Proceedings of the EuroGNC 2013, 2nd CEAS Specialist Conference
on Guidance, Navigation & Control, Delft University of Technology,
Delft, The Netherlands, April 10-12, 2013

Automatic Landing System of a Quadrotor UAV Using Visual Servoing

H.W. Ho and Q.P.Chu

Abstract This paper presents a fully autonomous system for a quadrotor unmanned aerial vehicle (UAV) that employs the visual measurements to perform automatic landing task on a specific landing platform. Basically, there are two main control tasks discussed in this paper. The first task refers to the auto-land mission that tracks the platform horizontally and performs the vertical landing. It was accomplished by using the red blob tracking technique. The second task involves the robust motion tracking that activates the recovery action once the target is lost so that the vehicle is still able to hover steadily. It was realized by implementing the features accelerated from segment test (FAST) technique with the pyramidal Lucas-Kanade algorithm to detect the local feature in the image and compute the optical flow. From the visual results obtained, the position and velocity of the vehicle were estimated using a nested Kalman-based sensor fusion . The state estimation was validated in a series of experiments using a CNC milling machine. Lastly, the control architecture for automatic landing system was formed with the classical PID controller and the flight test proved the success of the proposed system.

1 Introduction

The recent development of the UAV system has motivated the growth of the cost effective inertial sensors and accurate navigation system. These advances are the key determinants of the accomplishment of the UAV mission. For many years, the successes of manned and unmanned aircraft navigation have been achieved using the standard conventional navigation sensors, such as GPS for providing position information, IMU for measuring orientation and specific forces, and pressure sen-

H.W. Ho
Delft University of Technology, 2600GB Delft, The Netherlands, e-mail: hhannwoei@gmail.com

Q.P.Chu
Delft University of Technology, 2600GB Delft, The Netherlands, e-mail: Q.P.Chu@TUDelft.nl

sor for giving altitude measurement. Other modern sensors, like ultrasound, laser range finder, camera, etc., were also introduced to replace or fuse with the conventional sensors to perform some specific tasks. In fact, our particular interest involves a quadrotor UAV landing on a target platform. Thus, the GPS navigation technique is not reliable for this particular purpose which required high accuracy in positioning. An alternative way of measuring position is based on computer vision. Many computer vision techniques using camera have been implemented in ground robots to improve their autonomy and adaptability to operate in unstructured environment. In comparison with typical sensors, visual sensors are passive, i.e. do not emit any external signals, small, lightweight, and can provide rich information about aircraft self-motion and surroundings structure [5], [7], [8] . Hence, computer vision can be used for GPS-denied indoor, urban navigation, obstacle avoidance and accurate landing purposes. In this paper, Section 2 described a general idea on how an automatic landing of a quadrotor can be performed using visual servoing techniques. A new approach that combined two computer vision techniques was introduced to guarantee the success of the landing process and ensure the safety of the quadrotor when the target is not in the field of view (FOV). The algorithms of those techniques were further discussed in Section 3. Once we have obtained the visual information, the states of this flying robot were estimated using a Kalman-based sensor fusion method from [3] which was improved by fusing an additional altitude measurement from a Sonar sensor and it was presented in Section 4. The validation of the results from state estimation was shown in Section 5. In Section 6 the controller design for automatic landing tasks was proposed and also the real-time implementation of the algorithms was described in Section 7. Finally, a conclusion was drawn in Section 8 based on the results and discussions from previous sections.

2 Design Solution

An overview of the design solution using computer vision for automatic landing algorithm is described in this section. It is designed based on several logical scenarios that could happen during the landing process. In general, the proposed design solution for the project is illustrated in Fig. 1.

When the autopilot landing mode is activated, the quadrotor is guided by GPS Navigation Mode to move to a way point near and above the landing platform. Then the target searching mode is triggered to capture images continuously using onboard camera and perform red blob detection to search for the landing platform. As long as the platform is located in the FOV of the camera, blob tracking is carried out in order to obtain its position relative to the platform and the quadrotor then descends slowly. Until an altitude less than 20 *cm* and the vehicle is hovering closely to the center of the platform, it will land upon it. This is a perfect landing solution and can be easily accomplished by blob detection and tracking method.

However, when there is a small disturbance, such a gust or an airflow generated by itself, during the landing process, the vehicle will be forced away from the plat-

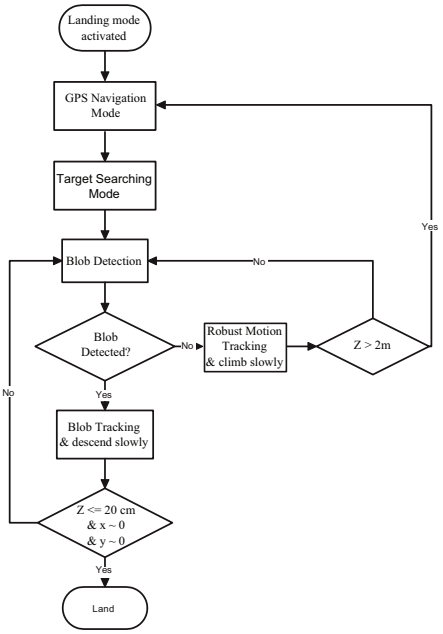

Fig. 1 Design Solution of An Automatic Landing Algorithm

form. Then the target is not in the FOV of the camera and the state information of the aircraft will be lost. The solution for this problem is to use or detect the local features in the image, such as corners, and perform corner tracking to compute the optical flow. Once we have the optical flow, the horizontal velocity of the vehicle can be estimated. Then the velocity feedback enables the quadrotor to hover and climb slowly until it detects the target again. As a safety precaution, the landing autopilot mode will be switched back to GPS navigation mode and the quadrotor UAV will be repositioned to the starting way point where the landing autopilot mode was activated if the relative height is above 2 m. Lastly it will perform the landing process using the blob tracking method. It is called as a robust motion tracking due to the fact that it does not depend on the visibility of the platform in the images but it relies on the local features in the current image.

