2018 15th International Conference on
Control, Automation, Robotics and Vision (ICARCV)
Singapore, November 18-21, 2018

Quadrotors' Low-cost Vision-based Autonomous Landing Architecture on a Moving Platform

Jiaqi Jiang, Yuhua Qi, Muhammad Ibrahim, Jianan Wang, Chunyan Wang and Jiayuan Shan

Abstract— **In this paper, a low-cost vision-based autonomous landing architecture for an unmanned aerial vehicle (UAV) on a moving platform is presented. First, a novel landing pad was designed for a monocular camera to robustly detect the pad in both high and low altitudes. In order to solve mirror effect and occasional misidentification, a 3D points cluster algorithm for relative position estimation is presented. Second, the dynamics of the quadrotor is simplified for this lading task and a PD controller is designed based on the estimated relative position. Finally, the low-cost system architecture of the quadrotor and the experiment results are both presented to show the effectiveness of the proposed method.**

I. INTRODUCTION

In recent years, UAVs especially vertical take-off and landing (VTOL) UAVs have been used in many applications such as search and rescue, reconnaissance, surveillance and exploration. As an important part of flight task, autonomous landing is a fundamental yet challenging problem of quadrotor UAV.

Due to the importance of the autonomous landing task, it received great attention in the last decade. Representative works are as follows: [1] presented an onboard monocular vision system for autonomous landing on a typical landing pad which consists of the letter H surrounded by a circle. [2] presented a nonlinear controller for a UAV using the measurement of average optical flow on a spherical camera along with the IMU data to land on a moving platform. In [3], an image-based visual servo control for the landing problem of a VTOL vehicle was presented. The image of the centroid for a collection of landmarks was used as position measurement and translational optical flow was used as velocity measurement for the autonomous landing. The authors in [4] designed a landing pad for the situation that parts of the target are not visible and a vision based detection algorithm for estimation of the 3D position of the UAV relative to the landing pad.

To date, many vision-based algorithms are developed for autonomous landing of UAVs. Published works investigated the landing on static, moving as well as unknown sites. These algorithms usually require pre-known marker to be attached to the landing sites. In [5], the detection of the marker is based on the color detection algorithm with morphological filter. In [6], a pad design aiming for possible detection from high as well as small altitudes is presented. The pattern comprised one red square and one blue circle, which can increase the robustness and range of detection. [1] presented a landing pad which consists of the letter H inside a black circle. The 5 DOF pose was calculated from the elliptic projection of the circle. The remaining geometric ambiguity was resolved after incorporating the estimation of the gravity vector using the vehicles inertial measurement unit. Respectively, the author in [7] presented an algorithm which dose not need a pre-know marker, the landing site was defined by a template image and identified using the matching of detected ORB features. The relative pose was estimated by matching the landing site features to the PTAM map points. But it consequently loses its accuracy in the case of the absence of the map.

In the real applications, it is important to decrease the cost and enhance the practicability of the algorithm. In this paper, aiming at actual application and low-cost autonomous system, a whole solution for autonomous landing of a quadrotor UAV including visual algorithm, and control law are presented. The contributions of this paper are as follows:

- a novel landing pad is designed to increase the detection range and robustness.
- a 3D points cluster algorithm for pose estimation is presented to solve the problem of mirror effect and occasional misidentification.
- open source flight controller and other low-cost components are adopted in our experiments.

The rest of this paper is organized as follows. In Section II, the landing pad design and the detection algorithm are given. The estimation of the relative position between the UAV and landing pad is presented as well. The fundamental equations of the dynamic model and the PD controller for the issue is presented in Section III. Experiments setup and practical experiments results are given in Section IV. Finally, Section V concludes this paper.

II. DETECTION METHOD AND LANDING PAD DESIGN

In this section, the landing pad design and the onboard vision system for providing the relative pose between the UAV and the landing platform will be investigated.

