Development of a Quadrotor with Vision-Based Target Detection for Autonomous Landing

Gervin Ernest C. Guevarra, Allen Rafael Tatsuya S. Koizumi, John Nicholas B. Moreno,
Jeremy Christian B. Reccion, Carl Michael O. Sy and Jay Robert B. del Rosario
Department of Electronics and Communications Engineering, Gokongwei College of Engineering,
De La Salle University-Manila, 2401Taft Ave., Malate, Manila 1004, Philippines

***Abstract*— In the field of robotics, the quadrotors have rapidly gained interest and have made several breakthroughs involving it, which range from variable pitch to application of swarm robotics. With that said, this paper aims to also expand upon one of the current developments which is the automated landing of quadrotors on a designated landing zone. Without GPS, the prototype built in this research employs image processing techniques to detect the landing zone, as well as to determine the flight altitude. Using these information, the quadrotor is autonomously controlled via its control surfaces (throttle, roll and pitch) in order to perform the landing procedure. Additionally, the quadrotor is capable of tracking a moving target and safely land even with winds reaching up to 2.2m/s.**

***Index Terms*—Automatic Landing; Quadcopter; Target Detection; Vision-Based.**

I. INTRODUCTION

Small-scale Unmanned Aerial Vehicles (UAVs), particularly multirotors, have been rapidly gaining interest among hobbyists and researchers alike.

Among the different types of multirotors is the quadrotor (also called quadcopter) - a lightweight craft propelled by four rotors. Quadrotors are popularly used for aerial photography and videography due to their great portability and lower cost, as compared to the traditional aerial shots using passenger helicopters. A rising sport, quadrotor racing, is also gradually earning the attention of hobbysists. In fact, the first ever World Drone Prix was held in Dubai last March 2016. There are also other quadrotor appplications including but not limited to surveillance [1] , mapping [2], and agriculture [3]. In the academe, quadrotors have seen several developments in the recent years and have since then used as a research platform for robotics, demonstration of artificial intelligence, flight control. [4] One of the interesting and challenging problems in quadrotor.

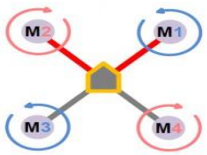

Figure 1: A quadrotor diagram showing the corresponding rotation of each motor (M1 to M4)

Roberts et al. have presented a quadrotor which employs minimal sensing in order to autonomously operate the quadrotor indoors [5]. Their system uses four infrared sensors facing each side, which are for drift correction and positioning. They also used downwards-pointing ultrasonic sensor for altitude control. The control strategy involves sensing the distance from the walls and attempting to maintain the quadrotor in the same position. However, their system is severely limited to the range and accuracy of the infrared sensors. In their paper, they have tested their system to work well inside an obstacle-free room measuring 6m by 7m. It is uncertain whether the system would still be able to achieve the same success had the room been larger.

Venables also worked on autonomous quadrotor flight [6]. He used color blob detection algorithm to detect objects on the ground. His algorithm, however, is GPS assisted and may no longer be applicable for indoor applications.

A lot of researches also delved with complex mathematical models to accurately define the dynamics of the quadrotor and control it, such as [7] and [8]. There are also those that implement SRUKF, SLAM or other 3D localization techniques, such as in [9], [10] and [11], where the the vision data is fused with the GPS data to determine the attitude of the craft. The control strategy in these papers proved to be effective, but may be resource intensive and IMU sensors tend to suffer from accumulated error. Moreover, these require that a mathematical model, which must be accurate to the quadorotor build, be created to be able to estimate and predict states.

In this paper, we present a quadrotor capable of landing on a marked landing area inside a GPS-denied environment. The quadrotor relies on a mounted downward-facing camera both for target tracking and altitude control, even without fusing the vision data with IMU data. It is the aim to develop a simpler control strategy by using only object-detection techniques and PD controller.

II. DESIGN AND IMPLEMENTATION

A. System Overview

The prototype system presented in this research can be divided into two subgroups: the ground station and the quadrotor itself. The ground station is the front-end and it consists of a laptop computer and the radio controller. The laptop is dedicated for monitoring and data gathering. It receives wireless telemetry data sent from the quadrotor. The radio controller (RC)is used to control the quadrotor movement, as well as the activation of the autonomous function.