3 Computer Vision Techniques

The proposed computer vision techniques consist of a target detection and a robust motion detection. Both methods are working in parallel in order to make sure that the visual information is always available for the state estimation of the vehicle during the landing process. The results of the state estimation are used by the visual servoing to perform the auto-land mission.

3.1 Target Detection and Tracking

In this project, it is essential to determine the position of the quadrotor relative to the landing platform. Therefore, a visual cue can be placed at the center of the landing platform to allow the computer vision to detect it based on images taken by the onboard camera and then compute its relative position. The closed-loop control will track the target using this relative position and land upon the platform.

3.1.1 Blob Detection and Tracking

A blob is a commonly-used object for target detection and tracking in computer vision. Due to the simplicity and feasibility, a red-colored blob with a solid circular shape of a radius of 135 *mm* is designed as an object of the target detection. For obtaining the relative position, an algorithm of red blob detection and tracking was developed. Basically, the goal of this algorithm is to eventually calculate the geometric center of the red blob. Thus, the following steps were taken to detect the red blob and compute its centroid:

1. Extraction of the Blue Green Red Color Mode (BGR) values of every pixel. A BGR image captured by a camera is represented by a grid of numbers or pixels. Each pixel is actually a vector with three parameters which have a value of 0 to 255 (8-bit integer) as shown in Eq. 1.

$$Image = \begin{bmatrix} (B,G,R)_{11} & (B,G,R)_{12} & \dots \\ (B,G,R)_{21} & (B,G,R)_{22} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{1}$$

2. Thresholding/ Conversion to a binary image. After getting these parameters, the BGR image can be easily converted to binary image by giving a value of one for which a certain condition is satisfied and vice versa. This condition defines which pixels are red. Theoretically, a pure red pixel is described by $[0,0,255]$ in BGR format. However, searching for pure red pixel is not practical as the change of light intensity and the presence of noise will result in the change of this absolute value. Hence, some tolerances need to be given for accepting the red-like object. In this case, the condition of selecting the red pixel is presented in Eq. 2.

$$R > T + B \quad and \quad R > T + G \tag{2}$$

   where T is the tuning parameter. If T is large, only the pure red is accepted and vice versa.
3. Advanced morphological transformation of the binary image. Since the thresholding is a subjective process, the resulting binary image may contain unwanted noise. Hence, one of the advanced morphological transformations, i.e. the open-

ing morphology can be performed to remove those noise particles. It is obtained by the erosion of an image followed by the dilation process.

4. Labeling of the connected parts of a binary image. This step uses an algorithm based on a paper from [2] to extract the contour of the blob. The advantage of implementing this approach is that it scans a binary image only once and thus it is computationally effective.
5. Detected Blob Centroid Calculation. The centroid of a blob is computed using the detected blob area as shown in Eq. 3. The advantage of computing the area centroid is that when only part of the circle appears in the FOV of the camera it also allows the vehicle to know the best estimated relative position between the target and itself.

$$\begin{aligned} \bar{\mathbf{x}} &= \frac{total\ moments\ in\ x-direction}{total\ area} = \frac{\sum_n A_n \bar{x}_n}{\sum_n A_n} \\ \bar{\mathbf{y}} &= \frac{total\ moments\ in\ y-direction}{total\ area} = \frac{\sum_n A_n \bar{y}_n}{\sum_n A_n} \end{aligned} \tag{3}$$

3.2 Robust Motion Detection and Tracking

As mentioned in Section 2, disturbances can cause the target out of the FOV of the camera during the landing process. Therefore, it is necessary to find out another way to obtain the state of the vehicle in order to feed it into the control loop. A robust motion detection and tracking is introduced in this section aiming for obtaining the velocity of the rotorcraft even without the presence of the target platform in the FOV of the camera.

3.2.1 Features Detection and Tracking

In section 3.1.1, a specific target is designed for platform tracking purpose. Without seeing the target, the questions on what is the kind of the local features can be trusted and how to detect and trace a selected good feature were addressed. In fact, corners were considered intuitively as the good features and were used in robust motion tracking as they are unique and contain enough information to be picked out from one frame to the next.

Furthermore, the selection of features detector depends on the mission tasks. In this project, features detection and tracking are performed in real-time processing. Therefore, the processing speed is the top priority followed by the quality of the features detected. A comparison of the performance between different feature detection algorithms[1] has been done and it was used as a reference for the selection.

