Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 1821 – 1838

doi:10.3906/elk-1809-204

Automatic landing of a low-cost quadrotor using monocular vision and Kalman filter in GPS-denied environments

Mohammad Fattahi SANI[1], **Maryam SHOARAN**[2,*], **Ghader KARIMIAN**[1]

[1]Department of Electrical Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

[2]Department of Mechatronics Engineering, School of Engineering-Emerging Technologies, University of Tabriz, Tabriz, Iran

Received: 30.09.2018 • **Accepted/Published Online:** 11.02.2019 • **Final Version:** 15.05.2019

Abstract: Unmanned aerial vehicles are becoming an important part of the modern life. Despite some recent advances in GPS-aided navigation of quadrotors, the concern of crash and collision still overshadows their reliability and safety, especially in GPS-denied environments. Therefore, the necessity for developing fully automatic methods for safe, accurate, and independent landing of drones increases over time. This paper investigates the autolanding process by focusing on an accurate and continuous position estimation of the drone using a monocular vision system and the fusion with the inertial measurement unit and ultrasonic sensors' data. An ARUCO marker is used as the landing pad, and the information is processed in the ground station through a real-time Wi-Fi link. In order to overcome the closed loop instability caused by the communication and localization delays, we propose a method called "movement slicing method". This method divides the moves around the marker into moving and waiting slices and makes the landing process not only more accurate but also faster. Experimental results show a successful landing of the UAV on a predefined location, while it is accurately aligned with the marker using the proposed method.

Key words: Autolanding, unmanned aerial vehicle, quadrotor, monocular vision, pose estimation, low cost

1. Introduction

Unmanned aerial vehicles (UAVs) have recently attracted a great attention among aerospace, control, and robotic researchers. Quadrotors are one of the most useful types of UAVs and have been widely used in various research projects due to their simplicity, great controllability, and vertical take-off and landing capabilities. Manual landing of a quadrotor usually has some difficulties. For instance, it is hard for a pilot to reduce the height of the quadrotor gradually and land it smoothly and accurately on an intended location. Moreover, in most cases, we need to land the quadrotor without any human involvement. Although automatic landing of a quadrotor can be done using GPS feedback, the landing in GPS-denied areas is still an open research subject. In this paper, we propose a new method for automatic landing of UAVs which uses a monocular vision system for precise positioning of the UAV. Increased landing accuracy, safety, and reliability is the main advantage of our proposed method.

In the literature, simulation of various control methods such as sliding mode [1] and nonlinear controllers [2] has been studied. Some researchers studied on landing quadrotors on predefined places and charging them via wireless chargers [3] or by connecting them to some specific landing platforms [4, 5]. In [6, 7], the infrared camera

*Correspondence: mshoaran@tabrizu.ac.ir

is used for tracking the quadrotor. IR-Lock system[1] has been also introduced as an off-the-shelf solution for precise tracking and landing of UAVs. However, it has some shortcomings such as limited accuracy [8], necessity of another dedicated infrared camera, and finally having a marker which consumes electrical power. On the other hand, landing on a moving target is also an important research subject. Authors in [9–11] developed a system, in which an AR.Drone UAV lands on a ground vehicle. Other researchers studied automatic landing of a PID-controlled low-cost quadrotor [12], or used H-shaped markers on pushcart carriers as the landing target [13]. However, the direction of the quadrotor was not considered in these studies. Researchers in [14] intended to land the quadrotor while it was aligned with the marker. In some other studies, the automatic landing was simulated in the Gazebo simulation environment [15–17]. Furthermore, load transporting problem in UAVs was investigated using neural networks in [18], where the take-off, landing, and target tracking missions were all carried out autonomously in real time. A report from Berkeley University [19] mostly focuses on the optimization of a real-time image processing hardware and software, including multithread processing and pattern recognition libraries. Authors in [20, 21] tried to determine a safe landing area for a quadrotor using several methods.

Clearly, some of the aforementioned solutions suffer from being expensive, or some others do not focus on time efficiency and accuracy of the landing process. Therefore, in this article we implement an effective solution for automatic landing of an AR.Drone 2.0 quadrotor on a predefined marker in the presence of closed loop delays. As shown in Figure 1, first, the quadrotor approaches the marker using the inertial measurement unit (IMU) sensor and the Kalman filter. Then, the drone position w.r.t. the marker is estimated once the marker becomes visible in the drone camera. Finally, it aligns with the marker and lands on it. We illustrate different types of landing markers in Section 2. The position estimation with IMU and vision sensors are discussed in Section 3. In Section 4, the control loop and the proposed algorithm for landing using both IMU and vision sensors are explained. Finally, the proposed method is evaluated by the experiments illustrated in Section 5.

2. Landing markers

One of the key points in vision-based landing is to use a proper marker. Various types of markers used in the former studies are shown in Figure 2. Markers a and b, which were used in [22] and [23], have an advantage of simplicity, but their symmetrical shapes do not let us estimate 3D position. Marker c, which was used in [13, 14], and also marker f used in [9], are good symbols for the landing pad due to their distinguishable shapes and colors, but they are also symmetric. We can estimate the height using marker e by detecting different sizes of circles [24]. Marker d in [25] consists of two colors for better detection, and marker g used in [19, 26], is a little more complex. Despite the fact that both of the markers d and e enable us to perform 3D pose estimation, their unique configuration prevents us from having multiple markers with different ID codes. Marker h (ARUCO marker [27]), however, can produce 1024 markers with various ID numbers. Marker i (AprilTag marker) which is utilized by the researchers in [17] is a multilevel marker and can be detected from a wide range of distances. Detection of these markers could be very challenging though, due to the presence of small black and white features. Researchers in [28] conducted a useful survey on various markers and approaches to recognize them. In this research, we use ARUCO marker.

3. Position estimation

We need a reliable feedback of the states in order to control an unstable system like a quadrotor. Due to the IMU sensor's aggregated error through time, we use both vision and IMU feedbacks for more accuracy. There

[1]IR-LOCK (2019) infrared tracking systems for drones [online]. Website https://irlock.com [Accessed on 01/01/2019].

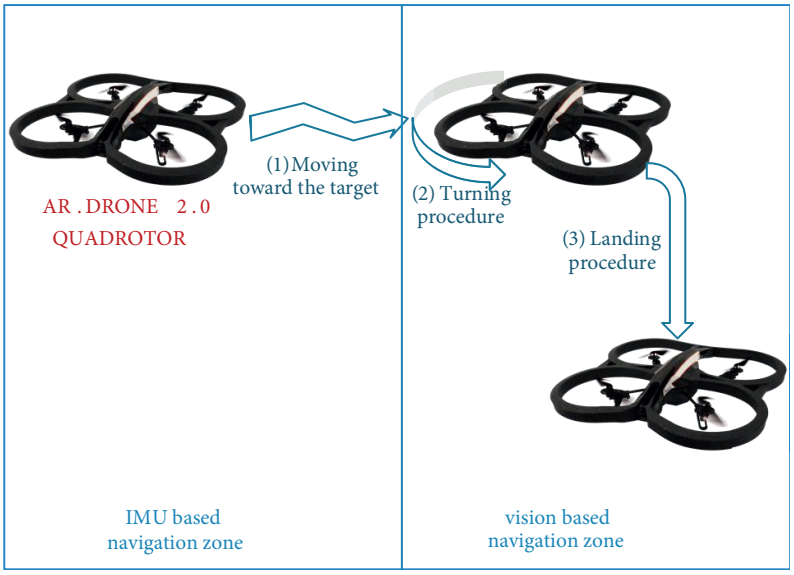

Figure 1. Various stages of drone navigation.

are two reference systems to formulate the quadrotor's equations, (1) the ground coordinate system X_G, Y_G, Z_G, and (2) the drone's body coordinate system X_c, Y_c, Z_c. The center of the marker is assumed to be the center of the ground coordinate system, while optical center of the camera is assumed to be the center of the drone's coordinate system. We use u and v in this paper to represent the image coordinate system. Figure 3 illustrates the aforementioned coordinate systems.