The Arduino-Raspberry Pi coordination is the core of this system. The Arduino is mostly for the autonomous control and data transmission, while the Raspberry Pi is dedicated for image processing.

The quadrotor can be controlled in all of its principal axes (roll, pitch, yaw) by the control surfaces (aileron, elevator, throttle, rudder). Each control surface has a channel (signal wires) connecting the corresponding ports from the RC

receiver to the flight controller. Normally, PWM signals are passed from RC receiver to the FC. These PWM signals are generated in accordance to the input of a human on the RC controller. To control the quadrotor autonomously, Arduino is inserted between the connection of RC receiver to FC, and the period or width of the PWM signals are timed using interrupts. With that in mind, the FC could also be controlled by generating appropriate PWM signals from Arduino, thereby effectively gaining control of the quadrotor through Arduino.

Figure 2: Prototype of the quadrotor presented in this paper

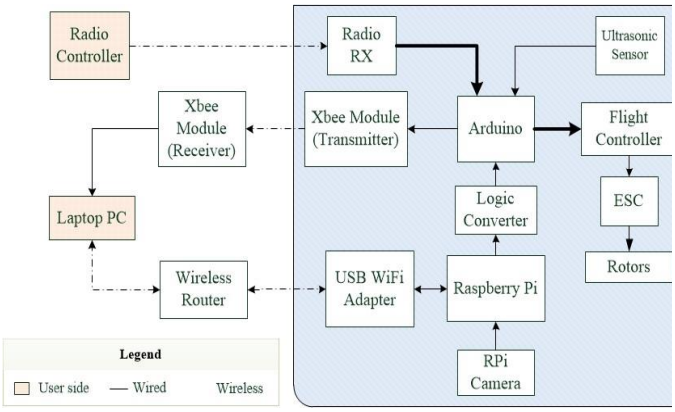

Figure 3: Complete block diagram of the system

B. Control Strategy

The system has two modes of control: manual and automatic. In manual mode, the quadrotor can be controlled as usual, using the RC transmitter. In this mode, the user has the full control of the quadrotor movement. In automatic mode, once engaged, assumes control and performs its alignment and landing routine.

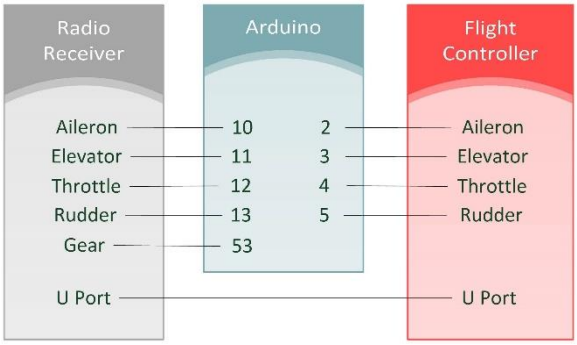

Figure 4: Block diagram of RC Receiver, Arduino, and Flight Controller connection

The control mode could be switched on or off using the auxiliary switch on the RC transmitter. The landing area, however, should be already visible to the quadrotor to successfully engage the automatic mode. As a safety mechanism, the quadrotor retains the manual mode in case the user attempts to engage the automatic mode even without the landing area nearby. Take note that the control strategy presented in this paper does not include 3D navigation and thus requires the user to control the quadrotor except for the landing, where the automatic mode is already an option. The automatic mode can be subdivided into three sub-components: Altitude-hold, Auto-align, and Autodescend. The altitude hold performs hovering at the instance the automatic mode is engaged. The process then proceeds to auto-align the quadrotor with respect to the center of the landing area using PD controllers for Aileron and Elevator. The algorithm regularly checks for the alignment of the quadrotor, and attempts realignment if the quadrotor drifts away. When the quadrotor is within an acceptable range near the center, it starts to decrease altitude.