[1] Eugene, K.: Comparison of the OpenCVs Features Detection Algorithm (2012). URL http://computer-vision-talks.com/

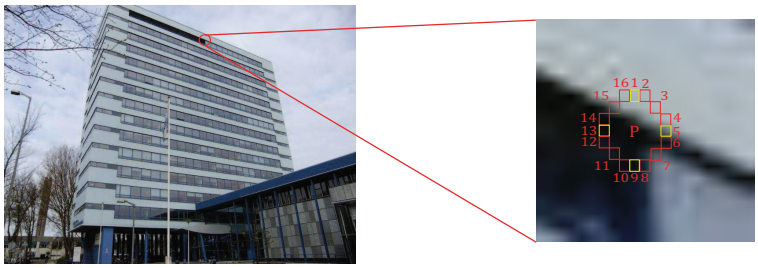

Fig. 2 FAST

(a) GFTT: Quality level = 0.1;
no. of detected corners = 149; time = 29.066 ms

(b) FAST: Threshold = 45;
no. of detected corners = 143; time = 0.837 ms

Fig. 3 Comparison between GFTT and FAST

FAST algorithm, by its name, has the fastest average detection time if compared to other features detectors. Regarding the quality of this features detector, its average feature point drift is low.

According to [6], a point of interest or a pixel P in Fig. 2 is tested by examining a circle of 16 pixels surrounding this point at a fixed radius. If at least 12 contiguous pixels have the intensities above or below the intensity of point p by some threshold, this point of interest can be considered as a good feature, i.e. a corner. Furthermore, the test is optimized by first examining the pixel 1, 9, 5 and 13. The candidate pixels can be rejected quickly if two of these four testing pixels do not satisfy the condition.

In order to validate its performance, a test has been carried out to compare its performances with a well-known Shi and Tomasi corner detection (GFTT). To make them comparable, the quality level of the GFTT and the threshold value of FAST were tunned in such a way that the numbers of detected corners from both methods were almost the same. After that, the processing speeds were measured and the detected features were compared. In Fig. 3, we can observe that these two methods detected almost the same corners. However, FAST can achieve approximately 30$\times$ faster than GFTT, i.e. it required only 0.837 *ms* while GFTT needed 29.066 *ms*. Hence, FAST is preferable as a corner detector for the real-time implementation.

In this project, we are interested to know the motion of the quadrotor based on a sequence of images taken by an onboard camera. Between two subsequent frames,

a pattern of apparent motion of the local features caused by the relative motion between the camera and the features can be observed and it is called the optical flows. To compute the optical flow, a famous gradient technique, Lucas-Kanade (LK) method, is used to derive the local information from a small window surrounding each of the points of interest [4].

4 State Estimation Using Sensor Fusion

From the result of the computer vision, the position of the vehicle relative to the platform was computed from the blob tracking while the optical flow was also calculated from the corner tracking. Now, those measurements in pixel have to be transformed into measurements in physical unit. It was done using the information from multiple sensors and camera calibration. The camera calibration was carried out using a method proposed by [9] to solve for the camera intrinsic parameters, i.e. the focal length in pixel and the correction for the image center, and obtain the distortion model parameters using [1] model.

4.1 Position Estimation

The relative position was estimated using a simple pinhole camera model as shown in Fig. 4. Focal length in pixel, f was obtained from the camera calibration and height, Z was measured using Sonar sensor. After the relative position, x_i in pixel was computed from the blob tracking, the horizontal position of the vehicle in the real world relative to the blob, X_o can be calculated using Eq. 4.

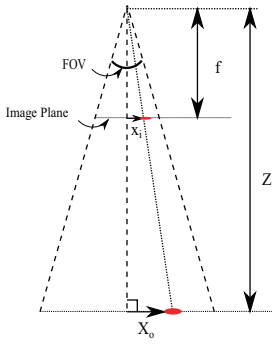

Fig. 4 Pinhole Camera Model

$$x_i = f\frac{X_o}{Z} \tag{4}$$

4.2 Velocity Estimation

Since quadrotor UAV is an under-actuated system, its horizontal motion is generated by the pitch and roll angular displacements. Therefore, a single camera mounted on the body of the vehicle measures the optical flow $(\dot{x}_{OF}, \dot{y}_{OF})$ from both translational (T_{OF}) and rotational (R_{OF}) motion as presented in Eq. 5. The relation of the optical flow with the vehicle motion can be derived in terms of body velocities (V_x, V_y, V_z) and angular rates $(\Omega_x, \Omega_y, \Omega_z)$ as given in Eq. 6. x and y are the corner coordinates in body x and y axes.

$$\begin{bmatrix} \dot{x}_{OF} \\ \dot{y}_{OF} \end{bmatrix} = T_{OF} + R_{OF} \tag{5}$$

$$\begin{bmatrix} \dot{x}_{OF} \\ \dot{y}_{OF} \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix}\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} + \begin{bmatrix} \frac{xy}{f} & -f-\frac{x^2}{f} & y \\ f+\frac{y^2}{f} & -\frac{xy}{f} & -x \end{bmatrix}\begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \tag{6}$$

In Fig. 5, the quadrotor rotates to move to the left side. At the instant of rotation, it measures a certain amount of optical flow and this causes the velocity estimated in an opposite direction. As a result, it will in turn cause a problem in the controller action. For instance, the velocity which is used for motion damping in the controller will eventually act as an accelerator.

