Intelligent Automation And Soft Computing, 2020
Vol.26,no.3,489–500
DOI: 10.32604/iasc.2020.013925

The Design of a TLD and Fuzzy-PID Controller based on the Autonomous Tracking System for Quadrotor Drones

Pi-Yun Chen and Guan-Yu Chen*

Department of Electrical Engineering, National Chin-Yi University of Technology, Taiwan

ABSTRACT
The objective of this paper is to design a new Quadrotor Autonomous Following System, and the main three contents are as follows: Object tracking, quadrotor attitude determination and the controller. The image tracking portion performs object detection and keeps tracking by way of the Tracking-Learning-Detection (TLD), and gets the information of the target motion estimation positions. The attitude determination of the Quadrotor has adopted the Inertial Navigation System and sensors of the accelerometer, gyroscope and electronic compass, etc. for retrieving the information. The Kalman filter is also utilized for estimating the current values in order to reduce external interference, improve the accuracy, and obtain the current posture of the Quadrotor. As for the control method, the mathematical modeling on the Quadrotor is performed first so that the Quadrotor can obtain the correct posture through the three-axis compensations, and then the fuzzy-PID controller may obtain the three-axis following angles for output to execute the commands and track the users. Finally the feasibility of this method is verified by using the flying software simulation and tangible flight experiments.

KEY WORDS: Tracking-learning-detection (TLD), quadrotor, Kalman filter, fuzzy-PID.

1 INTRODUCTION

A search of the literature shows that over the past few years (Low & Wang, 2008; Liu & Prior, 2015), robot positioning is most commonly implemented by triangulation using GPS or Wi-Fi receivers. Positioning-enabled mobile devices, equipped by the user, also facilitate the trajectory tracking by the robot. The Euclidean distance between the user and also the robot can be computed for tracking if they are close enough together. Besides the positioning, another method of tracking uses a variety of sensors. Well-developed RGD-D sensors (Vetrella, Savvaris & Fasano, 2015) are frequently used in robot tracking. The images of color and depth offer information such as; current distance, posture and body frame, which is used for tracking. This greatly reduces the failure rate and allows more stable tracking.

Although the GPS positioning is used mostly by commercial available quadrotor drones, it has error sources due to environmental interference and safety concerns. The development of the GPS independent drones has become the center of research in recent years. Unlike an RGB-D sensor system a dual-camera stereo vision system, which uses two cameras to derive and synthesize the depth of the pixels, without using infrared depth sensing, and this greatly reduces the size and power needed by the sensors. A single camera has been used for object tracking in some studies (Engel, Sturm & Cremers, 2012; Dang, Pham & Pham, 2013; Valenci & Kim, 2018). The system has simple hardware architecture but a more complicated tracking algorithm, and delivers good tracking performance very suitable for the quadrotor drones. Single camera image tracking uses a foreground recognition method based on a hue shift. Images are first converted from the original RGB format to the HSV color space, with the hue representing the color feature distribution of the object to be tracked. The thresholds of saturation and value can be adjusted to suit ambient lighting and accommodate light changes.

In this study a tracking method based on a Tracking-Learning-Detection (TLD) algorithm proposed by Kalal et al. (Kalal, Mikolajczyk & Matas, 2012) has been used. It combines the tracking, detection and an online learning mechanism. A description of how the machine vision is introduced

CONTACT Pi-Yun Chen chenby@ncut.edu.tw

into a quadrotor drone controller is also presented. The hardware equipment and system architecture will be covered in Section 2, which gives a clear description of the operational flow. Further details about this method are given in Section 3. The methods for the quadrotor posture estimation and control are introduced in Sections 4 and 5. Experimental results are presented in Section 6.

2 HARDWARE AND SYSTEM ENVIRONMENT

THE hardware architecture of the quadrotor drone autonomous tracking system is comprised of computational equipment on the ground-side, and communication side and the flight control on the drone itself. The posture control and image processing are handled by both the communications side on the drone and the ground, as shown in Figure 1. The flight controller is mainly responsible for computing the posture and interpreting throttle commands. The current posture of the drone is received and transmitted to the ground side. The ground side has a set of bi-directional 2.4 GHz radio frequency modules and a smartphone. These deal mainly with complicated image computation but also compensate for the lack of sufficient flight controller computational power.

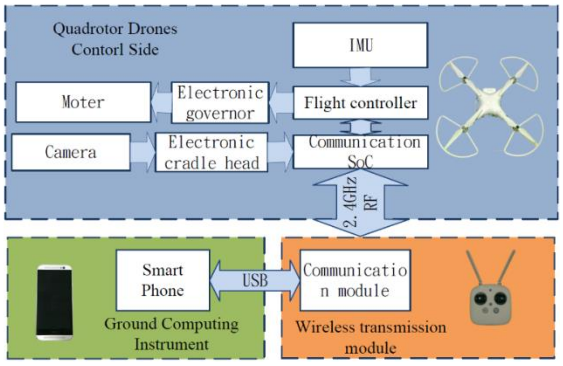


Figure 1. **Hardware Architecture.**

The hardware used is the Phantom 3 Advanced quadrotor drone developed by DJI. The autonomous tracking system was implemented by integrating the existing quadrotor drone system and the algorithms represented in this paper. To integrate resources among different platforms, C++ coded image processing functions are incorporated into the Android system. Figure 2 shows a detailed flow chart of the system.