A. Landing Pad Design & Detection

Fig. 1 presents the landing pad configuration which is designed to achieve the goals of keeping the desired target within the camera Field of View (FOV), and reducing measurement noise. The accuracy can be improved by using the landing pad instead of using an expensive high resolutions camera. The landing pad consists of different size ArUco

*This work was supported by NSFC Grant No. 61503025.

Jiaqi Jiang, J.N. Wang, C.Y. Wang and J.Y. Shan are all with Key Laboratory of Dynamics and Control of Flight Vehicle within Ministry of Education, School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China. e-mail: wangjianan@ieee.org.

978-1-5386-9582-1/18/$31.00 ©2018 IEEE

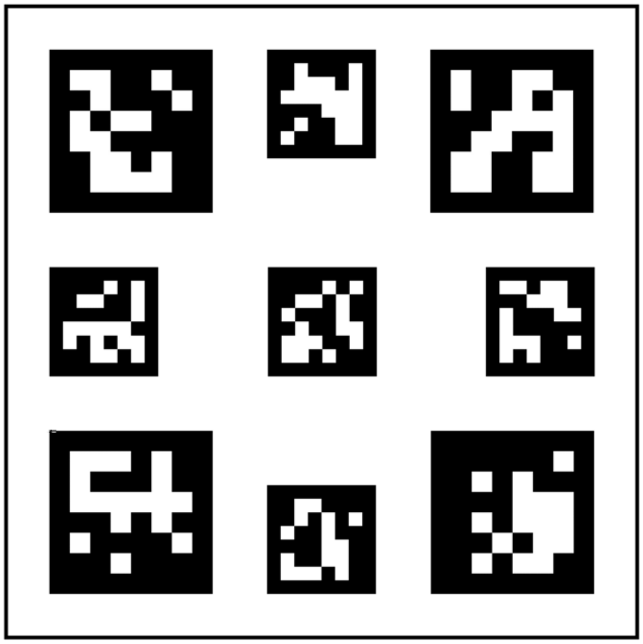

Fig. 1. Landing pad configuration.

markers, which makes it can be detected by camera from high as well as low altitudes and decreases the possibility of loss of tracking target. The bigger patterns can be detected from high altitudes. When the UAV is close to the platform during landing, these patterns gradually leave the camera FOV, while the smaller ones could still be detected as shown in Fig. 2. In terms of different application scenario, it is easy to modify detection range by changing the pattern size.

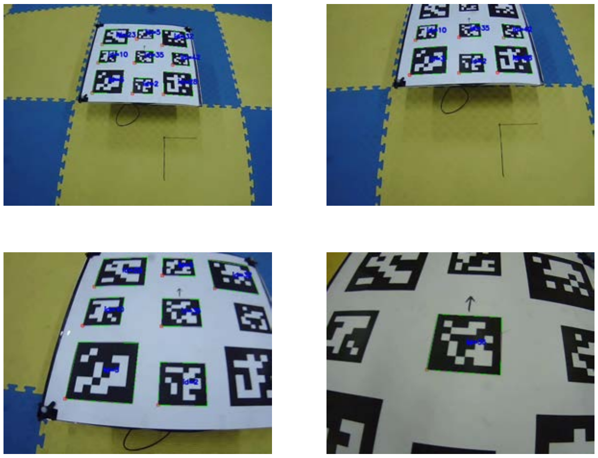

Fig. 2. Detection from different ranges.

The ArUco library [8] is chosen for the design of landing pad and the onboard vision system. Its detection pipeline is much more effective and faster than that of the similar AprilTag library [9], which is suited for our low-cost onboard computer. The marker detection process is composed of two main steps : the detection of marker candidates and codification analysis. In order to detect square shapes which are candidates, adaptive thresholding and contour extraction in combination with additional extra filtering are used in the detection process. The objective of the second step is to analyze inner codification. Firstly, perspective transformation is used to obtain the marker in canonical form. Then, aiming at the separation of white and black sections, Otsu thresholding method is used. Finally, bit by bit analysis is performed in order to determine whether the marker belongs to a specific tag dictionary. It is sufficient to estimate its full pose in absolute scale with the position of the four marker corners in the image and the real size of the marker.

