A Dual-loop Approach with Visual Servoing Fuzzy Control for Marker Navigation Quadcopter

Chen-Huan Pi, Voong Bin Sheng, Stone Cheng*

National Chiao Tung University, Department of Mechanical Engineering, Taiwan

*Corresponding Author: stonecheng@mail.nctu.edu.tw

Abstract

This paper is to present a dual-loop control systems for the AR.Drone 2.0 quadcopter with internal controller treated as a black-box system to perform visual-servoing hovering task based on the vision feedback from its bottom camera and a visual marker placed on the ground. This system is implemented by using Robot Operating System (ROS). This paper describes the development, testing and implementation of marker detection method using Open Source Computer Vision (OpenCV) for image processing and also Fuzzy Logic Control to a real quadcopter, AR.Drone for controlling its position based on the marker on ground.

Keywords: Quadcopter, Visual-Servoing, Fuzzy Logic Control.

1. Introduction

In recent years, there is an increased interest on the research of Micro Aerial Vehicles (MAVs). By making an autonomous Micro Aerial Vehicles (MAV) which can perform the aforementioned tasks, it will open a lot of possibilities such as following or picking up ground objects, charging wireless ground sensors, landing on docking or charging station without human intervention in order to increase its flight time, aerial monitoring and inspection. Quadcopter is a class of four-rotor aerial vehicles which has the capabilities to provide stable acrobatic flight, navigate into places that ground robots or human cannot reach and perform tasks which are difficult or dangerous for human being such as agricultural surveillance and explosive landmines detection. Many commercially-available, off-the-shelf (COTS) quadcopters are available on market. The price range of COTS quadcopters varies depending on

their size, weight, functionalities, onboard sensors and hardware. The Parrot AR.Drone 2.0 has been chosen as the quadcopter platform for this research due to its low price and rich onboard sensors (inertial measurement unit, ultrasonic sensor and cameras). The Software Development Kit (SDK) of AR.Drone which released publicly by the manufacturer has allowed developers to create programs that can communicate wirelessly (WiFi) with the quadcopter. Hence, this has opened up possibilities for a variety of research purposes, for instance high level autonomous and control algorithm can be developed for the quadcopter to perform specific tasks.

Some previous studies have been made on the vision-based control of quadcopters. One of the visual servoing method for quadcopter is by using color-based tracking algorithm⁽¹⁻³⁾. This color-based tracking method is simple and fast by filtering the color of object from the background having different color. When the pixel position of colored object in camera is obtained, its difference with the center pixel position of camera is used as input for the control algorithm to compute control signal for quadcopter. Besides that, a low cost sensor such as the Wii remote infrared (IR) camera is also used for tracking a pattern of IR lights under conditions with no direct sunlight as introduced in literature^(3,4). However, the methods mentioned above are easily being affected by the environment. Several research have employed motion capture system such as Vicon or OptiTrack which can obtain the pose of quadcopter with good accuracy. However, the high price and the short visual operation range have limited the vision range of the system. The method proposed in this paper has greater flexibility by the usage of onboard sensors and cameras only for its operation without the need of external devices. Besides that, multiple control algorithms are used for quadcopter system: PID control^(1,3,5,6), PD control^(2,3), Predictive

Functional Control⁽¹⁾ and Fuzzy Logic Control^(7,8). Every control algorithm has its advantageous in performance (stability, robustness and disturbance rejection) over the others.

In this paper, we present a new visual servoing method by using onboard camera and ArUco marker that combining with inertial measurement unit. A fuzzy logic controller is designed to navigate quadcopter in indoor environment and compare with the original onboard controller loaded on AR.Drone 2.0.

2. Quadcopter Platform

2.1 Parrot AR.Drone 2.0

The quadcopter platform used in this research is AR.Drone 2.0 which manufactured by the Parrot company.

Fig. 1 AR.Drone 2.0 with indoor hull (left) and outdoor hull (right). Source: AR.Drone Developer Guide SDK 2.0⁽⁹⁾.

It comes with onboard flight controller which can give users the stable and secure flying experience wirelessly through smartphone.

2.2 Hardware Specifications

Its weight and size are 420g and 51.7cm x 51.7cm with indoor hull and 380g and 45.1cm x 45.1cm without hull as shown in Fig. 1. Its hardware specifications are listed in Table 1.