3 IMAGE TRACKING

FIGURE 3 shows the image tracking flow chart. To ensure the display of the real-time screen input, the latest image is loaded from the buffer only at the start of a new cycle, the others are being discarded. However, this method has limitations in the optical flow applications. One condition that needs to be satisfied in the optical flow equation is that the displacement of the object in nearby frames must be small. The optical flow computation might not produce good results if too many images are discarded. Therefore, it is necessary to compress the images as much as possible. In this study the TLD algorithm was employed for tracking. It is suitable for long term tracking of the object and is not subject to the influence of dynamic backgrounds. It also has fewer limitations than other algorithms when used in the outdoor image tracking.

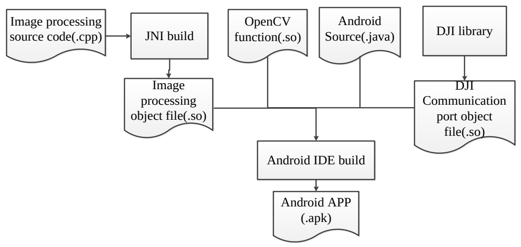


Image processing source code(.cpp)
JNI build
OpenCV function(.so)
Android Source(.java)
DJI library
Image processing object file(.so)
DJI Communication port object file(.so)
Android IDE build
Android APP (.apk)

Figure 2. **The Flowchart of the Software Processing Established on the Ground Station.**

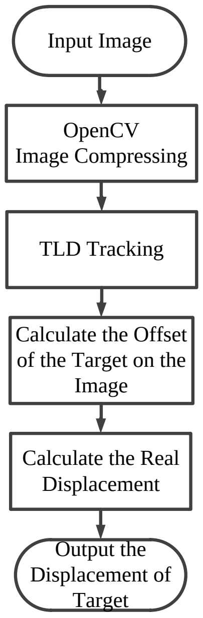

Input Image
OpenCV Image Compressing
TLD Tracking
Calculate the Offset of the Target on the Image
Calculate the Real Displacement
Output the Displacement of Target

Figure 3. **The Flowchart of the Image Tracking.**

3.1 *Tracking-Learning-Detection (TLD)*

The TLD algorithm includes three parts that work simultaneously, see Figure 4. These are tracking, detection and learning. The tracker estimates the moving direction of the object. When the tracker loses the position of the object, the detector starts. The Learning evaluates the tracking results and online learning allows better tracking. The information of the object's dynamic estimation includes the next possible position and the moving speed of the object in the picture, which will be extracted as control parameters. A frame, generally known as the Region of Interest (ROI), will be initiated at the start of the TLD algorithm. This frame, including its position, will be updated regularly by the TLD algorithm during the tracking process.

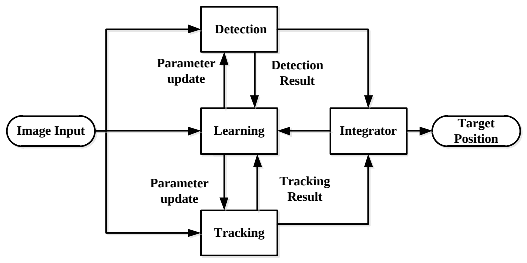


Figure 4. TLD Architecture.

3.1.1 Optical Flow Learning

For tracking, the TLD algorithm uses a method that is an improvement on the Pyramid, Lucas-Kanade Optical Flow method (Bouguet, 2001). The basic principle of L-K optical flow is the detection of the positional change of each pixel between the nearby frames, by the differential method, to obtain the direction and velocity of optical flow. Assume pixel Q has a displacement between two nearby frames, and all pixels a_n in the neighborhood space also have the same displacements, then the optical equation is assumed to hold. The intensities of the pixel in the three-dimensional coordinates x, y and time t are represented as I_x, I_y and I_t, respectively. The optical flow rates of V_x, V_y of pixel Q and adjacent points a_n should satisfy;

$$\mathrm{I}_x(a_n)V_x + \mathrm{I}_y(a_n)V_y = -\mathrm{I}_t(a_n) \qquad (1)$$

Since there are several pixels in this hypothetical area, a set of simultaneous equations can be solved.

3.1.2 P-N Learning

The TLD method uses semi-supervised P-N learning, in which the detector's errors are estimated by the P-N experts. The incorrectly classified positives and negatives are given separately to the P-expert and N-expert for analysis. The P-expert updates false negatives to positives and adds them to the training sample set. Similarly, the N-expert updates false positives to negatives and adds them to the training sample set. Figure 5 shows a diagram of the P-N learning architecture.

3.1.3 The Detector

The tracker proposed in the TLD method is a cascaded classifier, see Figure 6. It is structured as three sub-classifiers; patch variance, ensemble classifier, and nearest neighbor classifier. Since the nearest neighbor classifier needs more computation resources, it is not suitable for the application and to all of the patches. Therefore, the process is divided into three sequential stages. The patches are first filtered by the patch variance and the ensemble classifier. The patches that meet the criteria are fed into the nearest neighbor classifier.