B. Relative Pose Estimation

The problem of estimating the pose of a calibrated camera using n reference 3D points and their corresponding 2D projections is commonly referred as Perspective-n-Point (PnP) problem. Towards the ArUco markers, non-iterative methods like Efficient Perspective-n-Point (EPnP) are not considered, because they are not robust in the planar case. The homography method for planar targets [10] followed by non-linear optimization using the well-known Levenberg-Marquardt (LM) algorithm to minimize reprojection error is selected to estimate the transformation matrix T_{MC} from camera frame to marker frame.

$$\begin{bmatrix} x_M \\ y_M \\ z_M \\ 1 \end{bmatrix} = T_{MC} \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix}, \tag{1}$$

where C denotes camera frame, M denotes marker frame.

It is more convenient to estimate the optic central position relative to the static markers than to estimate the marker position relative to the UAV. But the planar surface of the landing pad and the limitation of the camera make mirror effect and occasional misidentification of some of the detected markers happen more frequently. So the method of just weighting the average of raw estimated pose could not provide more accurate result. To solve this issue, a cluster algorithm is implemented which does not need to define the amount of the clusters. Firstly, the error between two estimated positions $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ is defined in (2).

$$error_{ij} = ||\boldsymbol{p_i} - \boldsymbol{p_j}||_2 \tag{2}$$

Then, a threshold is defined for checking whether two positions are in the same cluster. After checking all estimated positions, the cluster with the largest data is the correct detection cluster. At last, the accurate relative position is obtained by weighted average processing of correct detection data. Particularly, if more than one clusters have the largest data, then we choose the one which is the nearest to the last estimated position as the correct detection cluster.

Fig. 3 and Fig. 4 show the relative position using raw detection results and 3D points cluster algorithm ,respectively. In Fig. 3, occasional misidentification and mirror effect make the estimated pose oscillate severely. As shown in Fig. 4, it is clearly seen that the erroneous pose estimations are

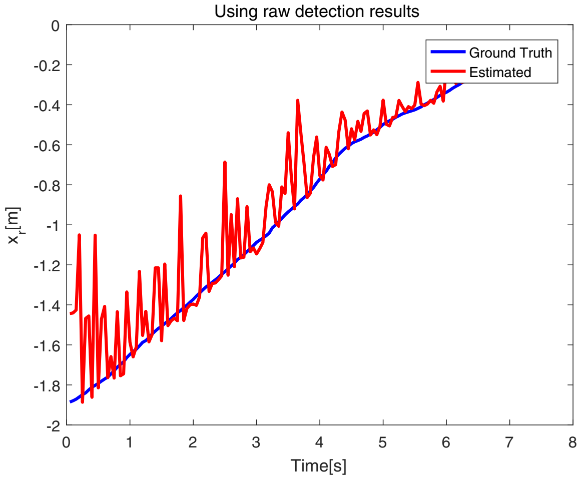

Fig. 3. Pose estimation result using raw detection results.

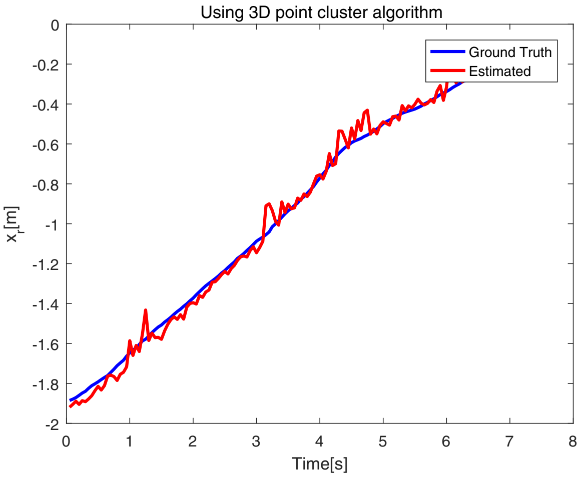

Fig. 4. Pose estimation result using 3D points cluster algorithm.

completely removed after introducing the 3D points cluster algorithm.