3.1. Position estimation using inertial sensors

The IMU sensor is used to provide the velocity, the acceleration, and the angle of the drone in all directions. We can estimate the drone's current position using dead reckoning [29] and the drone's former positions and velocities. The Kalman Filter is employed in order to improve measurements of the IMU sensor [30–32]. Therefore, the state vector in step k, x_k, is as follows:

$$x_k = \begin{bmatrix} x_{G_{IMU}} & y_{G_{IMU}} & z_{G_{IMU}} & \dot{x}_{G_{IMU}} & \dot{y}_{G_{IMU}} & \dot{z}_{G_{IMU}} \end{bmatrix}_k^T, \tag{1}$$

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, the state of the system in step k could be obtained from the state of the system in step $k-1$, with the following equation,

$$x_k = F_k x_{k-1} + B_k u_k + w_k, \tag{2}$$

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 a zero mean Gaussian white noise with covariance Q_k.

$$w_k = \mathrm{N}(0, Q_k) \tag{3}$$

We define the state transition matrix (F_k), and the observation matrix (H_k) as in the following,

$$F_k = \begin{bmatrix} I_3 & dtI_3 \\ O_3 & I_3 \end{bmatrix}, \; H_k = \begin{bmatrix} O_{4\times 2} & I_4 \end{bmatrix} \tag{4}$$

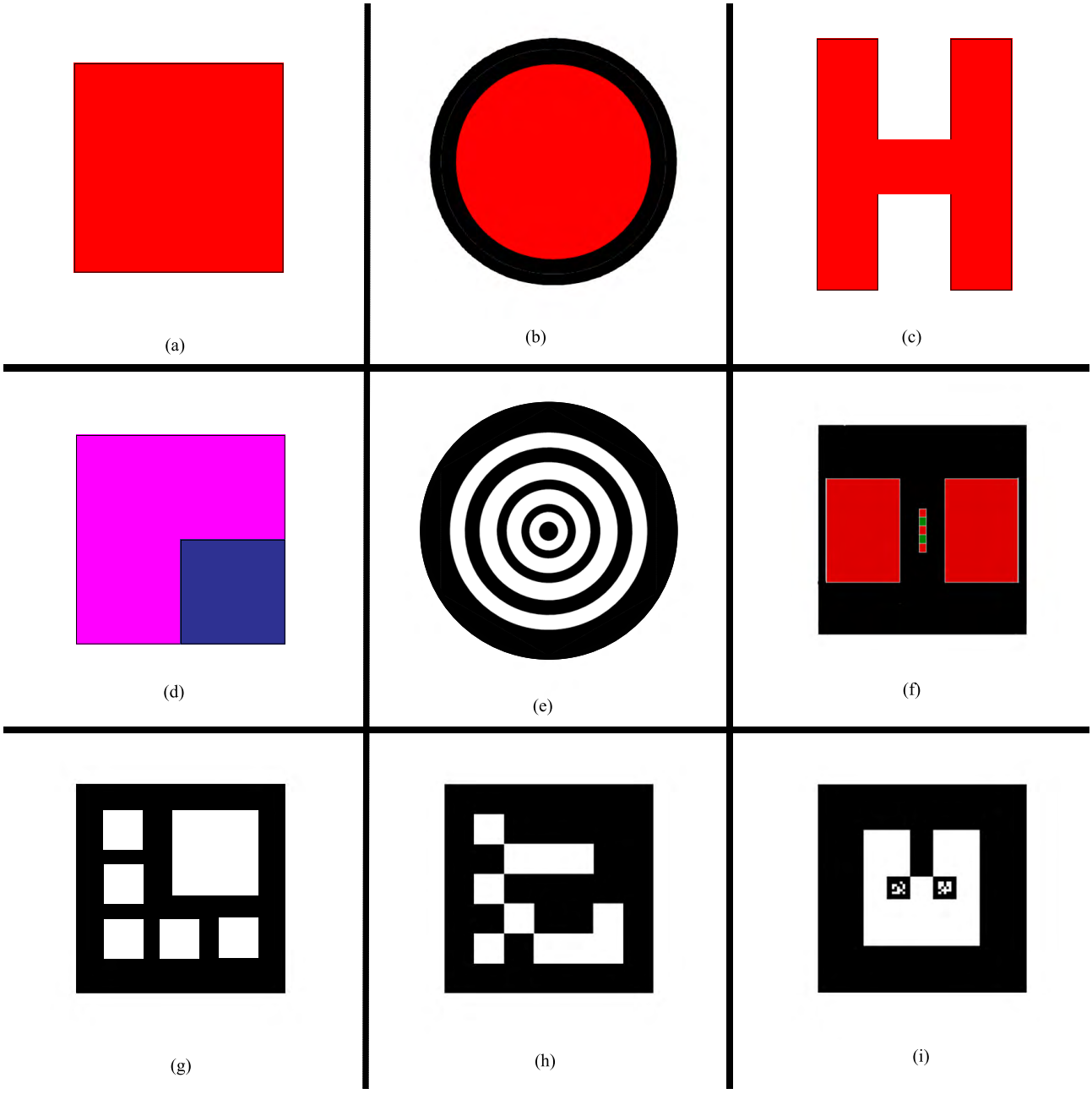

Figure 2. Various types of landing markers used in different studies.

where $dt = 0.041$ s is the sampling time, I_x is $x \times x$ identity matrix, and O_x is $x \times x$ zero matrix. A measurement of the state x_k at step k is performed according to the following equation:

$$m_k = H_k x_k + v_k, \tag{5}$$

where v_k is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_k.

$$v_k = \mathrm{N}(0, R_k). \tag{6}$$

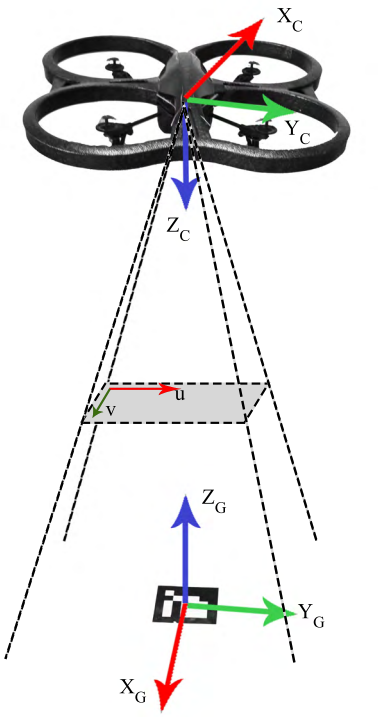

Figure 3. All three frames: drone's camera frame, image frame, and reference or ground frame

We assume that Q_k and R_k of the Kalman filter obtained from the experimental tests are as follows.

$$Q_k = \begin{bmatrix} 0.1I_3 & O_3 \\ O_3 & 0.3I_3 \end{bmatrix}, \; R_k = \begin{bmatrix} 0.1I_2 & O_2 \\ O_2 & 0.05I_2 \end{bmatrix} \tag{7}$$

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

$$V_C = \begin{bmatrix} \dot{x}_{C_{IMU}} & \dot{y}_{C_{IMU}} & \dot{z}_{C_{IMU}} \end{bmatrix}^T. \tag{8}$$

In order to transform these vectors to the ground frame, we have to multiply them by the rotation matrix. This matrix is formed through multiplying three other matrices $R(\varphi), R(\theta)$, and $R(\psi)$, where $R(\varphi)$ represents rotation around its own longitudinal axis, $R(\theta)$ denotes rotation around its own transverse axis, and $R(\psi)$ indicates rotation around an axis which is perpendicular to the other two axes [33], and φ, θ, and ψ are given by the AR.Drone's gyroscope sensor. Therefore, the velocity vector w.r.t. the ground frame (V_G) is as follows.

$$V_G = R(\psi) * R(\theta) * R(\varphi) * V_C. \tag{9}$$

Finally, the measurement vector for the Kalman filter will be as follows.

$$m_k = \begin{bmatrix} z_{G_{IMU}} & V_G \end{bmatrix}^T = \begin{bmatrix} z_{G_{IMU}} & \dot{x}_{G_{IMU}} & \dot{y}_{G_{IMU}} & \dot{z}_{G_{IMU}} \end{bmatrix}^T. \tag{10}$$

3.2. Position estimation using visual signs

In this section, first, we address the methods used for detection and identification of the marker. Then, vision-based position estimation is explained.