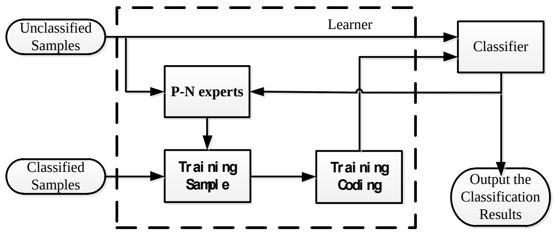


Figure 5. P-N Learning Architecture.

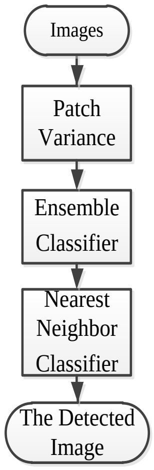


Figure 6. The Flowchart of the Cascade Classifier.

The purpose of the nearest neighbor classifier is to train the object, model M, which is the set used to represent the object and its data in the surrounding environment. It is a set of patches that contain positives and P_n^+ negatives P_n^-. P^+ and P^- represents the patches of the object and the background, respectively. In this method, the spatial similarity of two tracking frames is measured with overlap, which is defined as the summation of the intersection and union of the two tracking frames. The shape of the object is represented as patch p. The similarity between the two pictures P_i and P_j can be represented as shown in Equation (2).

$$S(P_i, P_j) = 0.5(\mathrm{NCC}(P_i, P_{j+1}) + 1), \qquad (2)$$

where the NCC is the normalized correlation coefficient and for patch p and the object model M, several quantitative indicators are defined in the P-N learning method:

1. **Similarity with the positive nearest neighbor:**
$$S^{+}(p,M)=\max_{p_i^{+}\in M} S(p,P_i^{+}) \tag{3}$$
2. **Similarity with the negative nearest neighbor:**
$$S^{-}(p,M)=\max_{p_i^{-}\in M} S(p,P_i^{-}) \tag{4}$$
3. **Similarity with the positive nearest neighbor considering 50% earliest positive patches:**
$$S_{50\%}{}^{+}(p,M)=\max_{p_i\in M\wedge i<m/2} S(p,P_i^{+}) \tag{5}$$
4. **Relative similarity:**
$$S^{r}=\frac{S^{+}}{S^{+}+S^{-}} \tag{6}$$
5. **Conservative similarity:**
$$S^{c}=\frac{S_{50\%}{}^{+}}{S_{50\%}{}^{+}+S^{-}} \tag{7}$$

If $S^{c}(p,M)>\theta_{NN}$, where the threshold $\theta_{NN}=0.6$ and its value is the empirical value, then patch p is the Positive samples of final output by detection.

3.2 *Inverse Perspective Mapping (IPM)*

When the position of the object on a two-dimensional plane has been obtained, it must be mapped onto a three-dimensional space to calculate its actual position and distance. This has been achieved in this study using the inverse perspective mapping method (Muad, et al, 2004). The principle of the IPM is shown in Figure 7, where $[x_c, y_c, z_c]$ represents the camera's coordinate system, $[x_g, y_g, z_g]$ is the coordinate system of the earth, f is the camera's focal length, θ_c is the camera's forward leaning angle, φ_c is the deviation angle of the image plane, H_c is the height of the camera above the ground surface, o is the obstacle point on the ground surface, o' is the obstacle when o moves a distance d in the direction of x_g, g is the point on the ground surface that has a distance of x from the origin of the earth coordinate system, and g' is the point where point g moves distance d along the direction of x_g.

The coordinates of p, which represent the coordinates of the pixel on the original image, can be obtained from the picture. u represents the position in the world coordinate system. The relation between them is represented as:

$$\begin{bmatrix}u_x\\u_y\\u_z\end{bmatrix}=\begin{bmatrix}0\\0\\H_c\end{bmatrix}+\lambda\begin{bmatrix}p_x\cos\theta_c-f\cos\varphi_c\sin\theta_c+p_z\sin\varphi\sin\theta_c\\p_x\sin\theta_c+f\cos\varphi_c\cos\theta_c-p_z\sin\varphi\cos\theta_c\\f\sin\varphi_c-p_z\cos\varphi_c\end{bmatrix} \tag{8}$$

where λ is the mapping coefficient in the world coordinate system, λ can be obtained when the ground coordinate u_z is 0. Ground coordinates u_x and u_y, is the distance and width between the object and camera, and in the world coordinate system can be obtained as seen in Equations (9) and (10).

$$u_x=\frac{H_c p_x\cos\theta_c+(p_z\sin\varphi-f\cos\varphi_c)(H_c\sin\theta_c)}{p_z\cos\varphi_c+f\sin\varphi_c} \tag{9}$$

$$u_y=\frac{H_c p_x\sin\theta_c-(p_z\sin\varphi_c-f\cos\varphi_c)(H_c\cos\theta_c)}{p_z\cos\varphi_c+f\sin\varphi_c} \tag{10}$$

Since φ_c is the deviation angle of the image plane, which is normally set to 0, the equations are rewritten as Equations (11) and (12).

$$u_x=\frac{H_c p_x}{p_z\cos\varphi_c+f\sin\varphi_c} \tag{11}$$

$$u_y=\frac{H_c(f\cos\varphi_c-p_z f\sin\varphi_c)}{p_z\cos\varphi_c+f\sin\varphi_c} \tag{12}$$

The object's position in the world coordinate system has been obtained and the next question is how to control the drone to perform the object tracking? Related methods will be covered in the following section.