To verify robustness of the pose estimation system, let the uav move from 0.5m to 2.5m in marker's x-axis at a certain altitude of 2m. The accuracy and standard deviation of the estimation result at different distance are shown in Table I

TABLE I

ESTIMATION RESULT AT DIFFERENT DISTANCE

Distance in x-axis(m)	0.5	1	1.5	2	2.5
Mean Position Error(cm)	3.71	4.02	4.25	4.34	7.66
Standard Deviation(cm)	4.13	4.53	4.84	5.37	9.12

III. DYNAMIC MODEL AND CONTROL LAW

In this section, the fundamental equations of dynamic model and the PD controller for the landing task is presented.

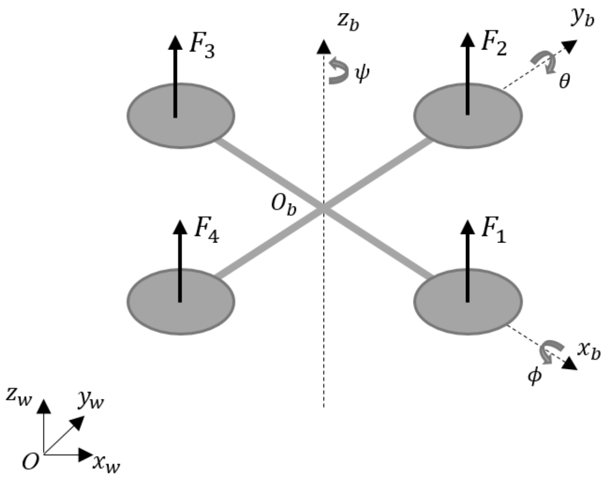

Fig. 5. The quadrotor UAV with corresponding frames.

A. Dynamic Model

Consider a quadrotor vehicle represented by a rigid body of mass m and of moment of inertia J, as shown in Fig. 5. Let $\sum_w = \{x_w, y_w, z_w\}$ be the world inertial frame, $\sum_b = \{x_b, y_b, z_b\}$ denote a set of coordinates attached to the UAV at the mass center. To describe the position and attitude of the UAV, $\boldsymbol{\xi}_b = [x, y, z]^T$ is defined to denote the position of the vehicle's mass center relative to the inertial frame $\sum_w$, $\boldsymbol{v_b} = [\dot{x}, \dot{y}, \dot{z}]^T$ is the linear velocity of the UAV in $\sum_w$,and $\boldsymbol{\phi} = [\varphi, \theta, \psi]^T$ is defined as the roll/pitch/yaw angles of the quadrotor, respectively. Hence the rotation matrix R_b from $\sum_b$ to $\sum_w$ is given as

$$R_b = \begin{bmatrix} c_\theta c_\psi - s_\varphi s_\theta s_\psi & -c_\varphi s_\psi & s_\theta c_\psi + s_\varphi c_\theta s_\psi \\ c_\theta s_\psi + s_\varphi s_\theta c_\psi & c_\varphi c_\psi & s_\theta s_\psi - s_\varphi c_\theta c_\psi \\ -c_\varphi s_\theta & s_\varphi & c_\varphi c_\theta \end{bmatrix}, \quad (3)$$

where c_γ and s_γ denote, respectively, $\cos\gamma$ and $\sin\gamma$.

In the frame system as shown in Fig. 5, the equations of motion for the quadrotor can be written as follows

$$\dot{\boldsymbol{\xi}}_b = \boldsymbol{v}_b, \quad (4)$$

$$m\dot{\boldsymbol{v}}_b = R_b \boldsymbol{F}_b, \quad (5)$$

$$J\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times J_\omega + \boldsymbol{\tau}, \quad (6)$$

where $\boldsymbol{F}_b = [0, 0, F_b]^T$is the force applied to the vehicle expressed in $\sum_b$, g is the gravitational constant and $\boldsymbol{\tau}$ is the torque applied to the vehicle.