4.2.1 Kalman-based Sensor Fusion

To overcome this problem, a kalman-based sensor fusion was proposed to eliminate the rotational optical flow and estimate the horizontal velocity. From [3], the

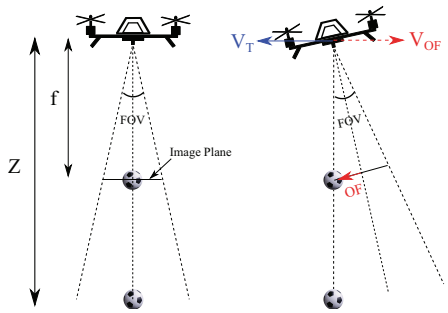

Fig. 5 Rotation Effect on Optical Flow

camera-IMU fusion was introduced to solve structure from motion problem. The 3-nested Kalman Filter (KF) was modified to fuse camera, IMU and Sonar sensors to improve the accuracy of the horizontal motion estimation. Fig. 6 presented the overview of a 3-nested KF where the estimated states for each step were used as the measurements for the following step. The first KF (KF I) was used for filtering the noise of the visual measurements. Furthermore, the second KF (KF II) fused the camera with IMU to compensate the rotational effect on optical flow. Lastly, the estimated translational optical flow was used in the last KF (KF III) for the horizontal velocity estimation using altitude measurement from Sonar and accelerations from IMU.

The discrete-time state space representation of the 3-nested KF is presented in details in Table 1. The dynamics of the KF I was modelled by using the Euler method and the brownian process. For the KF II, besides using the brownian process for its dynamics, the observation model was derived from Eq. 6 by replacing the first term in the equation with a vector $[\hat{x}_{OF_{trans}}, \hat{y}_{OF_{trans}}]^T$. Lastly, the KF III refers to an extended KF due to the nonlinear obvervation model which was obtained by

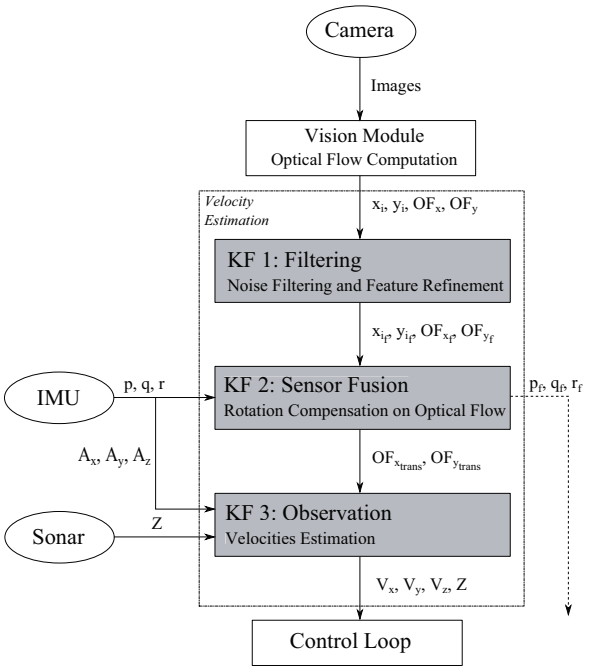

Fig. 6 Kalman-based Sensor Fusion for Rotation Compensation of the Optical Flow and Estimation of the Horizontal Velocity

equaling the first term from both Eq. 5 and Eq. 6 while its dynamic model was also obtained using Euler method.

Table 1 Kalman-based Sensor Fusion

	KF I	KF II	KF III
KF Model	$\underline{X_1}_{k+1} = A_{1_k}\underline{X_1}_k + \underline{w_1}_k$ $\underline{Z_1}_{k+1} = C_1\underline{X_1}_{k+1} + \underline{v_1}_{k+1}$	$\underline{X_2}_{k+1} = A_{2_k}\underline{X_2}_k + \underline{w_2}_k$ $\underline{Z_2}_{k+1} = C_2\underline{X_2}_{k+1} + \underline{v_2}_{k+1}$	$\underline{X_3}_{k+1} = A_{3_k}\underline{X_3}_k + B_{3_k}U_{3_k} + \underline{w_3}_k$ $\underline{Z_3}_{k+1} = \underline{h}\left[\underline{X_3}_{k+1}\right] + \underline{v_3}_{k+1}$
State vector	$\underline{X_1} = [\hat{x}, \hat{y}, \hat{x}_{OF}, \hat{y}_{OF}]^T$	$\underline{X_2} = [\Omega_x, \Omega_y, \Omega_z, \hat{x}_{OF_{trans}}, \hat{y}_{OF_{trans}}]^T$	$\underline{X_3} = [V_x, V_y, V_z, Z]^T$
State Matrix	$A_1 = \begin{bmatrix} 1 & 0 & \Delta t_k & 0 \\ 0 & 1 & 0 & \Delta t_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$A_2 = I^{5\times5}$	$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\Delta t_k & 1 \end{bmatrix}$
Input Vector	-	-	$U_3 = [A_x, A_y, A_z]^T$
Input Matrix	-	-	$B = \begin{bmatrix} \Delta t_k & 0 & 0 \\ 0 & \Delta t_k & 0 \\ 0 & 0 & \Delta t_k \\ 0 & 0 & 0 \end{bmatrix}$
Measurement vector	$\underline{Z_1} = [x_m, y_m, \dot{x}_{OF_m}, \dot{y}_{OF_m}]^T$	$\underline{Z_2} = [\hat{x}_{OF}, \hat{y}_{OF}, \Omega_{x_m}, \Omega_{y_m}, \Omega_{z_m}]^T$	$\underline{Z_3} = [\hat{x}_{OF_{trans}}, \hat{y}_{OF_{trans}}, Z]^T$
Output Matrix	$C_1 = I^{4\times4}$	$C_2 = \begin{bmatrix} \frac{\hat{x}\hat{y}}{f} & -f - \frac{\hat{x}^2}{f} & \hat{y} & 1 & 0 \\ f + \frac{\hat{x}^2}{f} & -\frac{\hat{x}\hat{y}}{f} & -\hat{x} & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ $\underline{h}\left[\underline{X_3}_{k+1}\right] = \begin{bmatrix} \frac{-f}{Z}V_x + \frac{x}{Z}V_z \\ \frac{-f}{Z}V_y + \frac{y}{Z}V_z \\ Z \end{bmatrix}$	

w_1, w_2, w_3 are the state noise vectors with covariance matrices Q_1, Q_2, Q_3 respectively.
v_1, v_2, v_3 are the measurement noise vectors with covariance matrices R_1, R_2, R_3 respectively.