3.2.1. Detection and identification of the landing marker

We use the ARUCO marker [27] in this study, which is very popular in augmented reality projects for estimating the position of the camera. The RGB image (Figure 4a) provided by the drone's camera is converted to a gray-scale one (Figure 4b) at the first stage. Then, the gray-scale image is converted to a binary image (Figure 4c) through applying an adaptive threshold method. This process is performed in order to achieve a smaller processing time.

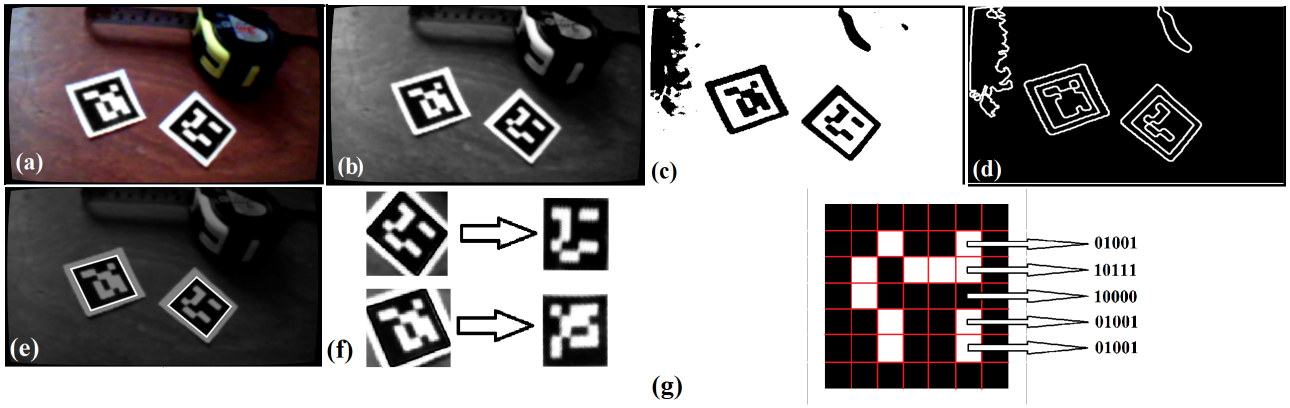

Figure 4. (a) Original image captured by AR.drone's downward camera, (b) gray-scale converted image, (c) binary image, (d) detected contours, (e) rotation and perspective correction of detected markers, (f) the zoned marker with the code of 586, and (g) detected markers.

To detect the marker contour detection algorithm explained in [34] is applied to the binary image. Figure 4d shows the results. This method returns a list of detected contours described by quadrilaterals. Next, the contours which are five times smaller than the original image are filtered out because they most probably do not contain any valid or detectable markers [35]. Similarly, after finding contours, if they have more or less than four corners, definitely they are not the appropriate ones. The effect of perspective projection should be compensated before we could detect the marker's code. Therefore, the perspective transformation of the image is found using four corresponding pairs of points, and then, this transformation is applied to our marker to make it a frontal view as shown in Figure 4e. At this stage, we apply another threshold function on the candidate marked area. This time we employ Otsu's algorithm which is more accurate [36]. This algorithm assumes a bimodal distribution and finds the threshold value that maximizes the extraclass variance while keeping a low intraclass variance. A marker is seen as a 7×7 binary matrix. The outer blocks form a black square so that it can easily be detected with image processing methods. Remaining parts form a 5×5 matrix as shown in Figure 4f, each row of which is composed of five bits. Just two bits out of five are data bits and the other three bits exist for the purpose of fault detection. Fault detection is done using a method similar to the hamming code. The difference between the hamming code and this code is in the first bit which is inverted here. So a zero line, which has the hamming code of 00000, is coded as 10000 here to prevent a blank square of becoming valid. This is due to the fact that a blank square could be easily found in the real world and we do not want to detect it as a marker. Eventually, we have 10 bit code as an outcome of the aforementioned procedure, using which we can identify 1024 various IDs of ARUCO markers [16]. At this point, correct rotation of the marker could be found by comparing the error values in each line because the proper rotation of the marker should end up having no error at all [35]. Now another corner finder function with subpixel accuracy is applied to the candidate contours to achieve a better performance. Figure 4g shows the contours found.

3.2.2. 3D position estimation

In this section, we are going to find the rotation and translation matrices between drone's camera frame and the reference or ground frame using the images taken by the drone's vision system. The following equation defines how the points in the image plane are the projections of the corresponding points in the world coordinate system (11),

$$sp_i = A[R_G|T_G]q_i, \tag{11}$$

where s is an arbitrary scale factor, q_i(i=0...n) is an arbitrary point in the world coordinates, and p_i is its projection in the image plane. R_G shows the rotation of a rigid body from camera to the world frame and T_G is the translation vector for camera-to-world frame. The joint matrix of $[R_G|T_G]$ is the camera's extrinsic parameters, whereas A is usually called the intrinsic parameters of the camera [37]. Matrix A can be found through the camera calibration. Eq. (11) can be reformed as follows,

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

where X_G, Y_G, Z_G, and $(u, v)^T$ represent a point in 3D world coordinate and its projection in 2D image coordinate system, respectively. $(c_x, c_y)^T$ denotes the center of the image. r_{ij} represents an element from (R_G), whereas t_i represents an element from (T_G). f_x and f_y stand for focal lengths in the scale of pixel. Finally, γ is the skew factor.

Given that the marker is placed at the center of the ground frame, the coordinates of four corners of the marker are known $(q1–q4)$. On the other hand, four corresponding corners of the marker in the image plan $(p1–p4)$ are already found using the image processing step. Figure 5 better illustrates this fact. Therefore, R_G and T_G can be found using Eq. (11) and the EPnP method proposed in [38].

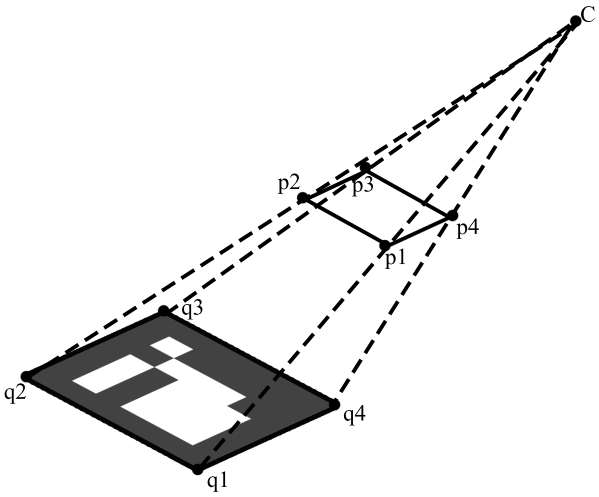

Figure 5. Projection of n points from the real world into the image plan (C is the camera center).