In most autopilots, the hierarchical control scheme as shown in Fig. 6 is used to control the quadrotor. In this case, only the translation dynamics is concerned. For the orientation dynamics of (6), a high-gain controller is used to ensure that attitude ϕ converges to the desired attitude ϕ_d. The translational dynamics of the multirotor are shown in (4), (5) and can also be represented as

$$\ddot{x} = \frac{F_b}{m}\left(\sin\theta\cos\psi + \sin\varphi\cos\theta\sin\psi\right), \quad (7)$$

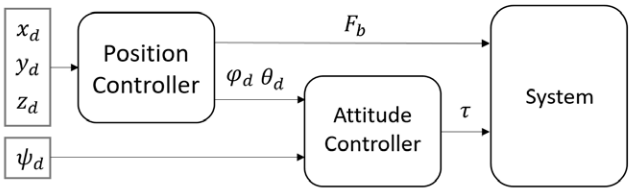

Fig. 6. Hierarchical control scheme of a quadrotor.

$$\ddot{y} = \frac{F_b}{m}\left(\sin\theta\sin\psi - \sin\varphi\cos\theta\cos\psi\right), \tag{8}$$

$$\ddot{z} = \frac{F_b}{m}\cos\phi\cos\theta - g. \tag{9}$$

B. Control Law

Since the yaw channel is independent of other channels, it is reasonable to let the yaw angle be zero and also x_b always point to the landing pad as shown in Fig. 7. In the landing task, roll and pitch angles are both small otherwise the landing pad will be out of camera FOV. The translation dynamics can be simplified as

$$\ddot{x} = \theta\frac{F_b}{m}, \tag{10}$$

$$\ddot{y} = -\varphi\frac{F_b}{m}, \tag{11}$$

$$\ddot{z} = \frac{F_b}{m} - g. \tag{12}$$

Thus, we only consider the translation dynamics (10) - (12) with an input $[\varphi_d, \theta_d, F_b]$.

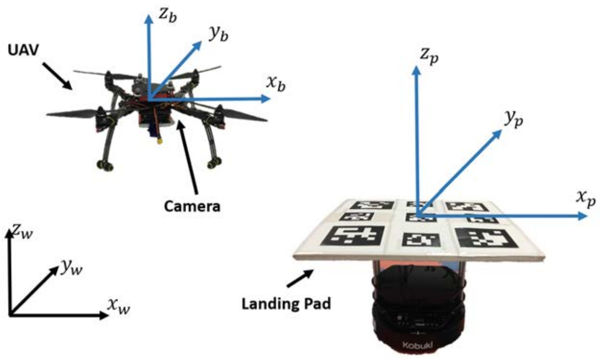

Fig. 7. Quadrotor UAV and landing pad.

As shown in Fig. 7, $[x_p, y_p, z_p]^T$ is the position of the landing pad in the frame $\sum_w$. Let $[x_r, y_r, z_r]^T = [x, y, z]^T - [x_p, y_p, z_p]^T$ represents the relative position between the quadrotor and the landing pad which can be directly obtained from onboard vision system. To simplify the task, one assumption is that the landing pad moves with a constant velocity in x_p direction. The translation dynamics are as follows

$$\ddot{x}_r = \theta\frac{F_b}{m}, \tag{13}$$

$$\ddot{y}_r = -\varphi\frac{F_b}{m}, \tag{14}$$

$$\ddot{z}_r = \frac{F_b}{m} - g. \tag{15}$$

The position control law for $[\varphi_d, \theta_d, F_b]$ can be designed using PD controller

$$\theta_d = -k_1 x_r - k_2\dot{x}_r, \tag{16}$$

$$\varphi_d = k_3 y_r + k_4\dot{x}_r, \tag{17}$$

$$F_b = mg - k_5 z_r. \tag{18}$$

where k_1, k_2, k_3, k_4 and k_5 are positive constants.