5 Validation for the State Estimation

It is a challenging task to validate the estimated position and velocity with a reliable source of reference. In fact, the conventional way to obtain the states of the UAV, i.e. using GPS, cannot be implemented for the validation because the noise of the GPS measurement can cause an error up to 10 m while we expected the camera can produce centimeter or even milimeter accuracy. Besides, validation using external cameras is not wise as it is also using the similar approach, i.e. the computer vision techniques to compare both results. Hence, an accurate source of reference with different calculation approach is needed to validate the estimated velocities. Thus, a validation method using computer numerical control (CNC) machine was proposed due to the fact that it has been widely used in precision engineering to produce highly accurate and precise movement. Therefore, the quadrotor can be attached to it and moved with the known position and velocity references. The setup of the validation test is presented in Fig. 7. The CNC machine used in the test is *Superior 1600* which has a positioning speed of 4000 mm/min, accuracy of repetition of $+/-$ 0.02 mm, and maximum positioning error less than 0.1 $mm/100$ mm. However, the drawback of using this CNC machine is that the real-time speed information is not available. In order to command the machine, a movement path needs to be designed in *XpertMill* software to move the quadrotor to a specific position with a designated speed.

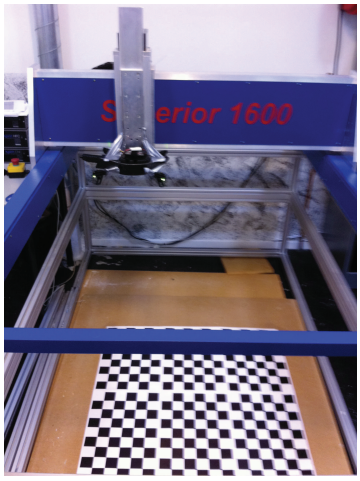

Fig. 7 Setup of the Validation Test for State Estimation

5.1 Position Validation

A test using CNC machine was carried out to check the accuracy of the estimated position. The movement path designed in *XpertMill* as shown in Fig. 8 moved the quadrotor to a distance of 50 *mm*, 100 *mm*, and 150 *mm* from the origin where a red blob was placed exactly underneath the onboard camera.

The relative position of the vehicle was estimated using Eq. 4 and compared with the ground truth as shown in Fig. 9. It is clearly seen that the position estimated from the result of the computer vision technique can achieve milimeter accuracy. Therefore, it validated that the proposed method is feasible to deliver accurate position of the quadrotor relative to the landing platform.

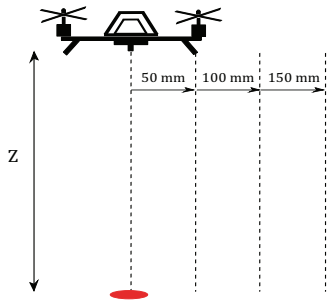

Fig. 8 Movement Path for Position Validation (Side View)

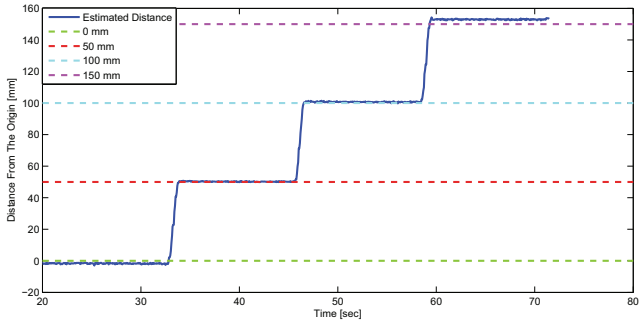

Fig. 9 Comparison of The Distance Estimated From Blob Tracking And The Ground Truth

5.2 Velocity Validation

Besides position validation, the body velocity estimated using Kalman-based sensor fusion from the optical flow also needs to be validated. There are two movement path designed for this purpose: 1. Rectangle 2. Triangle

5.2.1 Rectangular Movement Path

The easiest way to validate the estimated body velocities is to use CNC machine to move the quadrotor in a rectangular path as shown in Fig. 10. In this figure, the red dot represents the starting and end point. Besides, the speed has a designated value of 2000 mm/min (33.33 mm/sec) and the test was performed for two rounds in order to test the consistency of the estimated results.

Fig. 11 presented the estimated velocities in body-x and body-y axes of the quadrotor moving at 2000 mm/min. It is clearly seen that both estimated velocities from the optical flow can actually match well to the machine velocity and the consistency is extremely good as the quadrotor accelerated at the beginning, reached the constant velocity, and lastly decelerated to nearly zero. There were two spikes in the figure, one at the beginning and another one in the end, generated by the machine when it moved from its zero reference point to the starting point of the path. Thus, it can be ignored as it is irrelevant to the test. Hence, we can conclude that the horizontal body velocities estimated using optical flow with Kalman fusion are accurate and precise.