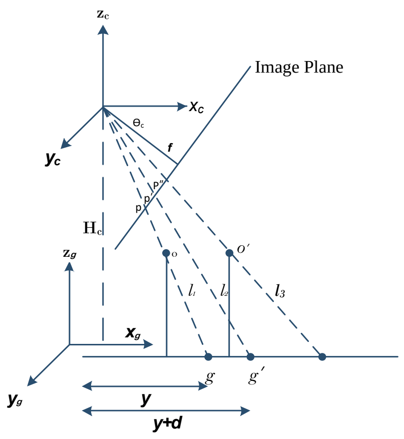


Figure 7. **The Principle of the IPM.**

4 POSTURE MEASUREMENT AND MODELING

THE posture control flow chart is shown in Figure 8. This section deals with the methods of posture measurement and coordinate transformation. The inertial measurement units, including gyroscope and accelerometer are used to generate information on six axes. After the Kalman filter has been applied, quaternions are used to calculate the current position of the drone.

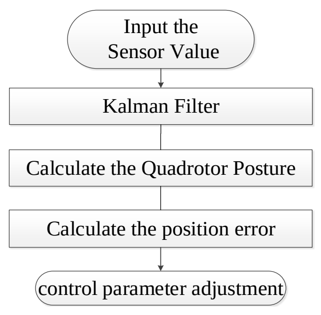

Input the Sensor Value
Kalman Filter
Calculate the Quadrotor Posture
Calculate the position error
control parameter adjustment

Figure 8. **The Flow Chart of the Posture Control Parameter Adjustment.**

4.1 The Kalman Filter

The Kalman filter (Ludeman, 2003) is recursive and widely used in communications and control systems. It is capable of producing an accurate estimate of a current state even with noisy measurements. The estimate of the previous state and current measurement are the only input required. The development of the Kalman filter is based on linear algebra and the Hidden Markov Model. The basic Kalman filter model is shown in Equation (13).

$$x_k = \boldsymbol{F}_k x_{k-1} + \boldsymbol{B}_k u_k + w_k \tag{13}$$

where x_k , is the estimate of the current state, x_{k-1} is the estimate of the previous state, $\boldsymbol{F}_k$ is the state transition matrix, u_k is the control vector, $\boldsymbol{B}_k$ is the Control-Input Model and w_k is the input noise. Assume the average of w_k is zero, the covariance matrix is $\boldsymbol{Q}_k$ and is under the multivariate normal distribution. Therefore,

$$w_k \sim N(0, \boldsymbol{Q}_k) \tag{14}$$

at the current time is k, the measurement state is z_k and is obtained based on the actual state of x_k.

$$_k = \boldsymbol{H}_k x_k + v_k \tag{15}$$

where $\boldsymbol{H}_k$ is the measurement model, which maps the actual state space into the measurement space, v_k is the measurement noise, whose average is assumed to be zero? The covariance matrix is $\boldsymbol{R}_k$ and is under normal distribution. Therefore,

$$v_k \sim N(0, \boldsymbol{R}_k) \tag{16}$$

Since the actual state of x_k is unknown and can only be observed through the measurement state ofz_k, the Kalman filter can be represented by two variables; $\hat{x}_{k|k}$ (Posteriori State Estimate) and $P_{k|k}$ (Posteriori Error Covariance Matrix). From there, it then estimates the current state through the two phases of predict and update. An estimate of the current state $\hat{x}_{k|k-1}$ is obtained from the estimate of the previous state, as shown in Equation (17). The current error covariance matrix $P_{k|k-1}$ is obtained from the previous error covariance matrix, as shown in Equation (18). Therefore, $\hat{x}_{k|k-1}$ is also referred to as the Priori State Estimate and $P_{k|k-1}$ as the Priori Estimate Covariance.

$$\hat{x}_{k|k-1} = \boldsymbol{F}_k \hat{x}_{k-1|k-1} + \boldsymbol{B}_k u_k \tag{17}$$

$$P_{k|k-1} = \boldsymbol{F}_k P_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k \tag{18}$$

The purpose of the updates is to find the K_k Optimal Kalman Gain in order to update the Posteriori State Estimate and Posteriori Estimate Covariance matrix, see (19) through (23).

$$\tilde{y}_k = z_k - \boldsymbol{H}_k \hat{x}_{k|k-1} \tag{19}$$

$$S_k = \boldsymbol{H}_k P_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{20}$$

$$K_k = P_{k|k-1} \boldsymbol{H}_k^T S_k^{-1} \tag{21}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \boldsymbol{K}_k \tilde{y}_k \tag{22}$$

$$P_{k|k} = (I - K_k \boldsymbol{H}_k) P_{k|k-1} \tag{23}$$

where $\tilde{y}_k$ is the Measurement Residual and S_k is the Residual Covariance. If initial states of $\hat{x}_{(0|0)}$ and $P_{(0|0)}$ are defined correctly, then all estimated errors will be zero and the covariance matrix will accurately reflect the estimated covariance, see (24) through (28).

$$E[x_k - \hat{x}_{k|k}] = E[x_k - \hat{x}_{k|k-1}] = 0 \tag{24}$$

$$E[\tilde{y}_k] = 0 \tag{25}$$

$$P_{k|k} = cov(x_k - \hat{x}_{k|k}) \tag{26}$$

$$P_{k|k-1} = cov(x_k - \hat{x}_{k|k-1}) \tag{27}$$

$$S_k = cov(\tilde{y}_k) \tag{28}$$

where $E[a]$ is the expected value of a and $cov(a) = E(aa^T)^T$.