To convert a rotation vector into a rotation matrix with the same number of rows, Rodrigues' equation is utilized [16]. After calculating R_G and T_G of the marker frame w.r.t. the camera frame, we have to find R_G and T_G of the camera frame w.r.t. the world frame. Given that D is any vector in the camera frame, by applying transformation on this vector we have:

$$D' = R_G D + T_G, \tag{13}$$

where D' is the transformed matrix of D in the ground frame. By reforming (13), we have:

$$D = R_G^T(D' - T_G),\ D = R_G^T D' - R_G^T T_G. \tag{14}$$

Thus, $R_C = R_G^T$ and $T_C = -R_G^T.T_G$ represent rotation and translation matrices from the ground frame to the camera frame, respectively [16]. Thus, we have:

$$[R_C|T_C] = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} & t'_1 \\ r'_{21} & r'_{22} & r'_{23} & t'_2 \\ r'_{31} & r'_{32} & r'_{33} & t'_3 \end{bmatrix}. \tag{15}$$

Now quadrotor's position and rotation could be found as follows:

$$x_{G_{Vision}} = t'_1,\ y_{G_{Vision}} = t'_2,\ z_{G_{Vision}} = t'_3,\ \varphi_{G_{Vision}} = \tan^{-1}(\frac{r'_{32}}{r'_{33}}), \tag{16}$$

$$\theta_{G_{Vision}} = \tan^{-1}(\frac{-r'_{31}}{\sqrt{(r'_{32})^2 + (r'_{33})^2}}),\ \psi_{G_{Vision}} = \tan^{-1}(\frac{r'_{21}}{r'_{11}}), \tag{17}$$

where $x_{G_{Vision}}, y_{G_{Vision}}$, $z_{G_{Vision}}$, and $\varphi_{G_{Vision}}, \theta_{G_{Vision}}$, and $\psi_{G_{Vision}}$ are vision-based position and roll, pitch, and yaw angles of the quadrotor w.r.t. the reference frame, respectively.

4. Autolanding control algorithm

Our algorithm is divided into two main parts, position control with the IMU feedback and with the vision feedback, each of which is discussed in this section.

4.1. Navigation control algorithm using IMU

The drone is far from the landing area at the beginning, and the marker is out of the camera's field of view. The approximate location of the marker is given to the drone, and the drone moves toward the marker using the IMU sensors' feedback. One of the famous controllers that can be used for this purpose is the PID (proportional–integral–derivative) controller. For calculating the gains of the controller, first, the proportional gain is improved up to the point that the system starts to oscillate. Then, the derivative gain is added until the system reaches the border of damping. Finally, some integral gain is added to compensate the steady state error.

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 $X = (x_{G_{IMU}}, y_{G_{IMU}}, z_{G_{IMU}})$ as the current position of the drone, the PID controller is applied on 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. Considering that the error may not be differentiable, we apply the differential gain directly to the feedback value instead of the error [16]. Equations of this phase are as follows,

$$V_x = K_{px_{IMU}} e_x - K_{dx_{IMU}} \frac{dx_{G_{IMU}}}{dt} + K_{ix_{IMU}} \int_0^t e_x dt, \tag{18}$$

$$V_y = K_{py_{IMU}} e_y - K_{dy_{IMU}} \frac{dy_{G_{IMU}}}{dt} + K_{iy_{IMU}} \int_0^t e_y dt, \tag{19}$$

$$V_z = K_{pz_{IMU}} e_z - K_{dz_{IMU}} \frac{dz_{G_{IMU}}}{dt} + K_{iz_{IMU}} \int_0^t e_z dt, \tag{20}$$

where V_x, V_y, and V_z are linear velocities which are applied to the drone in directions X_G, Y_G, and Z_G, respectively, K_p, K_i, and K_d are the proportional, integral, and differential coefficients of the PID controller, respectively, which are obtained experimentally as follows.

$$\begin{pmatrix} K_{px_{IMU}} \\ K_{ix_{IMU}} \\ K_{dx_{IMU}} \end{pmatrix} = \begin{pmatrix} K_{py_{IMU}} \\ K_{iy_{IMU}} \\ K_{dy_{IMU}} \end{pmatrix} = \begin{pmatrix} K_{pz_{IMU}} \\ K_{iz_{IMU}} \\ K_{dz_{IMU}} \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.001 \\ 0.01 \end{pmatrix}. \tag{21}$$

4.2. Vision-based position control algorithm

When the landing marker becomes visible by the drone's camera, the vision-based position estimation turns active. Unfortunately, the drone does not move linearly proportional to the desired velocity, and slips easily. Furthermore, the position estimation and communication delays are other problems in this regard. It means that the commands, which should be executed in a particular position and time, would be executed later due to the drone's inertia, delay, and internal controller. Therefore, it will not stop at the desired position, and the drone will oscillate around the marker. In order to overcome this problem, close to the marker we divide the moving time into smaller time slots and we place halting slots between every moving slot to control the drone accurately. We call this method "movement slicing method".

The drone's speed is usually set proportional to the drone's position error as in the following,

$$V_x = K_{px_{Vision}}(x_{d_G} - x_{G_{Vision}}),\ V_y = K_{py_{Vision}}(y_{d_G} - y_{G_{Vision}}),\ V_z = K_{pz_{Vision}}(z_{d_G} - z_G), \tag{22}$$

where $K_{px_{Vision}}$, $K_{py_{Vision}}$, and $K_{pz_{Vision}}$ are the proportional coefficients of the controller in directions X_G, Y_G, and Z_G, respectively, $x_{G_{Vision}}$ and $y_{G_{Vision}}$ are the vision-based estimated positions w.r.t. the ground frame in directions X_G and Y_G, respectively, and z_G is the weighted sum of the data received from the vision, barometer sensor, and ultrasonic sensors as follows:

$$z_G = w_1 z_{G_S} + w_2 z_{G_{Vision}}, \tag{23}$$

where the value z_{G_S} is received from the ultrasonic and barometer sensors of the AR.Drone and $z_{G_{Vision}}$ is the vision-based z_G estimation. The best values of coefficients w_1 and w_2 are 0.7 and 0.3, respectively, obtained through the experiments. However, we also set the moving time proportional to the drone's position error as in the following,

$$T_x = K_{ptx_{Vision}}(x_{d_G} - x_{G_{Vision}}),\ T_y = K_{pty_{Vision}}(y_{d_G} - y_{G_{Vision}}), \tag{24}$$

where $K_{ptx_{Vision}}$ and $K_{pty_{Vision}}$ are the proportional coefficients and T_x and T_y are the calculated moving periods in directions of X_G and Y_G, respectively. Figure 6 shows the movement slicing method in direction X_G. There are two periods of actions: moving slot and halting slot. For example, in time slot 1, the drone is moving with the velocity of V_x for the calculated period of T_x. Next, it waits in time slot 2 until the drone becomes stable and the unwanted slips disappear completely, and then, it moves during the time slot 3 again. This is repeated until the drone reaches an acceptable area around the center of the marker. This algorithm eliminates the effects of any kind of time delays in the robot control, thereby ensuring its stability. In addition, the position estimations are taken into account during the halting period rather than the moving period, which also increases the accuracy of the estimations. Finally, according to the experimental results, the drones using the proposed algorithm perform the landing mission more smoothly with almost no oscillations around the target.