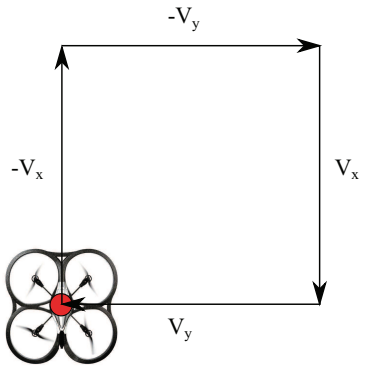

Fig. 10 Rectangular Movement Path (Top View)

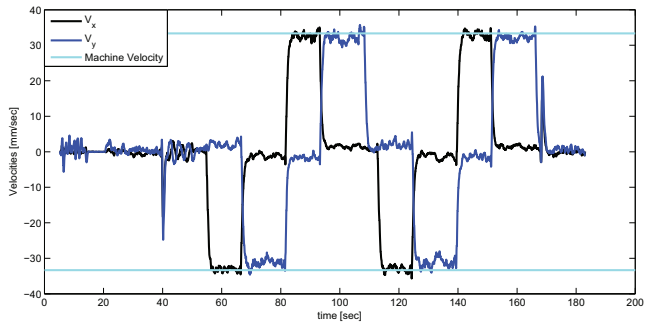

Fig. 11 Estimated and Machine Velocities in Body Axes of the Rectangular Movement path (2000 mm/min)

5.2.2 Triangular Movement Path

Additionally, it would also be interesting to see what the estimated horizontal velocities will be when the quadrotor moves not just along its body axes. From this idea, a triangular movement path as presented in Fig. 12 was designed to examine the estimated results.

Fig. 13 presented the estimated velocities in body-x and body-y axes of the quadrotor moving at 2000 mm/min. As we expected, when the quadrotor was moved in the hypotenuse of this triangular path, it measured the components of the machine velocity in body axes. Since the movement path is a right isosceles triangle, the estimated velocities in body axes should be the cosine of fourty-five degree of the machine velocity. The results are surprisingly good and the horizontal velocities of the quadrotor can be accurately estimated as we expected.

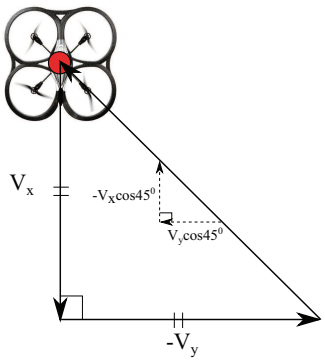

Fig. 12 Triangular Movement Path (Top View)

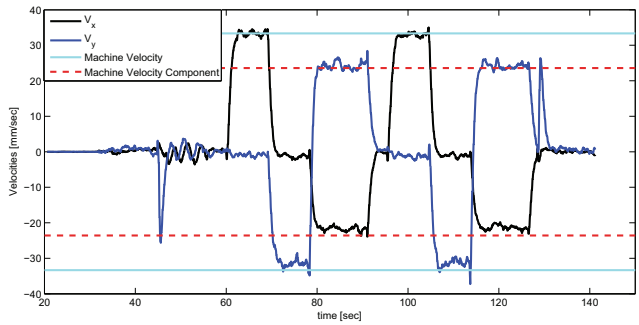

Fig. 13 Estimated and Machine Velocities in Body Axes of the Triangular Movement path (2000 mm/min)

6 Controller Design

Since a quadrotor UAV operating with four actuators has six degree of freedom in flight, it is thus considered as an under-actuated vehicle. Designing a controller to perform a particular task for this kind of vehicle is a challenging task. Moreover, it is recognized as dynamically unstable meaning that a closed-loop control is necessary.

6.1 Visual Servoing

The feedback control technique using computer vision is called visual servoing. In this case, indirect visual servoing is preferable because the visual information is used and obtained in the outer loop so that it allows a lower sample rate for visual control.

Furthermore, a more detailed control architecture designed to achieve the goal of this project is illustrated in Fig. 14. It mainly consists of two nested control loops in the design, i.e. inner and outer loops. The inner loops are responsible for attitude and angular rate control while the outer loops are in charge of position and velocity controls. It can be observed in the figure that the outer loops are further divided into horizontal and vertical control loops. In addition, for safety and other purposes, the control design in the outer loop is made so that the pilot can always intervene the autopilot control by using a joystick input. State estimation using sensor fusion, as discussed in Section 4, is also presented because it is needed to coorperate with the control strategy.

In this proposed control architecture, the preferred control method is the classical PID control as it is not only recommended by many literatures and also it is simple to be implemented as well as it can perform well. The PID control structure

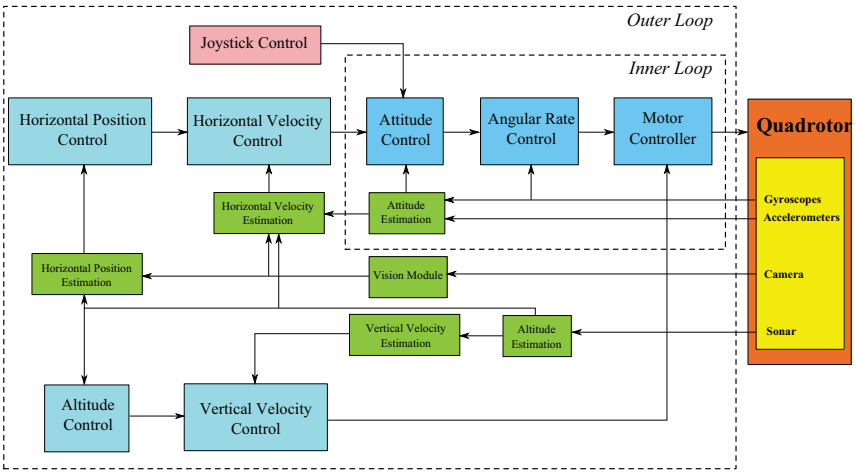

Fig. 14 Control Architecture for Autonomous Landing System

in its complete form consists of three components, i.e. Proportional, Integral, and Derivative parts as presented in Eq. 7.

