

For transforming these vectors to the ground frame, we should multiply them with the rotation matrix. This matrix has been made up of the product of multiplying three other matrixes $R(\varphi), R(\theta)$, and $R(\psi)$, where $R(\varphi)$ represents rotating around its own longitudinal axis, $R(\theta)$ represents rotating around its own transverse axis, and $R(\psi)$ represents rotating around an axis which is perpendicular to the other two axes [52], and φ, θ and ψ are obtaining from the gyroscope sensor of AR.Drone. So, the velocity vector, w.r.t the ground frame (V_G), would be as below:

$$V_G = R(\psi) * R(\theta) * R(\varphi) * V_C \quad (12)$$

Finally, the measurement vector for the Kalman filter would be as below:

$$m_k = \begin{bmatrix} z_{G_{IMU}} & V_G \end{bmatrix}^T = \begin{bmatrix} z_{G_{IMU}} & \dot{x}_{G_{IMU}} & \dot{y}_{G_{IMU}} & \dot{z}_{G_{IMU}} \end{bmatrix}^T \quad (13)$$

Therefore, if the vision data is available, position of the drone would be estimated from marker, otherwise, the drone would estimate its position using IMU sensors' data.

III. Navigation Control Algorithm of Auto Landing Procedure

First, the drone was far from the landing area, and the marker was in the frontal camera's sight of view. The position of the drone would be calculated w.r.t the marker, and the drone would move toward the marker. One of the famous controllers that can be used for this purpose is PID controller. To apply the PID controller, a desired point $X_d = (x_{d_G}, y_{d_G}, z_{d_G})$, is defined above the marker. Assuming $X = (x_G, y_G, z_G)$ as the current position of the drone, the PID controller is applied to the position error $E = X_d - X$, where $E = (e_x, e_y, e_z)$, and e_x, e_y, and e_z are the position errors in directions X_G, Y_G, and Z_G, respectively. Equations of this phase are as follows:

$$V_x = K_{px} e_x - K_{dx} \frac{dx_G}{dt} + K_{ix} \int_0^t e_x dt \quad (14)$$

$$V_y = K_{py} e_y - K_{dy} \frac{dy_G}{dt} + K_{iy} \int_0^t e_y dt\ ,\ V_z = K_{pz} e_z \quad (15)$$

Pseudo Code 1: Auto Landing Algorithm

```
While 1 do
    frame ← get-frame();
    (x_GVision, y_GVision, z_GVision) ← pose-estimation (frame);
    (x_GIMU, y_GIMU, z_GIMU) ← Kalman-filter ();
    if marker is detected then
        (x_G, y_G, z_G) ← (x_GVision, y_GVision, z_GVision); l ← 0;
    else
        l++;
        if l > 5 then
            change to the bottom-facing camera;
        end if
        (x_G, y_G, z_G) ←(x_GIMU, y_GIMU, z_GIMU);
    end if
    (V_x, V_y) ← PID-calculate{ (0, 0), (x_G, y_G)};
    if −0.05 < x_G < 0.05 and −0.05 < y_G < 0.05 then
        V_z ← −0.8;
        if z_G < 0.2 then
        turn off motors;
        end if
    end if
    move-drone (V_x, V_y, V_z);
end while
```

Where V_x, V_y and V_z are the linear velocities which are applied to the drone in directions X_G, Y_G, and Z_G, respectively. Furthermore, K_p, K_d and K_i are the proportional, integral, and differential coefficients of the PID controller, respectively, which were obtained in the experimental examinations.

The following conditions should be satisfied before the quadrotor could decrease its height: 1) the quadrotor should be above the center of the marker, 2) speed of the drone in any direction should be zero in order to stop reducing the height when the drone is moving. The drone will gradually decrease its height as soon as the above conditions are satisfied. In this step, an ultrasonic, barometer and vision sensors were used altogether to achieve the minimum error.

The overall algorithm of the auto landing procedure is briefly presented in pseudo code 1. Experimental results showed that the drone was uncontrollable in the heights less than 20 cm due to the wind effect of the drone's propellers on the ground and its reaction, which is known as the near ground effect. Thus, according to the experiments and similar to [24], motors were turned off at the height of 20 cm, and the drone did not suffer any damages.

IV. Experimental Results

Even though many researchers prefer to use their own costly custom set up of quadrotor, we, similar to [6, 15, 18, 24, 27], use AR. Drone quadrotor for its relatively lower cost and also to maintain the consistency with the previous research on this topic. The frontal camera of the AR.Drone 2.0 quadrotor has the HD quality and 92 degree field of view, and its vertical camera has the QVGA quality and 64 degree field of view. Fig. 4 shows AR.Drone 2.0 while it was navigating toward the marker using the frontal camera. The drone was controlled online from the ground station automatically, and a DELL Studio 1558 laptop with Core i-7 Q720 CPU, 6 GB of Ram, and Ubuntu Linux 14.04 LTS Operating System was used for

implementing the ground station program.