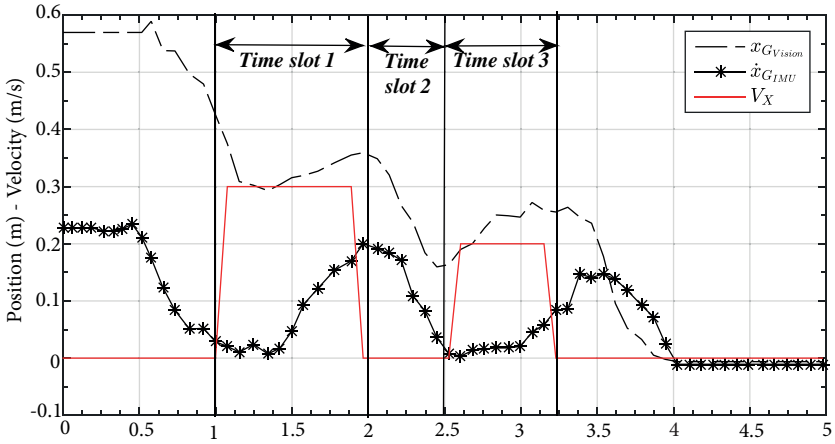

Figure 6. Timing graph of the moving and halting time slots in the movement slicing method in direction X_G.

4.3. Drone's alignment with the marker

The goal is landing the quadrotor aligned with the marker. First, the drone should be positioned above the center of the marker so that the marker does not exit from the camera's field of view when it is turning. Therefore, we put a 3D reference point on the center of the marker to check whether this point's image is in the valid region of the image or not, as in Figure 7. Next, we apply a turn command in order to align the quadrotor with the ground frame. Hence, another PID controller is designed for the angle (ψ) error as follows:

$$e_\psi = \psi_{d_G} - \psi_{G_{Vision}}, \tag{25}$$

where ψ_{d_G} is the drone's desired angle w.r.t. the ground frame and e_ψ is the angle's error. To align the drone with the marker, the drone's longitudinal axis is turned towards the marker's X_G axis with the rotation speed of ω_ψ. This speed depends on the angle error between the drone and the marker, an integral term and a derivative term which form the PID controller, as follows:

$$\omega_\psi = K_{p\psi} e_\psi - K_{d\psi} \frac{d\psi_{G_{Vision}}}{dt} + K_{i\psi} \int_0^t e_\psi dt, \tag{26}$$

where $K_{p\psi}$, $K_{i\psi}$, and $K_{d\psi}$ are coefficients of the controller found experimentally as follows:

$$\left(\begin{array}{cccc} K_{p\psi}, & K_{i\psi}, & K_{d\psi} \end{array} \right) = \left(\begin{array}{cccc} 0.5, & 0.001, & 0.001 \end{array} \right) \tag{27}$$

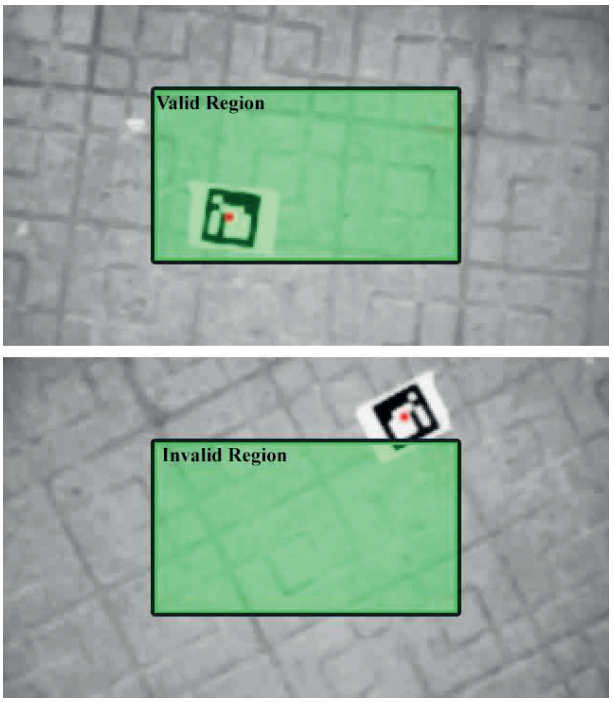

Figure 7. Checking the reference point's positioning in the valid central region of the image.

4.4. Landing algorithm

Prior to the landing, three important conditions have to be met. The drone should (I) be positioned exactly above the center of the landing marker, (II) have zero speed (it should be still), and (III) have the zero yaw angle w.r.t. the ground frame. After that, the quadrotor can gradually land on the marker. It is worth to mention that in this step, vision, ultrasonic, and barometer sensors are utilized altogether to reach the highest accuracy. In addition, another PID controller is used for the quadrotor's height control as follows:

$$V_z = K_{pz} e_z - K_{dz} \frac{dz_G}{dt} + K_{iz} \int_0^t e_z dt \tag{28}$$

where z_G is the drone's height (calculated by Eq. 23), e_z is the height error, K_{pz}, K_{iz}, and K_{dz} are the proportional, integral, and derivative coefficients, respectively obtained experimentally as follows:

$$\left(\begin{array}{cccc} K_{pz}, & K_{iz}, & K_{dz} \end{array} \right) = \left(\begin{array}{cccc} 0.5, & 0.01, & 0.001 \end{array} \right) \tag{29}$$

Figure 8 shows the autolanding procedure flowchart and Figure 9 shows the drone's control flowchart. According to the experimental studies, controlling the quadrotor gets very difficult when it is closer than 20 cm to the ground. This behavior, which is caused by wind effect of the propellers, is called "near ground effect". Researchers in [17] also reported unstable behavior of the AR. Drone due to the sensors' recalibration and wrong measurement in lower heights. Therefore, we found it safe enough to turn all of the four motors off as soon as the drone is closer than 15 cm to the ground.

5. Experimental results

The proposed method is implemented and evaluated on an AR.Drone 2.0 quadrotor using its downward camera, with the quality of QVGA and the field of view of 64 degrees. Figure 10 shows AR.Drone 2.0 while it is landing on a marker.

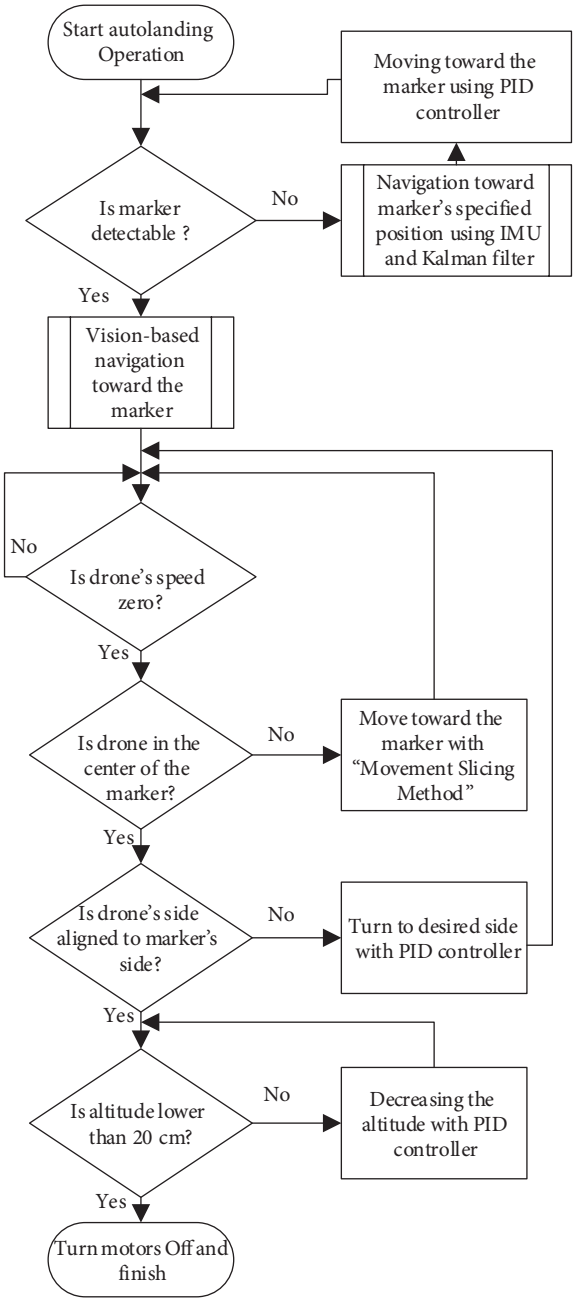

Figure 8. Detailed flowchart of the UAV's automaticlanding.