$$\mathrm{u}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{7}$$

where K_p, K_i, K_d are the PID gains, e is the error, t is the instantaneous time, and τ is the variable of integration which values computed from starting time to the current time.

The inner control loops in the control achitecture contain the attitude control loop and angular rate control loop. The first one calculates an angular rate setpoints based on the errors of the estimated attitudes and the attitude setpoints.

The attitudes are estimated from the fusion of gyroscope and accelerometer measurements. This fusion of inertial sensors is used to correct for the biases during the initialization of the flight. During hovering, the attitude setpoints are assigned to zero values. Those computed angular rate setpoints are tracked with a PI controller in the angular rate control loop. Then this control loop controls the motor with the proportional controllers in the inner loop.

The outer control loops are divided into the horizontal and vertical control loops. The horizontal control loop performs the hovering and the target tracking using visual servoing while the vertical control loop also supports the hovering and carries out the landing process using the feedback of the Sonar measurement. Basically, the error between the desired position, x_{sp} and the estimated position, x_m, and the error between the desired velocity, V_{sp} and the estimated velocity, V_m of the vehicle are feedback to make sure that the desired position is tracked and the motion of the quadrotor is damped correspondingly. Therefore, this controller action can be interpreted as a PD controller as presented in Eq. 8. For horizontal control loop, x_{sp}

is set to zero value while for vertical control loop, it is set to an appropriate value for hovering, landing, and recovery tasks in the autopilot. For both control loops, V_{sp} is assigned as zero value. In addition, the controller gains (K_p, K_d) were tunned manually in the flight to optimize its performance practically.

$$\mathrm{u}(t) = K_p(x_{sp} - x_m) + K_d(V_{sp} - V_m) \tag{8}$$

6.2 Landing Autopilot

Generally, landing autopilot combines both hovering and tracking control in horizontal control loop with additional control to the vertical control loops in order to hover and land the quadrotor upon the target. It starts with a target searching and recovery task. This flight task is automatically activated when the target is not in the FOV or lost. When this happens, the vertical control loop, in coorperated with the horizontal control loop for hovering stabilization, increases the altitude by 10 *cm* every 1 *sec*. When the target is seen in the image, the target tracking and descend tasks will take over the autopilot automatically. The vertical control loop decreases its current altitude by 10 *cm* every 1 *sec* when the target is observed and tracked by the horizontal control loop with the position of the vehicle relative to the target not more than 5 *cm*. After the altitude is lowered to below 20 *cm*, the landing task is activated to land the quadrotor on the landing platform.

7 Real-Time Implementation

In this section, the proposed control strategy and method were implemented and tested in the indoor flight test. During the flight test, the important data were logged and the results will be discussed in this section. Generally, the real-time architecture of the vision-based autopilot system is presented in Fig. 15. The required measurements of the onboard sensors including video from the camera were transmitted through WIFI to the ground station. Then, the ground computer processed the images, performed computer vision techniques and state estimation, and lastly sent the commands for the outer loop control back to the onboard computer for carrying out the tasks of the autopilot mode.

7.1 Flight Test

A flight test has been conducted to evaluate the proposed automatic landing algorithm. For the purpose of challenging this autopilot, the quadrotor was placed at a distance away from the blob such that the blob was not inside the FOV after take-

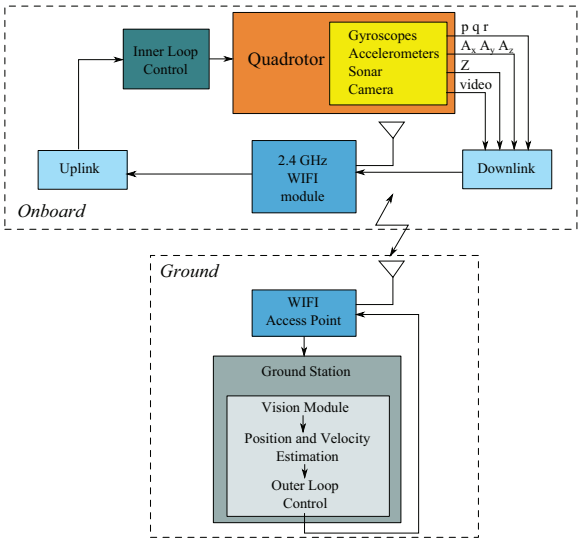

Fig. 15 Real-Time Architecture of the Vision-based Autopilot

off. Once the quadrotor took-off and the blob was not detected, the target searching and recovery task was automatically performed. It is clearly seen from Fig. 16 that the relative horizontal position was zero indicating that the blob was not inside the FOV. The vertical control loop performed the searching task by steadily increasing the altitude until the blob was detected while the horizontal control loop made sure that the horizontal velocity was controlled to zero. The real-time estimated horizontal velocities and measured vertical velocity are plotted in Fig. 17. Besides, the measured body attitude is shown in Fig. 18.