4.2 Definition of Coordinate Systems

Before computing postures, the ground coordinate system and body-fixed coordinate system must be defined. The ground coordinate system is made up of three orthogonal axes; x_g, y_g and z_g. The body-fixed coordinate system is made up of x_b, y_b and z_b. As shown in Figure 9 ϕ, θ and ψ are the Euler angles along the pitch, roll and yaw axes. The detailed notations are shown in Table 1.

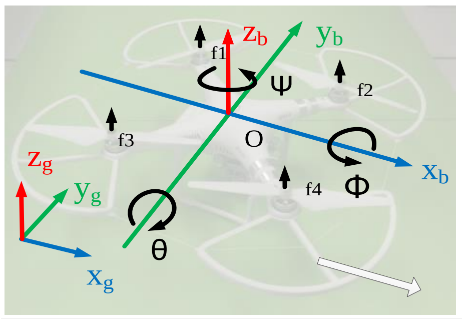


Figure 9. The System Coordinates of the Quadrotor.

The detailed definitions of ϕ , θ and ψ are summarized in Table 1 below:

Table 1. Defined Posture Angle.

ϕ	Roll	Rotate x_b	$-\pi \le \phi \le \pi$	Counter clockwise
θ	Pitch	Rotate y_b	$-\pi \le \theta \le \pi$	

ψ	Yaw	Rotate z_b	$-\pi \le \psi \le \pi$	Rotation

The first step of the coordinate transformation (Castillo, Dzul & Lozano, 2004) is to find the rotation matrix for all the Euler angles. As shown in the equation below, $R_{x_b}^{x_g}$, $R_{y_b}^{y_g}$ and $R_{z_b}^{z_g}$ are the rotations generated by ϕ, θ and ψ, respectively. The complete rotation matrix can be obtained after the completion of rotations along the three axes, shown in Equation (29).

$$R_b^g = \begin{bmatrix} cos\theta cos\psi & sin\psi sin\theta & -sin\phi \\ cos\psi sin\theta sin\phi - sin\psi cos\phi & sin\psi sin\theta sin\phi + cos\psi cos\phi & cos\theta sin\phi \\ cos\psi S_\theta cos\phi + sin\psi cos\phi & sin\psi sin\theta cos\phi - cos\psi sin\phi & cos\theta cos\phi \end{bmatrix} \tag{29}$$

4.3 The Conversion of Quaternions

Quaterions (Diebel, 2006) are generally used in solving the problems of singularities associated with the Euler angles and Gimbal lock. Quaternions are used here, because it is more convenient and faster to retrieve quaternions during frequent posture changes. A Quaternion is a four-dimensional complex number, which has four parts; q_0, q_1, q_2 and q_3, where ${q_0}^2+{q_1}^2+{q_2}^2+{q_3}^2=1$. Like the rotations of the Euler angles, assume that after α, β and r times of rotations, $(\cos^{-1}\alpha, \cos^{-1}\beta, \cos^{-1}r)$ is equivalent to the rotation of μ. The quaternion can be defined as:

$$q_0 = cos\frac{\mu}{2}, \quad q_1 = \alpha sin\frac{\mu}{2}$$
$$q_2 = \beta sin\frac{\mu}{2}, \quad q_3 = rsin\frac{\mu}{2} \tag{30}$$

It can be represented using the Euler angles in Equation (31).

$$\begin{cases} sin\theta = 2(q_0q_2 - q_3q_1) \\ tan\psi = \dfrac{2(q_0q_3 + q_1q_2)}{{q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2} \\ tan\phi = \dfrac{2(q_0q_1 + q_2q_3)}{{q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2} \end{cases} \tag{31}$$

The three-axis outputs of the gyroscope are; g_x, g_y and g_z. Therefore, the quaternion is updated in Equation (32):

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \tag{32}$$

Assume the initial gyroscope values are; $g_x = g_y = g_z = 0$. Initializations of quaternion are; $q_1 = q_2 = -q_3 = 0$, with $q_0 = 1$ only. The posture of the drone can be updated continuously by the quaternion iterations.

5 QUADROTOR DRONE CONTROL

AS shown in Figure 10, the main function of the control system takes place in two stages: The first stage involves the stabilization of the posture, and the second stage is position control, through which autonomous tracking can be achieved.

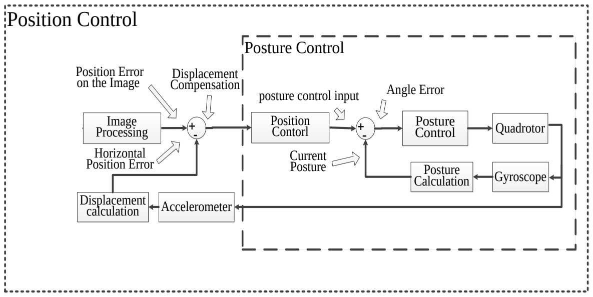


Figure 10. The Principle of the Quadrotor Tracking.