Table 1 The Hardware Specifications of AR.Drone 2.0.

Hardware	Specification
Processor	1GHz 32 bit ARM Cortex A8 processor with 800MHz video DSP TMS320DMC64x
Operating System	Linux 2.6.32
RAM	1GB DDR2 RAM at 200MHz
Sensors	Inertial Measurement Unit (IMU): <ul style="list-style-type: none"> • 3 axis gyroscope 2000°/second precision • 3 axis accelerometer +/-50mg precision • 3 axis magnetometer 6° precision • Pressure sensor +/- 10 Pa precision Ultrasound sensors for ground altitude

	measurement up to 6 meters
Cameras	<ul style="list-style-type: none"> • Front HD Camera for video recording and streaming: 720p (1280x720), 30FPS, 92° wide-angle diagonal lens • Vertical camera for ground speed measurement: QVGA (320x240), 60FPS, 64° field of view
Wireless	Wi-Fi b/g/n wireless network
Battery	3 cell 1000 or 1500 mA/H LiPo rechargeable official battery (flight time: ≈10~15 minutes)
Motors	4 brushless inrunner motors. 14.5W 28,500 rpm

2.3 Black-box Model and Body of Reference

The firmware running onboard is not open-source, hence it is not possible to modify the software for tuning the internal controller or adding new functionalities such as visual-servoing. The internal controller can be treated as a black-box model where it receives inputs (roll angle, pitch angle, yaw rate and vertical velocity) and gives outputs as depicted in Fig. 2. Besides that, the body reference frame and motors arrangement are also illustrated in Fig. 2.

Fig. 2 AR.Drone black-box model and body reference frame $\{X_B, Y_B, Z_B\}$.

3. Visual-Servoing Control for AR.Drone

In this paper, the visual-servoing technique used is Position-based Visual-Servoing (PBVS) with eye-in-hand configuration since the camera is located at the bottom. The position relative to the tracking object (marker) is obtained by adopting the open source marker detection algorithm, ArUco⁽¹⁰⁾.

3.1 ArUco Marker

ArUco is a minimal C++ library for Augmented Reality applications developed by the University of Cordoba and it is based on OpenCV. The marker detection process is simple, fast and only requires a single camera.

3.2 Tuning the Marker Detection Process

From experiments, the low resolution image from bottom camera contains noises which can affect the detection results are shown in Fig. 4. Thus, a low-pass filter called Gaussian Blur filter is proposed to reduce the amplitude of image variations by replacing the intensity of each pixel by weighted average of intensity values in its local neighborhood. The weighted scheme follows 2D Gaussian function (x and y is horizontal and vertical distance in pixel respectively while σ is standard deviation):

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

In this paper, Gaussian Blur filter with 3x3 block size and $\sigma = 1$ is applied on the image from bottom camera and the result is shown in Fig. 3 and 5. By comparing the results of Fig.4 and Fig. 5, it is can be concluded that the application of Gaussian Blur filter has greatly improved the marker detection result. It eliminates the high-frequency contents (noises) of the image captured by the low resolution bottom camera of AR.Drone and thus allows a more stable marker detection result.

Fig. 3 The result of applying Gaussian Blur filter on image from bottom camera: raw image (left) and binary image (right).

3.3 Vision-Based Quadcopter Pose Estimation

The position of marker relative to camera can be obtained from tuned ArUco marker detection algorithm as introduced previously through camera extrinsic matrix $[R, T]$, where R and T denote the rotation and translation matrix respectively. Then, the inverse of camera extrinsic matrix is computed to obtain the transformation matrix of camera with respect to marker, $[R_{cam}, T_{cam}]$ as follows:

$$[R_{cam}, T_{cam}] = [R, T]^{-1} = [R^T, -R^T T] \quad (2)$$

Experiments showed that slight chattering (to simulate the real condition of quadcopter when flying) can cause the roll ϕ and pitch angles θ of detected marker with