IV. Experiments Results

In this section, the proposed approach for autonomous landing is validated with two scenarios. This section presents the experimental setup and results of experiments including landing on a static landing pad and on a moving vehicle. Then, analysis on the experimental results will be presented.

A. Experimental setup

Fig. 8. Overlook of the quadrotor.

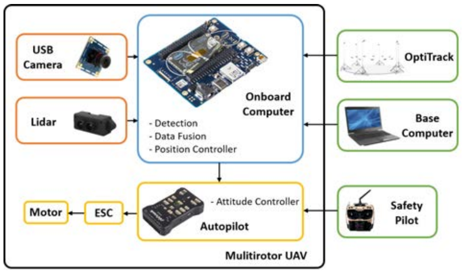

Fig. 9. Experimental setup.

The quadrotor used in this study is shown in Fig. 8. The overall system architecture, used to perform the experiments for validation of autonomous landing, is shown in Fig. 9. The quadrotor is developed based on the off-the-shelf frame. It weighs about 1.1kg with battery included and the axis length is 40cm. An open source autopilot hardware (Pixhawk [11]) and autopilot software (PX4 [12]) were adopted. The onboard computer (Intel Joule 570x [13]) with a single-core Intel Atom T5700 processor up to 1.7GHz runs three major software components responsible for

1) interfacing with camera hardware, landing pad detection and relative pose estimation;

2) computation of control inputs from the relative position. The control inputs are computed by the onboard computer and send to the flight controller;
3) interfacing with flight controller, sending control inputs to the inner-loop controller and receiving IMU data[14].

To detect the landing pad, a usb monocular camera is installed in front of the quadrotor with a setting angle. The camera has about 90° FOV and the resolution of the image is 640*480. The position of the quadrotor is measured by a motion capture system (OptiTrack [15]) and the translational velocity is estimated from this position measurements. Those values are computed at the ground computer and transmitted to the onboard computer via a Wi-Fi communication by using vrpn-clinet-ros package [16].

B. *Experimental results: Case 1*

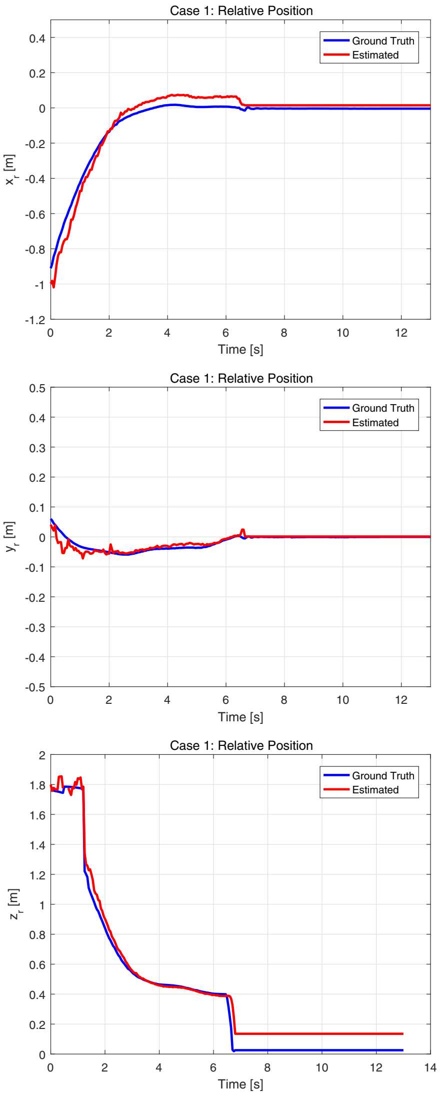

Fig. 10. Case 1: Relative position.