Auto landing application contained various subprograms, (such as framing and sending data, receiving sensor data, H264 decoder, vision-based position estimation, Kalman filter, PID controller, and etc.) each of which was processed in different threads to enhance the efficiency. The marker size was 130 mm in all of the experiments.

Fig. 5 shows the complete procedure of navigation and landing of a quadrotor. First, the drone estimated its position w.r.t the marker using the frontal camera. Its initial position was $(-2.3\ m, -1.5\ m, 0.75\ m)$ and it started to navigate toward the marker. Whenever the marker became invisible, like t= 5~7 s, drone still kept navigating thanks to the IMU sensor values. From t=7 s marker turned to be visible in the sight of the view of the vertical camera and the drone moved to adjust the marker in its center. Finally, the quadrotor started to decrease its height at 12th second and the task was completed in 16th second by the landing of the drone on the marker. This experiment was repeated 10 times; the average time of the complete landings was 16.5 seconds and the average distance from the center of the drone to center of the marker after successful landing was 6 cm. Some researchers reported 10 cm [26], 5 cm [39], and 25 cm [33] as landing error, and 10 s [26], and 14 s [17] as landing time of their experiments. However, there is no consistent method of testing the automatic landing in the literature, so we could not compare the results exactly.

V. Conclusion

In summary, we have presented a complete solution for indoor navigation and landing of a low-cost quadrotor. First, the accurate position of the drone was calculated using vision feedback from a specific marker named ArUco. Then, the quadrotor flew toward the marker having both vision and IMU sensors' feedbacks. In the case that vision based position estimation became unavailable, the drone still could navigate toward the defined position using IMU sensors. As soon as the marker became visible by the vertical camera, the drone estimated its position w.r.t the marker more accurately, and moved above the marker. Next, the quadrotor landed on it

Fig. 4. The AR.Drone 2.0 quadrotor navigating toward an Aruco marker

gradually and accurately with the maximum position error of 6 cm. Meanwhile, The Kalman filter and the PID controller were used in order to achieve better performance. The experimental tests which were conducted using a Low-cost commercial quadrotor named AR.Drone 2.0, was declared acceptable performance of the proposed algorithm. Our method could be applied to a variety of applications in which quadrotor needs to navigate short distances and land accurately in GPS-denied areas, especially when other expensive and sophisticated methods could not be used.

References

[1] M. Brossard, S. Bonnabel, and A. Barrau, "Unscented Kalman Filtering on Lie Groups for Fusion of IMU and Monocular Vision," 2017.

[2] A. Gautam, P. B. Sujit, and S. Saripalli, "Autonomous Quadrotor Landing Using Vision and Pursuit Guidance," IFAC-PapersOnLine, vol. 50, pp. 10501-10506, 2017/07/01/ 2017.

[3] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," IEEE Robotics and Automation Letters, vol. 2, pp. 404-411, 2017.

[4] S. Khatoon, M. Shahid, and H. Chaudhary, "Dynamic modeling and stabilization of quadrotor using PID controller," in Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on, 2014, pp. 746-750.

[5] V. Lippiello, F. Ruggiero, and D. Serra, "Emergency landing for a quadrotor in case of a propeller failure: A pid based approach," in Safety, Security, and Rescue Robotics (SSRR), 2014 IEEE International Symposium on, 2014, pp. 1-7.

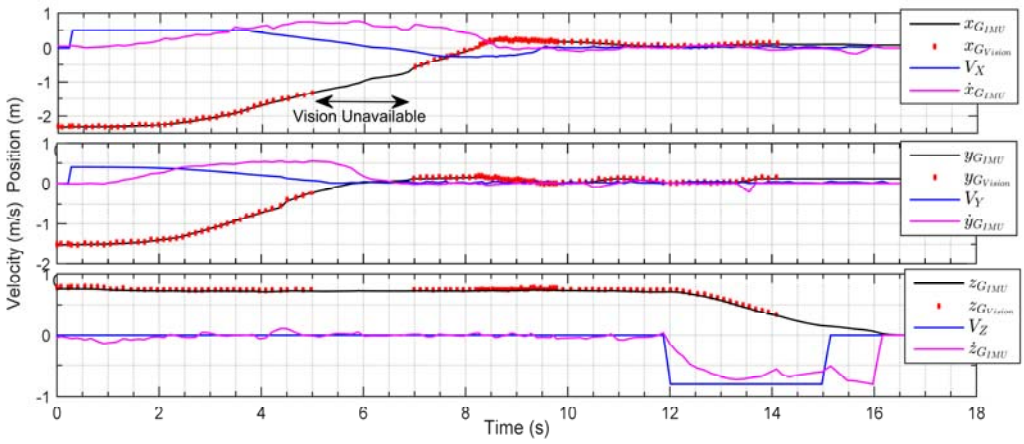

Fig. 5. Complete procedure of automatic landing: position of the drone obtained from IMU and Vision, applied velocity, and real velocity of the drone in direction X (upper chart), Y (middle chart), and Z (lower chart), respectively.