The main program which controls the drone through the Wi-Fi link runs on a laptop computer with the following specifications: DELL Studio 1558; CPU: Core i-7 Q720 1.6GHz; Ram: 6GB; OS: Ubuntu Linux 14.04 LTS 64 bit. Figure 11 illustrates the main block diagram of the drone and the ground station. Every part of the

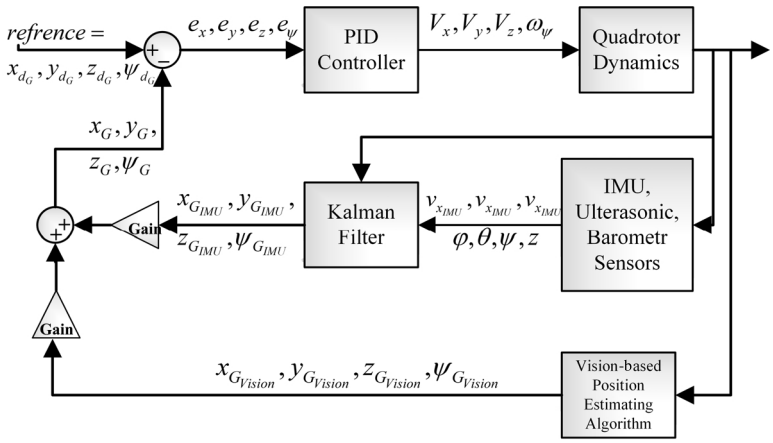

Figure 9. Control loop of automatic landing of UAV with two modes of using the vision or IMU sensors.

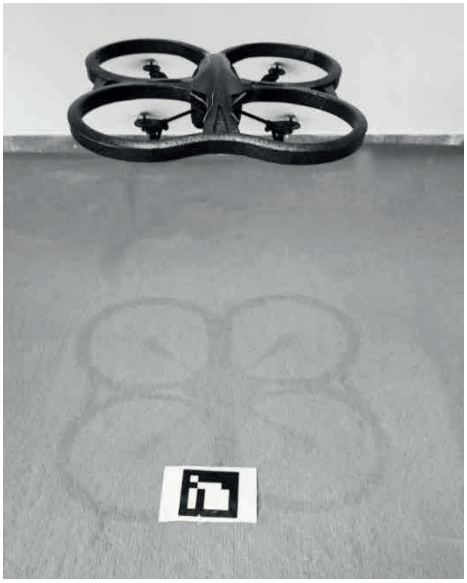

Figure 10. AR.Drone 2.0 quadrotor during autolanding on the marker.

program in the ground station is implemented in a different thread in order to achieve a better performance. Size of the markers is assumed to be 13×13 cm in all of the upcoming experiments.

5.1. Moving toward the marker

In Figure 12 the drone flies from position $(-1, 0, 1)$ to position $(0, 0, 1)$, where the marker is placed. A PID controller with IMU and vision feedback is used for this purpose. However, as Figure 12 shows the drone oscillates around the marker due to the drone's delay in executing the commands. For example, the PID controller has figured out that the position is zero at 6.5^{th} s, and it resets the drone's movement command to zero. However, the drone continues moving until the 8^{th} s and then, it executes this command in the 9^{th} s which is too late and the drone has some distance from the desired point. This happens again and again so the drone oscillates. However, as it is shown in Figure 13 when the movement slicing method is employed to solve the problem, the drone stops as soon as the marker is seen and performs the rest of the mission with the machine vision part.

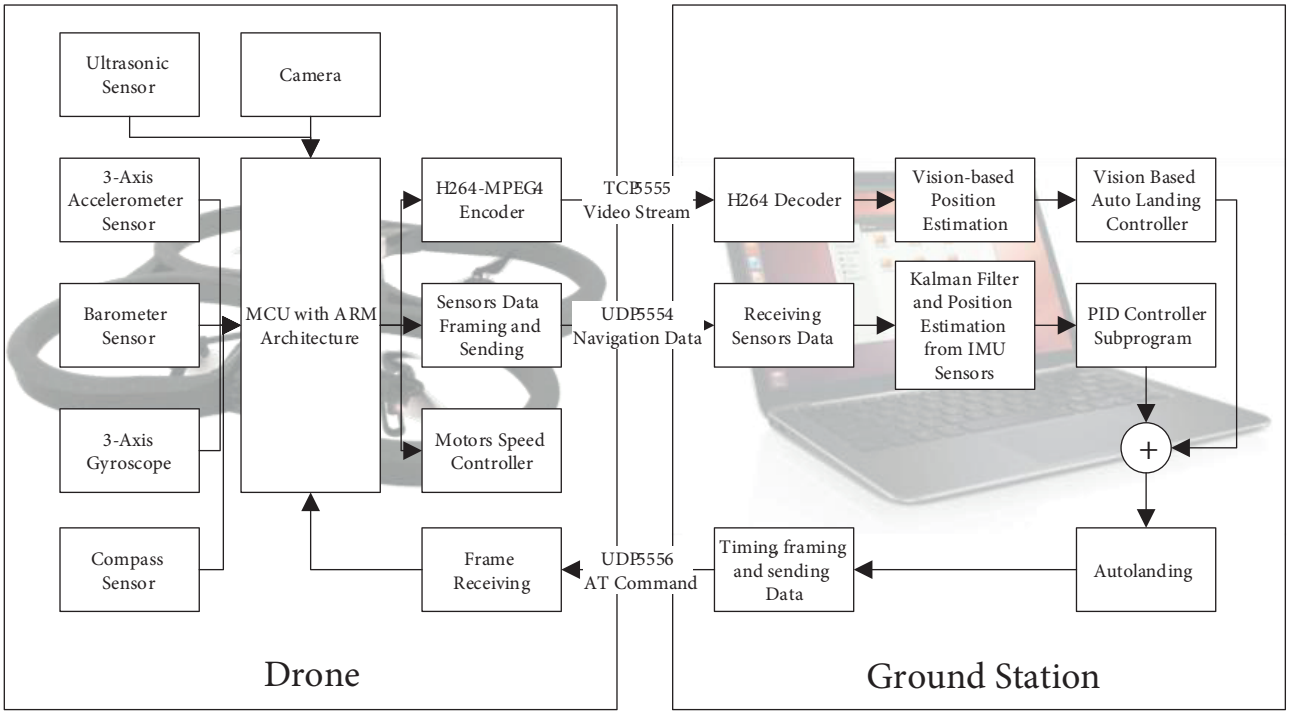

Figure 11. Main block diagram of the drone and the ground station.

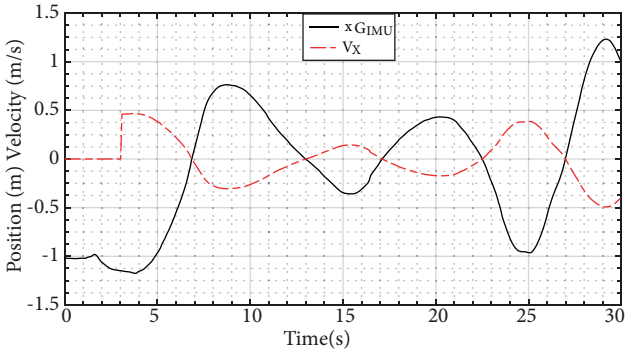

Figure 12. The velocity and the position of the drone in direction X_G during an unsuccessful flight without using "movement slicing method".