This section presents the autonomous landing scenario on a static landing pad. For the control gains in (16) - (18), we choose $k_1 = 2, k_2 = 2, k_3 = 0.2, k_4 = 0.2, k_5 = 1$. Fig. 10 depicts the relative position bet-ween the quadrotor and landing pad. The bule line represents the estimated value and black line represents the ground truth from OptiTrack. It can be seen that the relative position x_r, y_r, z_r converges to zero. The error after landing in z_r direction is due to the strategy that the power will be cut off when z_r is smaller than 20cm. Fig. 11 shows snapshots taken during the autonomous landing experiment.

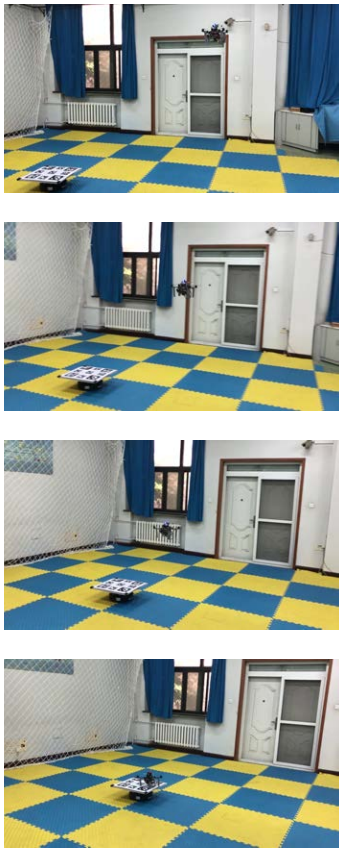

Fig. 11. Pictures taken during autonomous landing on a static landing pad.

C. *Experimental results : Case 2*

In Case 2, this section presents the autonomous landing scenario on a moving landing pad which is fixed on the Turtle Bot [17]. We choose the same control gains in Case 1. The experimental results are shown in Fig. 12. Fig. 13 shows snapshots taken during the autonomous landing experiment.

A video clip of these two experiments is posted on the following URL: `https://v.youku.com/v_show/id_XMzY5NjQ3MDY0MA==.html?spm=a2h0k.11417342.soresults.dtitle`

V. Conclusion

In this paper, a low-cost vision-based autonomous landing solution is proposed for quadrotors. For the autonomous landing task, a novel landing pad was designed for robust detection, ensuring the detectability from both high and low altitudes. A 3D points cluster algorithm for pose estimation

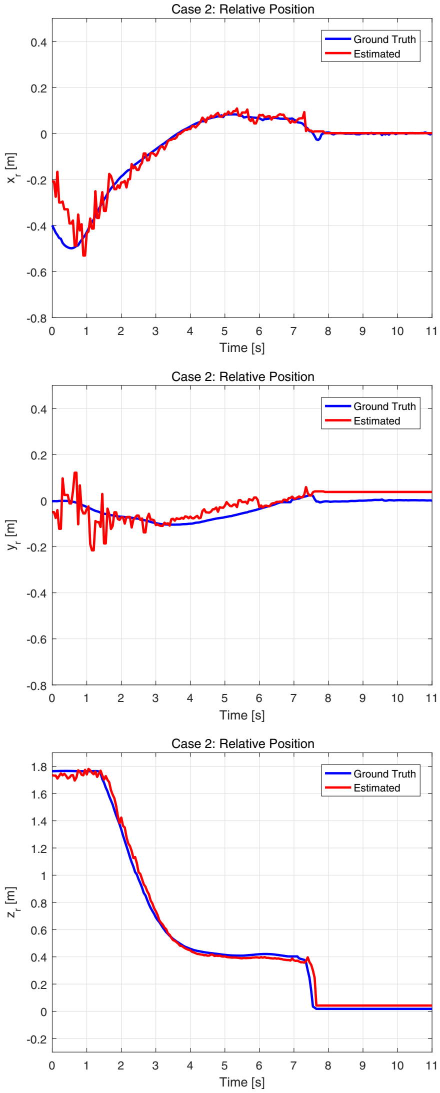

Fig. 12. Case 2: Relative position.

was presented to solve the problem of mirror effect and occasional misidentification. A simplified dynamic model for quadrotors in landing phase was proposed and thus a PD controller was designed accordingly to ensure the landing on an either static or moving pad. In future work, we plan to use the vision system and design robust control law to achieve autonomous landing on a variable-speed platform.