respect to camera to become unstable while the yaw ψ angle and translational distances remained stable. As a result, when these unstable values are applied to compute the inverse of camera extrinsic matrix $[R, T]$ by using equation (2), it will lead to an unstable result of $[R_{cam}, T_{cam}]$. To remedy this problem, a solution is proposed by utilizing IMU data of AR.Drone to obtain a more robust camera pose relative to marker. Assuming the marker is placed on the ground where AR.Drone takes off, then the roll and pitch angles measured by IMU of AR.Drone can be utilized in place of roll and pitch angles obtained from camera in computing the inverse of camera extrinsic matrix. The flowchart of this proposed algorithm is depicted in Fig. 6 where it takes camera image and IMU data as the inputs and gives the quadcopter pose relative to marker $\{X_b, Y_b, Z_b, roll_b, pitch_b, yaw_b\}$ as the outputs. A translation is made to move the frame of reference from the camera frame $\{X_c, Y_c, Z_c\}$ to the body frame $\{X_b, Y_b, Z_b\}$ since the camera is not located at body center of quadcopter.

Fig. 4 The result of original marker detection (marker pose relative to camera) with bottom camera and without filter when quadcopter in static condition.

Fig. 5 The result of modified marker detection (marker pose relative to camera) with bottom camera and Gaussian Blur filter when quadcopter in static condition.

Fig. 6 The flowchart of Vision-Based Quadcopter Pose Estimation Algorithm.

4. Controller Design

To design the controller for visual-servoing, a fuzzy logic control library called fuzzylite⁽¹¹⁾ is used. It is a free and open-source library programmed in C++, thus it is possible to integrate into ROS environment. From the flight

test experiments, it is found that each relationship of control input and output motions can be treated as an independent Linear Time-invariant (LTI) system where every control input does not affect the outputs of other inputs. Hence, three SISO (single-input and single-output) fuzzy controllers for roll, pitch and yaw motions are designed independently by employing Mamdani's fuzzy inference method. The goal of these controllers is to maintain the body center of quadcopter above the marker center. The membership functions of Roll, Pitch and Yaw Controller were tuned empirically and they are shown in Fig.7 to Fig. 10 respectively.

The fuzzy rules of Roll controller:

1. If (Displacement_X is BigLeft) then (CmdVel_Y is MoveBigRightward) (1)
2. If (Displacement_X is SmallLeft) then (CmdVel_Y is MoveSmallRightward) (1)
3. If (Displacement_X is Centered) then (CmdVel_Y is Zero) (1)
4. If (Displacement_X is SmallRight) then (CmdVel_Y is MoveSmallLeftward) (1)
5. If (Displacement_X is BigRight) then (CmdVel_Y is MoveBigLeftward) (1)

The fuzzy rules of Pitch controller:

1. If (Displacement_Y is BigFront) then (CmdVel_X is MoveBigBackward) (1)
2. If (Displacement_Y is SmallFront) then (CmdVel_X is MoveSmallBackward) (1)
3. If (Displacement_Y is Centered) then (CmdVel_X is Zero) (1)
4. If (Displacement_Y is SmallBack) then (CmdVel_X is MoveSmallForward) (1)
5. If (Displacement_Y is BigBack) then (CmdVel_X is MoveBigForward) (1)

The fuzzy rules of Yaw controller:

1. If (Orientation is BigLeft) then (CmdVel(ang_Z) is TurnBigRight) (1)
2. If (Orientation is SmallLeft) then (CmdVel(ang_Z) is TurnSmallRight) (1)
3. If (Orientation is Centered) then (CmdVel(ang_Z) is Zero) (1)
4. If (Orientation is SmallRight) then (CmdVel(ang_Z) is TurnSmallLeft) (1)
5. If (Orientation is BigRight) then (CmdVel(ang_Z) is TurnBigLeft) (1)

Fig. 7 Membership functions for Roll controller.

Fig. 8 Membership functions for Pitch controller.

Fig. 9 Membership functions for Yaw controller.

5. System Experiments

5.1 Vision-Based Quadcopter Pose Estimation

AR.Drone 2.0 and a notebook computer as ground-based computer were used to conduct the experiments. The ground-based computer runs the developed ROS programs which can communicate with AR.Drone. The AR.Drone driver in ROS⁽¹²⁾ is able to receive sensor data, camera image and send control commands to AR.Drone. The system architecture is shown in Fig. 10.

Fig. 10 The depiction of complete system architecture.

The output command follows the order of roll, pitch and lastly yaw is sent to AR.Drone every 0.2 seconds. The input limit for euler angle and yaw rate is set to 0.1 rad and 1.2 rad/sec respectively.