5.2. Precise positioning above the marker and landing

In this section, we address the complete landing procedure with the moving and halting time slots as shown in Figure 14. In this figure, the drone starts to fly in the 6th s and rises up to the height of 1.3 m, and in the 10th s, the autolanding procedure starts. First, the drone could not see the marker with the camera, so it moves toward the defined position using the IMU sensors to reach around the marker. After the 17th s, it sees the marker and the feedback changes to the vision state. The drone waits until the speed is declined to almost zero; after that it moves in the X_G and Y_G directions to align the marker with the center of the drone. Next, as it can be seen in yaw angel the drone turns to be aligned with the marker using the PID controller. It moves again along the X_G and Y_G axes for 7 s in order to set the marker in the center of the drone precisely. After that, the drone

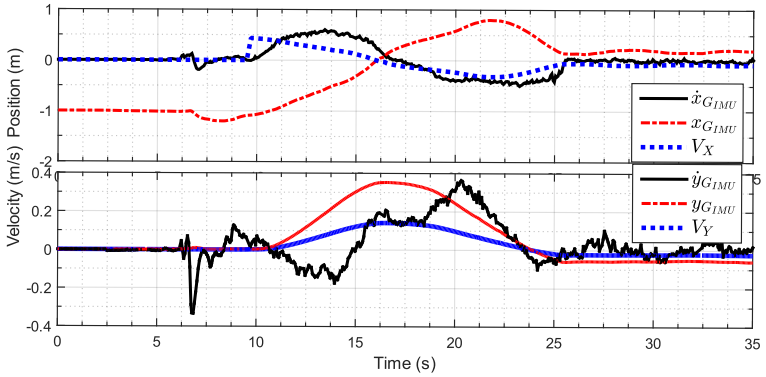

Figure 13. The velocities and the positions of the drone in directions X_G and Y_G during a successful flight using "movement slicing method".

slowly decreases the height to land on the ground precisely while keeping the marker in the center. The whole landing time, from where the autolanding procedure was started (1 m horizontal distance and 1.3 m vertical distance from the marker), took 15.7 s. Furthermore, the distance between the camera's center and the marker's center after landing is 2.3 cm. This experiment is carried out 20 times and the results are reported in Table 1. To the best of our knowledge, there is no similar report about the experimental parameters of automatic landing in the literature to compare our results with. However, some of the experimental results from other closer researches are reported as a comparison in Table 2. It is worth noting that in another recent study [39] regarding the vision-based landing of UAVs, an H-shaped marker was used as the landing pad, and similar to our work, the position estimation and control were implemented using PID controller and Kalman filter. As it is also mentioned in [39], one of the problems in that research is that the drone sometimes experiences sudden movements and instabilities while in our research, these kind of problems have been solved using the "movement slicing method". Furthermore, our proposed algorithm with the frame rate of 24 fps and the root mean square error (RMSE) of 2.5cm in position estimation has a more acceptable performance compared to 20fps frame rate and 1.37cm RMSE reported in [39].

Table 1. Experimental results

Number	Test topic	Result
1	The number of autolanding tests	20 times
2	The number of successful landings	17 times
3	The percentage of successful landings	85%
4	The average time of successful landings	14.3 s
5	The average distance from the center of the drone to center of the marker after successful landing	2.9 cm
6	The average frequency of processing of images	24 Hz

*Landings were done with a distance of 1 m and an angle of 90° from the marker.

6. Conclusions

We proposed a new method for automatic landing of a low-cost commercial quadrotor called AR.Drone 2.0. One of the challenging problems in vision-based position control of UAVs is the oscillation around the target

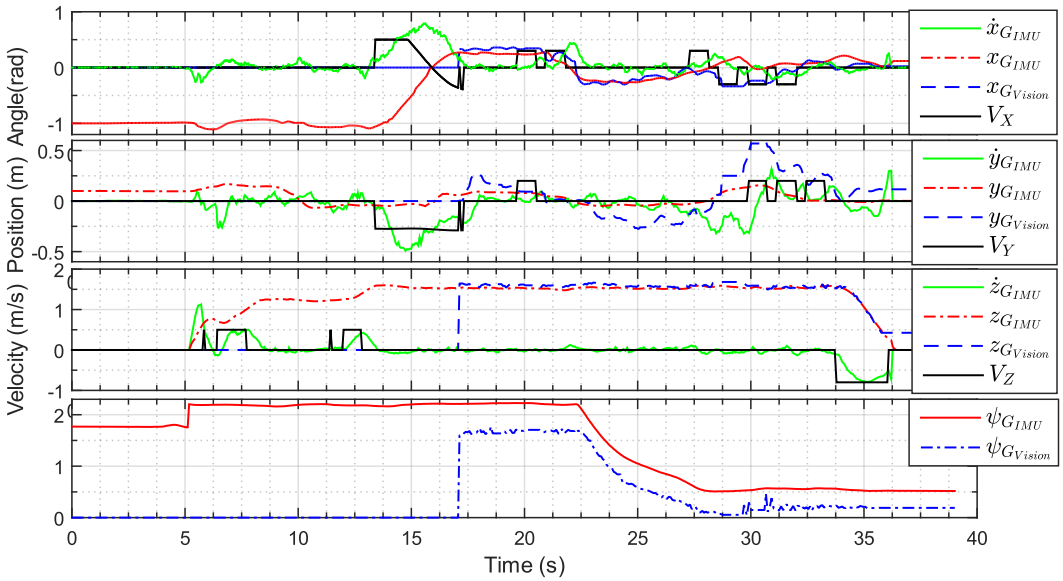

Figure 14. Complete autolanding experimental result showing the position and yaw angel (extracted from both IMU and Vision), real velocity and applied velocity in all three directions of X_G, Y_G, and Z_G.

Table 2. Automatic landing tests: a comparison

Ref	Test topic	Marker size	Position estimation max error	Height estimation max error	Landing time
This work	IMU and vision-based automatic navigation and landing on an ARUCO marker	13 × 13 cm	1.8 cm in 30 cm (7%)	11.2 cm in 160 cm (6%)	14.3 s*
[16]	Vision-based pose estimation algorithm using one marker (simulation results)	6 × 6 cm	1.4cm in 25cm (5.6%)	5.5 cm (3.92%)	60 s**
[26]	Vision-based state estimation	Not reported	5 cm (5%)	Not reported	Not reported
[11]	Stationary target landing in IPS*** (simulation results)	30 × 30 AprilTag	Not reported	Not reported	17 s****
[24]	Vision-based pose estimation	45 cm	(3.9%)	2 cm in 200 cm (1%)	Not reported

*Landings were done with a distance of 1 m and an angle of 90 ° from the marker

**Landings were done with a height of 1.6 m

***Indoor positioning system

****Landings were done from (x, y) = (0.2 m, 0.3 m) and height of 2.5 m

upon landing, which is due to UAV's inertia and position estimation and communication delays. Our movement slicing method solves this problem by dividing the moving time around the target into smaller time slots called moving and halting. The moving time is proportional to the UAV's position error and in halting time slot UAV waits until it becomes stable and unwanted slips disappear completely. Experimental results show that

this method not only stabilizes the whole system, but also achieves accurate landings (with position error of only 3cm) in a short time. The results of this study are useful for developing fully autonomous, accurate, and stable landing systems for package delivery drones. In addition, the proposed algorithm can be implemented on a drone equipped with a powerful on-board computer in order to carry out all the computations on board, thereby improving the accuracy and the efficiency of the system.