REFERENCES

[1] Yang, Shaowu, S. A. Scherer, and A. Zell. ”An Onboard Monocular Vision System for Autonomous Takeoff, Hovering and Landing of a Micro Aerial Vehicle.” *Journal of Intelligent & Robotic Systems* 69.1-4(2013):499-515.

[2] Hamel, Tarek, and R. Mahony. ”Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow.” *IEEE Transactions on Robotics* 28.1(2012):77-89.

[3] Serra, Pedro, et al. ”Landing of a Quadrotor on a Moving Target Using Dynamic Image-Based Visual Servo Control.” *IEEE Transactions on Robotics* PP.99(2016):1-12.

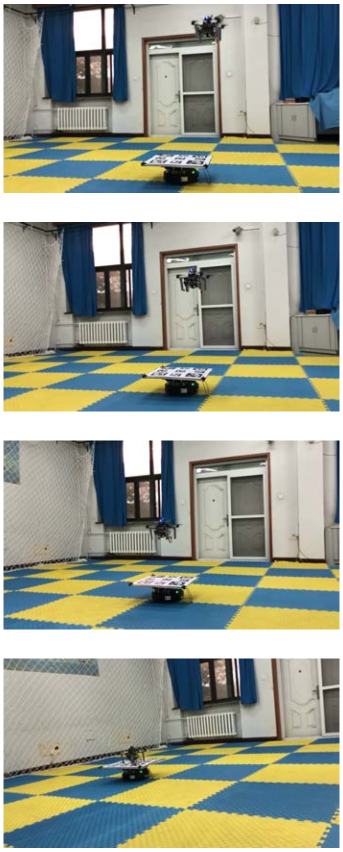

Fig. 13. Pictures taken during autonomous landing on a moving landing pad.

[4] Lange, S, N. Sunderhauf, and P. Protzel. ”A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments.” *International Conference on Advanced Robotics IEEE*, 2009:1-6.

[5] Lee, Hanseob, S. Jung, and D. H. Shim. ”Vision-based UAV landing on the moving vehicle.” *International Conference on Unmanned Aircraft Systems IEEE*, 2016:1-7.

[6] Chen, Xudong, et al. ”System integration of a vision-guided UAV for autonomous landing on moving platform.” *IEEE International Conference on Control and Automation IEEE*, 2016:761-766.

[7] Lee, Hanseob, S. Jung, and D. H. Shim. ”Vision-based UAV landing on the moving vehicle.” *International Conference on Unmanned Aircraft Systems IEEE*, 2016:1-7.

[8] Garrido-Jurado, S., et al. ”Automatic generation and detection of highly reliable fiducial markers under occlusion.” *Pattern Recognition* 47.6(2014):2280-2292.

[9] Olson, Edwin. ”AprilTag: A robust and flexible visual fiducial system.” *IEEE International Conference on Robotics and Automation IEEE*, 2011:3400-3407.

[10] Malik, Shahzad, and G. Roth. ”Robust 2D Tracking for Real-Time Augmented Reality.” *Proc.conf.vision Interface* (2002).

[11] Pixhawk. http://www.pixhawk.com/.

[12] PX4. https://github.com/px4/Firmware.

[13] Intel Joule 570X. https://ark.intel.com/products/96414/.

[14] Mavros. https://github.com/mavlink/mavros.

[15] OptiTrack. http://www.optitrack.com/.

[16] vrpn-client-ros. https://github.com/ros-drivers/vrpn-client-ros.

[17] Turtle Bot. http://www.turtlebot.com/