[6] Y. Bi and H. Duan, "Implementation of autonomous visual tracking and landing for a low-cost quadrotor," Optik-International Journal for Light and Electron Optics, vol. 124, pp. 3296-3300, 2013.
[7] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing of a quadrotor on a moving target using dynamic image-based visual servo control," IEEE Transactions on Robotics, vol. 32, pp. 1524-1535, 2016.
[8] D. V. Rao and T. H. Go, "Automatic landing system design using sliding mode control," Aerospace Science and Technology, vol. 32, pp. 180-187, 2014.
[9] J. M. Daly, Y. Ma, and S. L. Waslander, "Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays," Autonomous Robots, vol. 38, pp. 179-191, 2015.
[10] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle," Journal of intelligent & robotic systems, vol. 61, pp. 221-238, 2011.
[11] K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, "Efficient Optical flow and Stereo Vision for Velocity Estimation and Obstacle Avoidance on an Autonomous Pocket Drone," IEEE Robotics and Automation Letters, vol. 2, pp. 1070-1076, 2017.
[12] W. G. Aguilar, G. A. Rodríguez, L. Álvarez, S. Sandoval, F. Quisaguano, and A. Limaico, "Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing," in International Work-Conference on Artificial Neural Networks, 2017, pp. 596-606.
[13] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle," Journal of Field Robotics, vol. 33, pp. 431-450, 2016.
[14] S. Yang, S. A. Scherer, X. Yi, and A. Zell, "Multi-camera visual SLAM for autonomous navigation of micro aerial vehicles," Robotics and Autonomous Systems, vol. 93, pp. 116-134, 2017.
[15] A. Balasubramanian and A. Ganesan, "Vision-Based Heading and Lateral Deviation Estimation for Indoor Navigation of a Quadrotor," IETE Journal of Research, pp. 1-7, 2017.
[16] H. Lee, S. Jung, and D. H. Shim, "Vision-based UAV landing on the moving vehicle," in Unmanned Aircraft Systems (ICUAS), 2016 International Conference on, 2016, pp. 1-7.
[17] W. Bai, F. Pan, B. Y. Xing, C. Pan, and M. X. Pei, "Visual landing system of UAV based on ADRC," in Control And Decision Conference (CCDC), 2017 29th Chinese, 2017, pp. 7509-7514.
[18] R. v. Ginkel, I. Meerman, and J. Peters, "Autonomous Landing of a Quadcopter on a Predefined Marker," July 5, 2013.
[19] M. Saska, T. Krajnik, and L. Pfeucil, "Cooperative μUAV-UGV autonomous indoor surveillance," in Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on, 2012, pp. 1-6.
[20] O. Garcia, D. Flores, O. Santos, H. Romero, S. Salazar, and R. Lozano, "Autonomous take-off and landing on a colored platform," in Unmanned Aircraft Systems (ICUAS), 2017 International Conference on, 2017, pp. 877-884.
[21] C. S. S. O. S. S. S. Sastry, "A Vision System for Landing an Unmanned Aerial Vehicle," in International Conference on Robotics & Automation, Seoul, Korea, May 21-26, 2001, pp. 1720-1727.
[22] S. Shah, "Real-time Image Processing on Low Cost Embedded Computers," Techincal report No. UCB/EECS-20142014.
[23] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," Pattern Recognition, vol. 47, pp. 2280-2292, 2014.
[24] P. Benavidez, J. Lambert, A. Jaimes, and M. Jamshidi, "Landing of an ardrone 2.0 quadcopter on a mobile base using fuzzy logic," in 2014 World Automation Congress (WAC), 2014, pp. 803-812.
[25] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny, "Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle," arXiv preprint arXiv:1611.07329, 2016.
[26] Z. Wang, H. She, and W. Si, "Autonomous landing of multi-rotors UAV with monocular gimbaled camera on moving vehicle," in Control & Automation (ICCA), 2017 13th IEEE International Conference on, 2017, pp. 408-412.
[27] L. V. Santana, A. S. Brandão, and M. Sarcinelli-Filho, "Navigation and cooperative control using the ar. drone quadrotor," Journal of Intelligent & Robotic Systems, vol. 84, pp. 327-350, 2016.
[28] J. Garcia-Pulido, G. Pajares, S. Dormido, and J. M. de la Cruz, "Recognition of a landing platform for unmanned aerial vehicles by using computer vision-based techniques," Expert Systems with Applications, vol. 76, pp. 152-165, 2017.