The purpose of the posture control in the first stage is to keep the drone's body as stable as possible by compensating the roll angle of ϕ and the pitch angle of θ so that the drone can be maintained in a horizontal attitude above the ground. The control of yaw angle ψ is mainly achieved through compensation, in other words, the introduction of the position difference of the object. Angle errors can be obtained by integrating the angular velocities measured by the gyroscope.

5.1 Position Control and Object Tracking

The second stage is the drone posture control. The error of each parameter is represented by e in the following, including situations where the drone moves at a fixed height, along the surface and during the user tracking. After this it is only necessary to handle the horizontal movement. The compensation of ex_b, and ey_b is carried out with the displacements obtained by the double integration of the readings of the accelerometer. Although some errors do occur when the drone hovers for a long time, tracking is not affected in the short term. Tracking errors are mainly caused by delays in image processing and the object's position deviation computed from the image tracking is regarded as part of the error for which the controller can compensate. It is intuitive to carry out the position control by changing yaw angle ψ and roll angle ϕ in the horizontally moving drone. However, if image tracking errors of ex_c and ey_c, and posture stability errors of ex_b and ey_b are introduced at the same time, it will be harder to adjust the controller parameters and it will not be so easy for the controller to achieve stability. Therefore, the compensation of yaw angle ψ is carried out first, before the roll angle ϕ, then the compensation of the relative position of x_b follows.

In addition to a concern for stability, another reason for the drone to change yaw angle ψ instead of roll angle ϕ is the existence of the obstacles to the sides. As shown in Figure 11, if the quadrotor vehicle tracks the user by compensating for ex_b and by changing the roll angle ϕ, it is likely to crash into an obstacle.

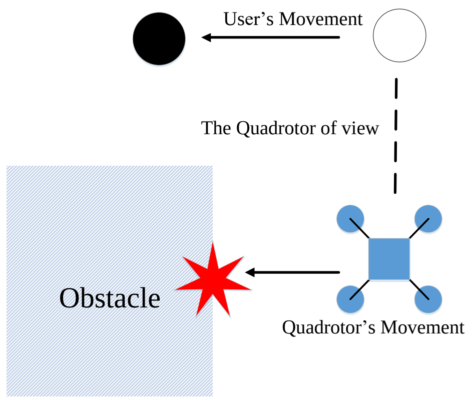


Figure 11. **User Tracking by Through Roll Angle φ.**

If yaw angle ψ is changed first, the obstacle might cause the drone to lose the object and enter the hovering state, as shown in Figure 12. This is not an ideal result. However, it is better than a crash where no provision for the handling of obstacles has been made.

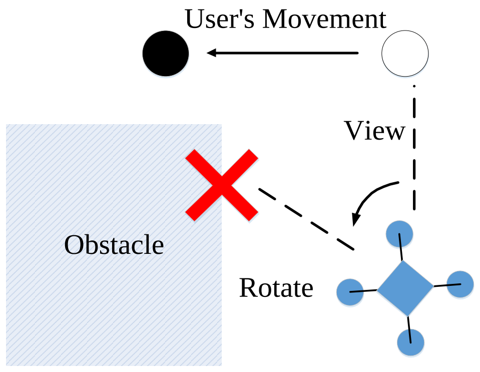


Figure 12. **User Tracking by Through Yaw Angle ψ.**

5.2 *Fuzzy-PID Controller*

The advantages of a PID controller are; easy implementation, a low parameter number and stability. In cases where a complete mathematical model is not available, a PID controller is a fairly practical choice. However, when dealing with the dynamically changing drone postures, a PID controller with fixed parameters may not be good enough. A better solution was the Fuzzy-PID controller used in this study. The PID parameters are dynamically adjusted through fuzzy control to adapt to a more complicated system and give better control of the drone posture. (Choi, 2015; Sato, 1995; Huang & Luo, 2018).

5.2.1 Fuzzy Control

Fuzzy control is a three part process. The input fuzzy set first has to be determined and then the member functions of the input and output must be defined. Multiple member functions are needed if there is more than one input or output. In this study every PID controller used error e and error change ce as input variables, while the output variable are k_p, k_i and k_d Arranging the elements in ascending order gives, the fuzzy set NB (Negative Big), NM (Negative Medium), NS (Negative Small), ZO (Zero), PS (Positive Small), PM (Positive Medium), and PB (Positive Big).

Figure 13 shows the member functions of the two input variables, error e and error change de. The normalized input values lie between -1 and 1, which is the domain. Any input or output values in applications beyond this domain must first be normalized. Exceptional handling is needed for the values outside the domain. Amplitude limitation is normally applied so that the boundaries will be in the shape of a trapezoid. Membership function is mainly used to describe the fuzzy input or output set.