5.2 Experimental Results

The results of displacements and orientation of quadcopter relative to the ground marker by using fuzzy control are displayed in Fig. 11 and 12. These results are compared to that of onboard controller which is displayed in Fig. 13 and 14 where the quadcopter pose converges to zero value after 40 seconds because the quadcopter had moved away and hence the marker was outside the viewing angle of its bottom camera. These two controller performance in terms of hovering on top of ground marker are evaluated by calculating the root mean square error (RMSE) which are listed in Table 2. The RMSE of fuzzy control is calculated over the full time period of experiment: 0s ~ 60s while that of onboard controller is calculated over 0s ~ 40s because the marker was lost after 40s.

Table 2 Controller Performance Results.

Controller	RMSE: X (mm)	RMSE: Y (mm)	RMSE: Yaw (deg)
Fuzzy	93.9923	54.8103	2.6251
Onboard	252.3302	50.2430	3.1555

Fig. 11 X and Y displacement of quadcopter relative to ground marker with the designed Fuzzy Control.

Fig. 12 Orientation of quadcopter relative to ground marker with the designed Fuzzy Control.

Fig. 13 X and Y displacement of quadcopter relative to ground marker with the onboard internal controller.

Fig. 14 Orientation of quadcopter relative to ground marker with the onboard internal controller.

6. Conclusions

In this paper, we have presented a fuzzy control system added to onboard internal controller for AR.Drone to perform hovering on top of the ground marker. The performance of using the designed fuzzy controller is better in overall than onboard internal controller. Moreover, the fuzzy controller lets the quadcopter always hovering on top of marker during the experiment while onboard controller cannot. However, the fuzzy controller can be improved by introducing more inputs such as velocity difference and its performance will be evaluated to confirm its improvement. Besides hovering, other visual-servoing tasks such as following and landing can also be designed in future.

References

- (1) Anusha Gururaj, Jamkhandi , Saeed Tulpule , Ashvini Chaturvedi and Jean-Noel Charvet. "Controlling the Position and Velocity in Space of the Quad-Rotor UAV AR.Drone Using Predictive Functional Control and Image Processing in OpenCV," 2012 International Conference on Signal Processing Systems (ICSPS 2012), pp. 14 – 18.
- (2) Chi-Tinh Dang, Hoang-The Pham, Thanh-Binh Pham and Nguyen-Vu Truong, "Vision Based Ground Object Tracking Using AR.Drone Quadrotor," 2013 International Conference on Control, Automation and Information Sciences (ICCAIS), pp. 146 - 151
- (3) JeongWoon Kim·David Hyunchul Shim, James R. Morrison, "Tablet PC-based Visual Target-Following System for Quadrotors," J Intell Robot Syst (2014) 74:85–95.

- (4) Karl Engelbert Wenzel, Andreas Masselli, Andreas Zell, "Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle", J Intell Robot Syst (2011) , pp. 221–238
- (5) Tiago Gomes Carreira, "Quadcopter Automatic Landing on a Docking Station", Instituto Superior Tecnico, October 2013.
- (6) Karl E. Wenzel, Paul Rosset, Andreas Zell, "Low-Cost Visual Tracking of a Landing Place and Hovering Flight Control with a Microcontroller", J Intell Robot Syst (2010) 57: 297.
- (7) Jeffrey W. Tweedale, "Fuzzy Control Loop in an Autonomous Landing System for Unmanned Air Vehicles", WCCI 2012 IEEE World Congress on Computational Intelligence, pp. 1 - 8.
- (8) Miguel A. Olivares-Mendez, Ivan F. Mondragon, Pascual Campoy, Luis Mejias, Carol Martinez, "Aerial Object Following Using Visual Fuzzy Servoing", Workshop on Research, Development and Education on Unmanned Aerial Systems RED-UAS 2011, At Seville, Spain.
- (9) Stephane Piskorski, Nicolas Brulez, Pierre Eline, Frederic D'Haeyer: AR.Drone Developer Guide SDK 2.0, December 11, 2012.
- (10) <http://www.uco.es/investiga/grupos/ava/node/26>
- (11) <http://www.fuzzylite.com/>
- (12) https://github.com/AutonomyLab/ardrone_autonomy