References

[1] Rao DV, Go TH. Automatic landing system design using sliding mode control. Aerospace Science and Technology 2014; 32 (1): 180-7. doi: 10.1016/j.ast.2013.10.001

[2] Daly JM, Ma Y, Waslander SL. Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays. Autonomous Robots 2015; 38 (2): 179-191. doi: 10.1007/s10514-014-9400-5

[3] Junaid A Bin, Lee Y, Kim Y. Design and implementation of autonomous wireless charging station for rotary-wing UAVs. Aerospace Science and Technology 2016; 54: 253-266. doi: 10.1016/j.ast.2016.04.023

[4] Mulgaonkar Y. Automated recharging for persistence missions with multiple micro aerial vehicles. PhD, University of Pennsylvania, Philadelphia, PA, USA, 2012.

[5] Cocchioni F, Pierfelice V, Benini A, Mancini A, Frontoni E, Zingaretti P et al. Unmanned ground and aerial vehicles in extended range indoor and outdoor missions. In: IEEE 2014 International Conference on Unmanned Aircraft Systems; Orlando, FL, USA; 2014. pp. 374-382.

[6] Achtelik M, Zhang T, Kuhnlenz K, Buss M. Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors. In: IEEE 2009 Mechatronics and automation; Changchun, China; 2009. pp. 2863-2869.

[7] Wenzel KE, Masselli A, Zell A. Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. Journal of Intelligent & Robotic Systems 2011; 61 (1-4): 221-238. doi: 10.1007/s10846-010-9473-0

[8] Yu K, Budhiraja AK, Buebel S, Tokekar P. Algorithms and experiments on routing of unmanned aerial vehicles with mobile recharging stations. Journal of Field Robotics 2018; 1-15. doi: 10.1002/rob.21856

[9] Saska M, Krajnik T, Pfeucil L. Cooperative μUAV-UGV autonomous indoor surveillance. In: IEEE 2012 Systems, Signals and Devices; Chemnitz, Germany; 2012. pp. 1-6.

[10] Borowczyk A, Nguyen DT, Phu-Van Nguyen A, Nguyen DQ, Saussié D, Le Ny J. Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. IFAC-PapersOnLine 2017; 50 (1): 10488-10494. doi: 10.1016/j.ifacol.2017.08.1980

[11] Ling K. Precision Landing of a Quadrotor UAV on a Moving Target Using Low-Cost Sensors. Msc, University of Waterloo, Waterloo, Ontario, Canada, 2014.

[12] Sani MF, Karimian G. Automatic navigation and landing of an indoor AR. drone quadrotor using ArUco marker and inertial sensors. In: IEEE 2017 Computer and Drone Applications; Kuching, Malaysia; 2017. pp. 102-107.

[13] Bi Y, Duan H. Implementation of autonomous visual tracking and landing for a low-cost quadrotor. Optik 2013; 124 (18): 3296-3300. doi: 10.1016/j.ijleo.2012.10.060

[14] Ginkel Rv, Meerman I, Mulder T, Peters J. Autonomous Landing of a Quadcopter on a Predefined Marker. Amsterdam, the Netherlands: University of Amsterdam, Project Report, 2013.

[15] Yu C, Cai J, Chen Q. Multi-resolution visual fiducial and assistant navigation system for unmanned aerial vehicle landing. Aerospace Science and Technology 2017; 67: 249-256. doi: 10.1016/j.ast.2017.03.008

[16] Carreira TG. Quadcopter automatic landing on a docking station. Msc, University of Lisbon, Portugal, 2013.

[17] Benavidez PJ, Lambert J, Jaimes A, Jamshidi M. Landing of a Quadcopter on a mobile base using fuzzy logic. In: WCSC 2013 Advance Trends in Soft Computing; San Antonio, TX, USA; 2014. pp. 429-437.

[18] Altan A, Aslan Ö, Hacıoğlu R. Real-time control based on NARX neural network of hexarotor UAV with load transporting system for path tracking. In: IEEE 2018 International Conference on Control Engineering & Information Technology; Istanbul, Turkey; 2018.

[19] Shah S. Real-time image processing on low cost embedded computers. California, CA, USA: University of California, Berkeley, Techincal report No. UCB/EECS-2014–117, 2014.

[20] Shin YH, Lee S, Seo J. Autonomous safe landing-area determination for rotorcraft UAVs using multiple IR-UWB radars. Aerospace Science and Technology 2017; 69: 617-624. doi: 10.1016/j.ast.2017.07.018

[21] Forster C, Faessler M, Fontana F, Werlberger M, Scaramuzza D. Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. In: IEEE 2015 International Conference on Robotics and Automation; Seattle, WA, USA; 2015. pp. 111-118.

[22] Mitra S, Land B. Autonomous quadcopter docking system. Ithaca, NY, USA: Cornell University, Report, 2013.

[23] Podhradsky M. Visual servoing for a quadcopter flight control. Msc, Luleå University of Technology, Sweden, 2012.

[24] Lange S, Sünderhauf N, Protzel P. Autonomous landing for a multirotor UAV using vision. In: 2008 International Conference on Simulation, Modeling, and Programming for Autonomous Robots; Venice, Italy; 2008 pp. 482-491.

[25] Krajník T, Vonásek V, Fišer D, Faigl J. AR-drone as a platform for robotic research and education. In: 2011 International Conference on Research and Education in Robotics; Berlin, Heidelberg, Germany; 2011. pp. 172-186.

[26] Sharp CS, Shakernia O, Sastry SS. A vision system for landing an unmanned aerial vehicle. In: IEEE 2001 Robotics and Automation; Seoul, South Korea; 2001. pp. 1720-1727.

[27] Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 2014; 47 (6): 2280-2292. doi: 10.1016/j.patcog.2014.01.005

[28] Chen Y, Liu HL. Overview of landmarks for autonomous, vision-based landing of unmanned helicopters. IEEE Aerospace and Electronic Systems Magazine 2016; 31 (5): 14-27. doi: 10.1109/maes.2016.150053

[29] Bowditch JI. The New American Practical Navigator. New York, NY, USA: E. & GW Blunt, 1857.

[30] Kalman RE. A new approach to linear filtering and prediction problems. Transactions of ASME, Series D, Journal of Basic Engineering 1960; 82 (1): 35-45. doi: 10.1115/1.3662552

[31] Maybeck PS. Stochastic Models, Estimation, and Control. New York, NY, USA: Academic Press, 1982.

[32] Bradski G, Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, CA, USA: O'Reilly Media Inc. 2008.

[33] Diebel J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Matrix 2006; 58 (15-16): 1-35.

[34] Suzuki S. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 1985; 30 (1): 32-46. doi: 10.1016/0734-189x(85)90016-7

[35] Baggio DL. Mastering OpenCV with Practical Computer Vision Projects. Birmingham, UK: Packt Publishing Ltd, 2012.

[36] Otsu N. A threshold selection method from gray-level histograms. IEEE Transactions on Systems Man and Cybernetics 1979; 9 (1): 62-66. doi: 10.1109/TSMC.1979.4310076

[37] Zhang Z. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence 2000; 22 (11): 1330-1334. doi: 10.1109/34.888718

[38] Lepetit V, Moreno-Noguer F, Fua P. Epnp: an accurate o(n) solution to the PnP problem. International Journal of Computer Vision. 2009; 81 (2): 155-166. doi: 10.1007/s11263-008-0152-6

[39] Patruno C, Nitti M, Petitti A, Stella E, D'Orazio T. A vision-based approach for unmanned aerial vehicle landing, Journal of Intelligent & Robotic Systems 2018; 1-20. doi: 10.1007/s10846-018-0933-2