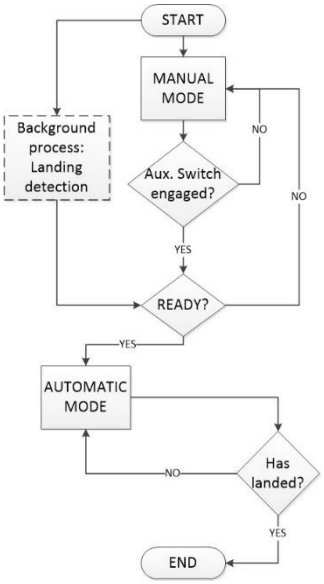

Figure 5: Flowchart for switching between Manual and Automatic Modes

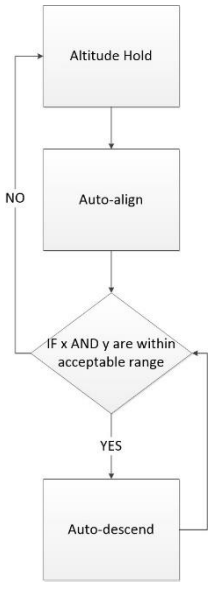

Figure 6 : Major blocks of the automatic mode

C. Landing Area Design

During experimentation, it was found out that a plain, circular red landing area cannot be detected correctly as the image processing algorithm fails to recognize a concrete contour once the quadrotor reaches lower heights during descent. Additionally, the camera's settings also change to preserve the white balance on the frame, which may lead to adverse effects on the data gathered by it, one of which is the change in the hue. The problem is resolved by adding concentric white circles, thereby allowing the algorithm to detect a good contour even at lower heights down to 0.25m.

D. Vision-based Altitude Measurement

The altitude measurement can be done in two ways: (1) using the radius calculations from the object-detection and (2) using the ultrasonic sensor. The radius of the detected area is inversely proportional to the altitude. Hence, the altitude can be known from the calculated radius from the target detection. However, this depends on the presence of the landing area in the cameras field of vision. In case that there is no landing area detected or if the quadrotor has drifted away too much, the altitude measurement will depend on the readings from the ultrasonic sensor. This acts as a safety mechanism as solely relying on the vision-based system could potentially cause a crash in the event that it has no reference (landing area) from the ground. Moreover, the difference between the measurements of the ultrasonic and the vision-based altitude is checked. This is a redundancy system to make sure that the altitude measurement is stable (not erratic).

Figure 7: Landing area design with white concentric circles. Radius of the largest red circle measures 0.2845m.

Due to the landing area having 3 regions at which the algorithm detects its radius, it would also need 3 equations in order to represent the relationship between the height and the radius. For clarities sake, the smallest circle would be called 1st circle, the middle circle is the 2nd circle, and the largest circle is the 3rd circle. After experimentation, the 1st and 2nd circles' radius to height relationship was able to produce linear equations which are, respectively:

$$h = -0.091r + 24.182 \quad (1)$$

$$h = -0.22r + 66.08 \quad (2)$$

while the 3rd circle was not able to be defined by merely a linear equation, but instead a power equation was used in order to more accurately define the relationship. This equation is:

$$h = 8733.8r^{-0.968} \quad (3)$$

Additionally, in order to properly define the switching of equations, the difference between the previous and current radius was compared to a set value, 40 in this case, since when the algorithm switches circles, there is an obvious difference on the radius, due to either the previous circle being bigger or smaller compared to the current circle. Thus, if this difference is negative, the equation used would transition from the bigger circles to the smaller circles, while if the difference is positive, the equation used would transition from the smaller circle's to the bigger circle's. Lastly, as a fail-safe the algorithm would automatically switch to the 3rd circle's equation if the radius seen is smaller than a constant value, 85 in this case since this is the smallest radius the other circles may provide.

E. Object-Detection

The script, written in Python, initializes the variables such as the mask values, frame resolution, and matrices. After this initialization, the script then starts to obtain the frames that the camera captures, which are then resized.

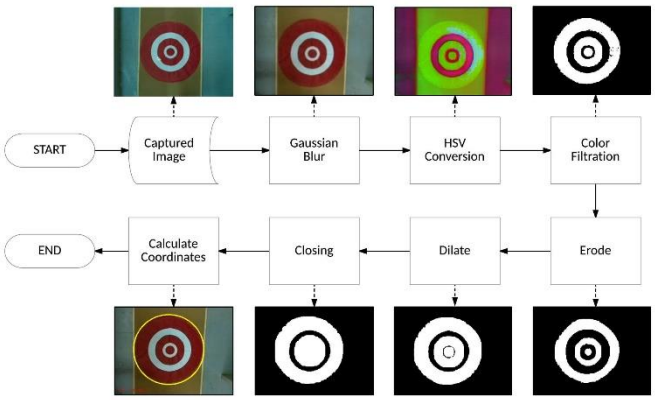

Figure 8: Flowchart of the object-detection program showing the output per process block.