[29] T. Nakamura, D. Magree, and E. N. Johnson, "Estimation Techniques in Robust Vision-Based Landing of Aerial Vehicles," IFAC-PapersOnLine, vol. 50, pp. 11664-11669, 2017/07/01/ 2017.
[30] Y. Chen and H.-L. Liu, "Overview of landmarks for autonomous, vision-based landing of unmanned helicopters," IEEE Aerospace and Electronic Systems Magazine, vol. 31, pp. 14-27, 2016.
[31] S. Jin, J. Zhang, L. Shen, and T. Li, "On-board vision autonomous landing techniques for quadrotor: A survey," in Control Conference (CCC), 2016 35th Chinese, 2016, pp. 10284-10289.
[32] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, Z. Milosevic, A. Garcia-Vaquero, and P. Campoy, "Towards fully autonomous landing on moving platforms for rotary Unmanned Aerial Vehicles," in Unmanned Aircraft Systems (ICUAS), 2017 International Conference on, 2017, pp. 170-178.
[33] I. Borshchova and S. O'Young, "Marker-guided auto-landing on a moving platform," International Journal of Intelligent Unmanned Systems, vol. 5, 2017.
[34] T. Hoang, E. Bayasgalan, Z. Wang, G. Tsechpenakis, and D. Panagou, "Vision-based target tracking and autonomous landing of a quadrotor on a ground vehicle," in American Control Conference (ACC), 2017, 2017, pp. 5580-5585.
[35] A. Borowczyk, D.-T. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny, "Autonomous Landing of a Quadcopter on a High-Speed Ground Vehicle," Journal of Guidance, Control, and Dynamics, 2017.
[36] C. Yu, J. Cai, and Q. Chen, "Multi-resolution visual fiducial and assistant navigation system for unmanned aerial vehicle landing," Aerospace Science and Technology, vol. 67, pp. 249-256, 2017.
[37] T. G. Carreira, "Quadcopter automatic landing on a docking station," Master's thesis, Instituto Superior Técnico, 2013.
[38] A. B. Junaid, Y. Lee, and Y. Kim, "Design and implementation of autonomous wireless charging station for rotary-wing UAVs," Aerospace Science and Technology, vol. 54, pp. 253-266, 2016.
[39] F. Cocchioni, E. Frontoni, G. Ippoliti, S. Longhi, A. Mancini, and P. Zingaretti, "Visual based landing for an unmanned quadrotor," Journal of Intelligent & Robotic Systems, vol. 84, pp. 511-528, 2016.
[40] F. Cocchioni, V. Pierfelice, A. Benini, A. Mancini, E. Frontoni, P. Zingaretti, et al., "Unmanned ground and aerial vehicles in extended range indoor and outdoor missions," in Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, 2014, pp. 374-382.
[41] L. Lasmadi, A. I. Cahyadi, R. Hidayat, and S. Herdjunanto, "Inertial Navigation for Quadrotor Using Kalman Filter with Drift Compensation," International Journal of Electrical and Computer Engineering, vol. 7, p. 2596, 2017.
[42] S. Suzuki, "Topological structural analysis of digitized binary images by border following," Computer Vision, Graphics, and Image Processing, vol. 30, pp. 32-46, 1985.
[43] D. L. Baggio, Mastering OpenCV with practical computer vision projects: Packt Publishing Ltd, 2012.
[44] N. Otsu, "A threshold selection method from gray-level histograms," Automatica, vol. 11, pp. 23-27, 1975.
[45] M. Magnusson, "Short on camera geometry and camera calibration," 2010.
[46] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on pattern analysis and machine intelligence, vol. 22, pp. 1330-1334, 2000.
[47] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," International journal of computer vision, vol. 81, pp. 155-166, 2009.
[48] O. Faugeras, Three-dimensional computer vision: a geometric viewpoint: MIT press, 1993.
[49] R. E. Kalman, "A new approach to linear filtering and prediction problems," Journal of basic Engineering, vol. 82, pp. 35-45, 1960.
[50] P. S. Maybeck, Stochastic models, estimation, and control vol. 3: Academic press, 1982.
[51] G. Bradski and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library: " O'Reilly Media, Inc.", 2008.
[52] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," Matrix, vol. 58, pp. 1-35, 2006.