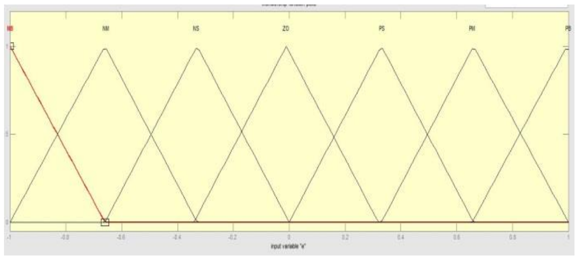


Figure 13. **The Member Functions of the Two Input Variables**

In addition to the member function, we also need to define the fuzzy rules. Each output has its fuzzy rules corresponding to the input variables, which are used to determine the set to which the output belongs and the gradient of the membership. Experience gained in adjusting the PID controller (the PID controller was discussed in the last section) allowed the development of the following set of rules. When error e is large and error rate de is small, a larger k_p will be chosen to speed up the compensation. When e and error rate de are both small, a larger k_i is chosen to compensate the steady state error as much as possible. When e is small but the error rate de is large, a relatively large k_d is chosen to suppress the error fluctuation. Once the rules are determined, individual fuzzy rule bases can be built for each output. The real-time adjustment of the parameters can also be achieved through fuzzy control. The Fuzzy rule bases used in this paper are shown in Tables 2 - 4.

Table 2. k_p **Fuzzy Rule Base.**

k_p		*e*						
		NB	NM	NS	ZO	PS	PM	PB
de	NB	NB	NB	NB	NM	NM	PS	PS
	NM	NB	NB	NM	NM	NS	PS	PS
	NS	NB	NM	NM	NS	ZO	PS	PM
	ZO	NM	NM	NS	ZO	PS	PM	PM
	PS	NM	NS	ZO	PS	PM	PM	PB
	PM	NS	NS	PS	PM	PM	PB	PB
	PB	NS	NS	PM	PM	PB	PB	PB

Table 3. k_i **Fuzzy Rule Base.**

k_i		*e*						
		NB	NM	NS	ZO	PS	PM	PB
de	NB	NB	NB	NM	NM	NM	NS	ZO
	NM	NB	NM	NM	NM	NS	ZO	PS
	NS	NM	NM	NM	NS	ZO	PS	PM
	ZO	NM	NM	NS	ZO	PS	PM	PM
	PS	NM	NS	ZO	PS	PM	PM	PM
	PM	NS	ZO	PS	PM	PM	PM	PB
	PB	ZO	PS	PM	PM	PM	PB	PB

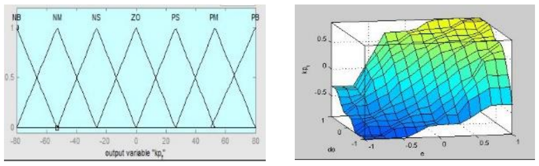
Table 4. Fuzzy Rule Base k_d.

k_d		*e*						
		NB	NM	NS	ZO	PS	PM	PB
de	NB	NS	NM	NB	NB	NB	NM	NS
	NM	NS	NS	NM	NM	NM	NS	NS
	NS	ZO	NS	NS	NS	NS	NS	ZO
	ZO	ZO	ZO	ZO	ZO	ZO	ZO	ZO
	PS	ZO	PS	PS	PS	PS	PS	ZO
	PM	PS	PS	PM	PM	PM	PS	PS
	PB	PS	PM	PB	PB	PB	PM	PS

Lastly, the output is obtained by the defuzzification of the results with reference to the member function of the output parameters. For the defuzzification, the Center Average Defuzzifier, as shown in Equation (33), was used where B' represents the fussy set, h the height of B', p_l the y value of the original center point of B' and l the l-th gradient of the membership.

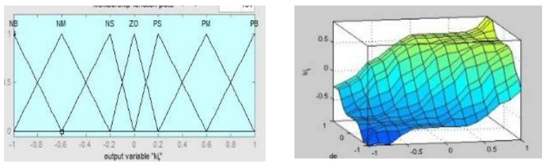
$$y^* = \frac{\sum_{l=1}^{m} p_l(B_l')}{\sum_{l=1}^{m} h(B_l')} \tag{33}$$

The domains of the member functions for the input and output are all between -1 and 1 by default. The parameters are adjusted gradually during the simulation and testing. In the first stage, only the fuzzy rules and input-output plots are established. The relations between the input and output are defined by different curves. Each triangle in the input membership function has the same area and height so that the output curves can be adjusted easier. Figures 14 - 16 are the output membership functions. 1:1 direct mapping was used to reduce fluctuation of the compensation ratio. For k_i, NM and PM are expanded to make it easier for the fuzzy controller to recognize the nuanced differences between the middle values. With the same principle, NB and PB are also expanded to achieve more precise suppression of error fluctuation. Figures 5.5 - 5.7 are the membership functions k_p, k_i and k_d, and their corresponding response curves versus the output. X axis is e and Y axis is de. The vertical axis in the 3D plot represents the Individual output.