The algorithms adopt the blob-detection technique resolution initialized beforehand. After which, the frame is then passed through a Gaussian filter to reduce the Gaussian noise present in the environment. Next, the frame is then converted into HSV color space from RGB in order to ease the process of color vision, which is more thoroughly explained in the previous section. Then the masks are used using the inRange function where its output would be a binary mask, where the white or foreground pixels are the parts which were allowed to pass by the mask. This mask is then processed through a series of erosions and dilations, collectively called morphologies. Opening, in which erosion is done followed by dilation, is done first. Then, the process is reversed closing, where dilation happens before erosion. These produce a frame that has reduced noise, while allowing the foreground to be more whole. After which, the findContours function is used in order to detect the white parts or blobs on the frame. The script would only go to the next step if it encounters at least one blob. In the case that it encounters multiple blobs, it would only recognize the largest blob using the max function. Afterwards, minEnclosingCircle() is used in order to determine the radius

and center of the blob, wherein it calculates the zeroth, and first order image moments to determine the centroid of the blob. As for the radius, it detects three points that form a line that passes through the blob, this lines length is then measured in order to determine the diameter, and subsequently the radius. If the radius of the blob is more than a set value the script proceeds, it calculates the distance of the center of the blob to the center of the camera. Lastly, the coordinates are then formatted by converting them into string. These are sent to the Arduino via serial.

III. Testing and Results

A. Landing Area Shape

A shape test was conducted in order to determine the optimal shape for calculating the centroid of the target using the image moments method. This test would be evaluated by calculating the standard deviation of the coordinates per shape, which were gathered over a period of 30 seconds. Take note that the shapes and the quadrotor are fixed in position during the test period to avoid unnecessary jitters in the recording of the coordinates. From the results shown on Table 1, the circle shape has the least deviation among the three shapes. Hence, it is chosen as the shape of the landing area and is used in the subsequent tests.

Table 1
Coordinate Deviation per Shape

Shape	Coordinate Deviation	
	x	y
Circle	0.584463682	0.338848367
Triangle	0.590398064	0.542359344
Square	0.622039634	0.797724035

B. Resolution vs Processing Time Test

The results conform with the hypothesis that higher resolutions would yield slower processing time, and vice versa. Thus, further experimentation was done in order to determine that the resolution/frequency which interacted with the flight control algorithm, wherein it produced the best behaviour during landing, was the 320 x 240 resolution with 12.17 Hz frequency. Therefore, in the final implementation this resolution would be used.

Table 2
Processing Time per Camera Resolution

Resolution (pixel x pixel)	Ave Processing Time(s)	Frequency (Hz)
240 x 240	0.061124	16.36028
250 x 250	0.071483	13.98939
320 x 240	0.082202	12.16518
320 x 320	0.108091	9.251472
640 x 480	0.332845	3.004397

C. Altitude Hold Test

The desired altitude is set around 1.57 meters and the data is recorded for 35 seconds. Figure 9 shows the graph between the vision-based altitude values and the desired altitude. The maximum absolute error calculated from the actual visionbased data is 0.1119 meters.