At the instant when the blob was detected, the tracking and descend task was automatically activated to perform continuous target tracking and descend. In Fig. 16, the quadrotor detected a distance away from the target at around 38 *sec* and reacted quickly by moving towards the target. After that, it slowly descended at a nearly constant rate when the estimated relative position tracked has an error below 5 *cm*. After the quadrotor moved close to the platform (below 20 *cm*), landing mode was automatically activated to land the quadrotor upon the platform. Therefore, it can conclude based on the results from the flight tests that the design and implementation of the fully autonomous landing system of the quadrotor using a single onboard camera has suceeded.

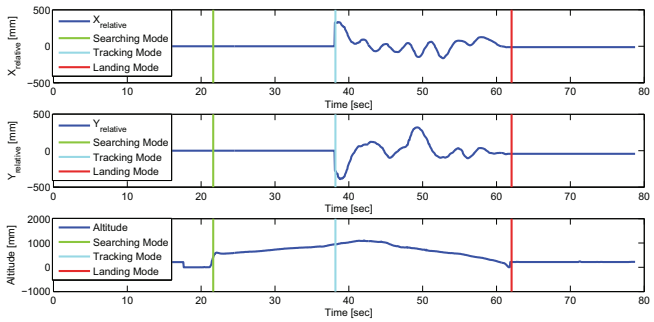

Fig. 16 Real-Time Estimated Relative Horizontal Position and Altitude Measurement during Autonomous Landing

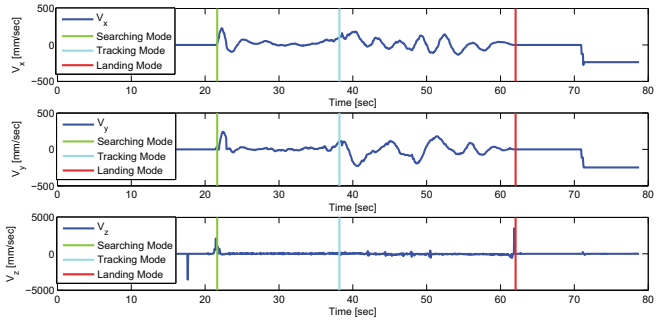

Fig. 17 Real-Time Estimated Body Horizontal Velocities and Vertical Velocity Measurement during Autonomous Landing

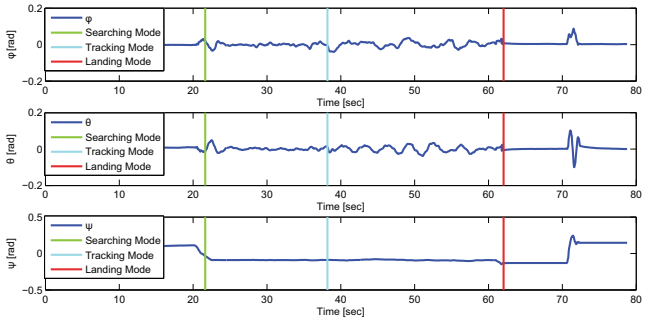

Fig. 18 Body Attitudes Measurement during Autonomous Landing

8 Conclusion

In this paper, an automatic landing system using visual servoing technique was presented. In order to achieve the goal of the project, a design solution for a compre-

hensive automatic landing algorithm comprising a new approach that combines two different computer vision techniques was proposed. A red blob tracking was implemented to obtain the position of the quadrotor UAV relative to the landing platform while a robust motion tracking using corner tracking was suggested working parallel with the target tracking to give additional state information to the controller when the target is not in the FOV of the camera. From the results obtained in 2-dimensional image, the position and velocity of the vehicle in physical units were estimated using a Kalman-based sensor fusion. The state estimation has been validated in a series of experiments using a CNC milling machine to show that the estimated state is accurate and precise. Finally, the control architecture for automatic landing system was formed with the classical PID controller. The flight tests proved that the proposed system succeeded.

References

1. Brown, D.: Close-range camera calibration. Photogrammetric engineering **37**(8), 855–866 (1971)
2. Chang, F., Chen, C., Lu, C.: A linear-time component-labeling algorithm using contour tracing technique. Computer Vision and Image Understanding **93**(2), 206–220 (2004)
3. Kendoul, F., Fantoni, I., Dherbomez, G.: Three Nested Kalman Filters-Based Algorithm for Real-Time Estimation of Optical Flow , UAV Motion and Obstacles Detection. In: Proceedings of the IEEE International Conference on Robotics and Automation, April, pp. 4746–4751. Rome, Italy (2007)
4. Lukas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the DARPA Image Understanding Workshop, pp. 121–130 (1981)
5. Nordberg, K., Doherty, P., Farneback, G., Forssen, P., Granlund, G., Moe, A., Wiklund, J.: Vision for a UAV helicopter. International Conference on Intelligent Robots and Systems (IROS'02) (2002)
6. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, vol. 2, pp. 1508–1515. IEEE (2005)
7. Saripalli, S., Montgomery, J.F., Sukhatme, G.S.: Visually Guided Landing of an Unmanned Aerial Vehicle. IEEE Transactions on Robotics and Automation **19**(3), 371–380 (2003)
8. Wu, A., Johnson, E., Proctor, A.: Vision-aided inertial navigation for flight control. JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION **2**(September), 348–360 (2005)
9. Zhang, Z.: A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(11), 1330–1334 (2000)