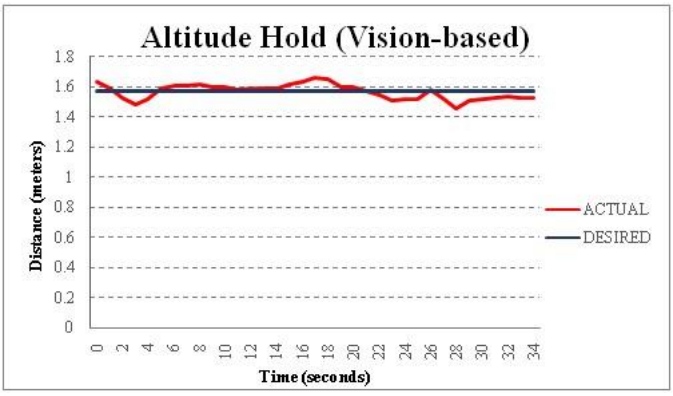

Figure 9: Vision-based altitude compared to the desired

IV. Conclusion

With the recent rise of drone usage and technology, the number of users who are opting to use drones in their field of study and expertise have also risen, but with the drones difficulty of control it may not be as simple as buying one and using it right after without proper experience as the drone may experience crashes, and such. One particular reason for crashes is during landing, especially in enclosed areas due to the low maneuverability inherent to such areas. As such, the researchers have provided a different means to alleviate these concerns with the use of a quadrotor with camera attachment that possesses an algorithm that allows it to land on a specific target with the flip of a button. This algorithm allows the quadrotor to land on a stationary circular red target, for 10 tests, where the algorithm was engaged at a starting altitude averaging at 1.72m, it produced an average deviation of 0.11 m, measured from the center of the landing area to the center of the quadrotor, and a landing time averaging to 7.87 seconds. This data shows a significant improvement to the control test, where it lands without tracking the target, due to it having an average deviation of 1.47 m away from the target, over 10 tests. This calculates to a reduction of deviation of 92.52%. Additionally, the algorithm was also tested for when the targets was moved by 1 m, and when the quadrotor was subjected to wind speeds of 2.2m/s, where it produced an average deviation of 0.14 m and 0.12 m, respectively.

Acknowledgment

The authors would like to thank Dr. Jonathan Dungca, Dean of the Gokongwei College of Engineering and Engr. Edwin Sybingco, Chairman of the Electronics and Communications Engineering Department of De La Salle University-Taft, Manila.

References

[1] L. Geng, Y. Zhang, J. Wang, J. Fuh, and S. Teo, ”Mission Planning of Autonomous UAVs for Urban Surveillance with Evolutionary Algorithms,” in Control and Automation (ICCA), 2013 10th IEEE International Conference, 2013, pp. 828833

[2] M. Nagai, T. Chen, R. Shibasaki, H. Kumagai, and A. Ahmed, ”UAVborne 3-D Mapping System by Multisensor Integration,” Geoscience and Remote Sensing, IEEE Transactions on, vol. 47, no. 3, pp. 701708, 2009.

[3] P. Aquino, D. Dela Cruz, K. Teves, J. Trinidad and K. Villacer, ”Design of an Android-App based crop dusting Hexacopter”, Undergraduate, De La Salle University - Manila, 2016.

[4] A. Gautam, P.B. Sujit, S. Saripalli, ”A Survey of Autonomous Landing Techqniques for UAVs,” in Unmanned Aircraft Systems (ICUAS), 2014 IEEE International Conference pp. 1210-1216.

[5] J. Roberts, T. Stirling, J. Zufferey and D. Floreano, ”Quadrotor Using Minimal Sensing for Autonomous Indoor Flight,” in European Micro Air Vehicle Conference and Flight Competition (EMAV2007), Toulouse, France, 2007, pp. 1-8.

[6] C. Venables, ”Multirotor Unmanned Aerial Vehicle Autonomous Operation in an Industrial Environment using Onboard Image Processing”, School of Electrical, Electronic & Computer Engineering, University of Western Australia, 2013.

[7] J. Haines, ”Vision-based Control of a Multi-rotor Helicopter”, Master of Science in Robotics, Pittsburgh, Pennsylvania, 2011.

[8] J. Daly, Y. Ma, S. Waslander, ”Coordinated Landing of a Quadrotor on a Skid-Steered Ground Vehicle in the Presence of Time Delays,” in Intelligent Robots and Systems (IROS), 2011 IEEE Conference pp.4962-4966.

[9] S. Yang, J, Ying, Z. Li, ”Precise Quadrotor Autonomous Landing with SRUKF Vision Perception,” in Robotics and Automation (ICRA), 2015 IEEE International Conference, 2015, pp. 2198-2201.

[10] S. Holmes, G. Klein, D. Murray, ”An O(N) Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.31, No.7, pp.1251-1263, 2009.

[11] F. Ababsa, ”Advanced 3D Localization by Fusing Measurments from GPS, Intertial and Vision Sensors,” Systems, Man and Cybernetics (ICSMC), 2009 IEEE International Conference, 2009, pp. 871-875.