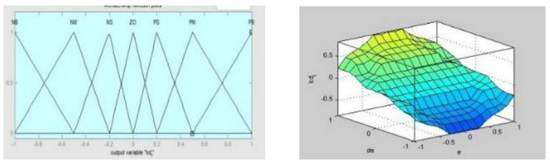
(a) Membership Function (b) The Relation between the Input and Output

Figure 14. k_p **Membership Function and the Relation between the Input and Output.**



(a) Membership Function (b) The Relation between the Input and Output

Figure 15. k_i **Membership Function and the Relation between the Input and Output.**



(a) Membership Function (b) The Relation between the Input and Output

Figure 16. k_d **Membership Function and the Relation between the Input and Output.**

5.2.2 Fuzzy-PID Architecture

Figure 17 shows the prototype fuzzy controller used in this study before the parameter adjustment. Blocks A and B in the figure are the fuzzy controller's gains for the input and output. They are used to adjust the input and output limits of the Fuzzy-PID controller. If the gains are set too high, the gradient of membership will always fall in the NS or PS interval. With gains that are too small, the gradient of the membership might always fall in the NB or PS interval. Neither condition is ideal for the controller. Appropriate adjustment of parameters is necessary during the experiment to obtain better results.

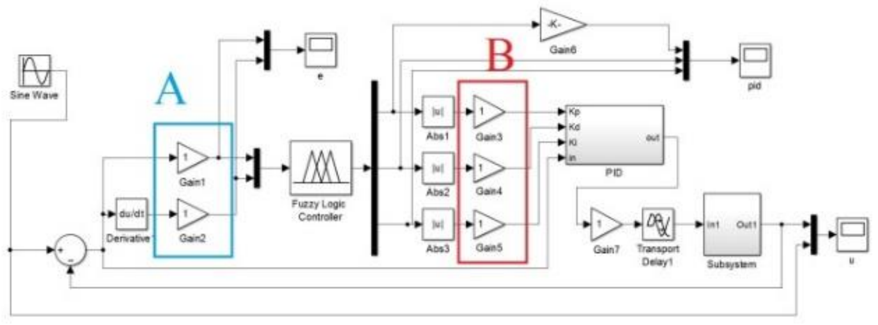


Figure 17. **The Prototype of the Fuzzy-PID Controller.**

6 EXPERIMENTAL RESULTS

THE algorithms used in this study have been described in the previous sections. The experimental results and final parameters used will be represented here and the limitations of the system operation will also be described.

6.1 Image Tracking Experiment

The TLD and IPM based methods were used for tracking in this experiment. A known reference point is needed for the IPM initialization. The best initial point is set in Figure 18 and a schematic diagram is shown in Figure 19.

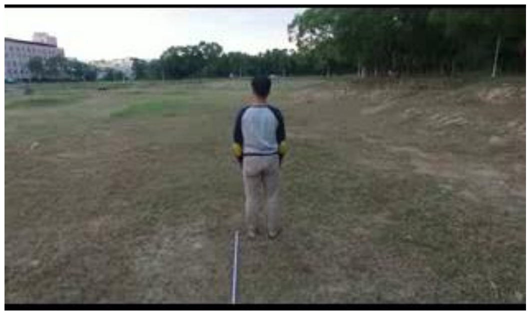


Figure 18. **Setting an Initial Position.**

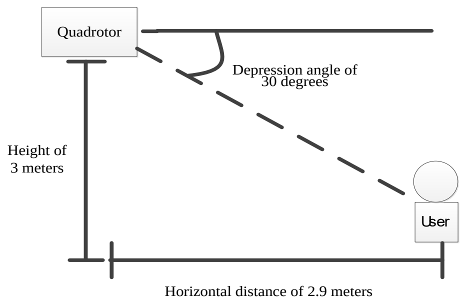

Quadrotor
Depression angle of 30 degrees
Height of 3 meters
User
Horizontal distance of 2.9 meters

Figure 19. **The IPM Side View of the Initial Position.**

If the user initializes the image at the relative position shown in Figure 19, the width of the projected picture and actual width are approximately 320 pixels and 3.2 meters, respectively. The proportion between the picture and the ROI size can be calculated easily. With this proportion of the original ROI size, the object, whether moving closer or farther away, will be totally reflected in the TLD tracking frames.

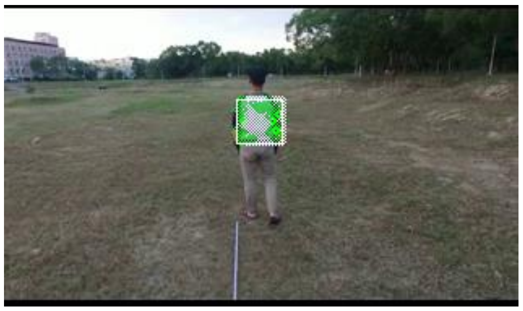


Figure 20. **Initial TLD Tracking.**

Figure 20 shows the initialization state of the TLD algorithm. The object to be tracked is marked manually. Figure 21 is the result of continuous tracking after the object moves for a certain time. Figure 22 shows that the object is not lost even if the drone moves and the background are updated.

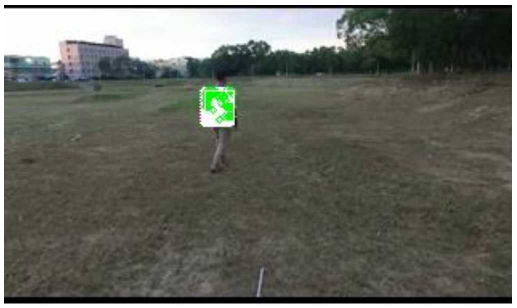
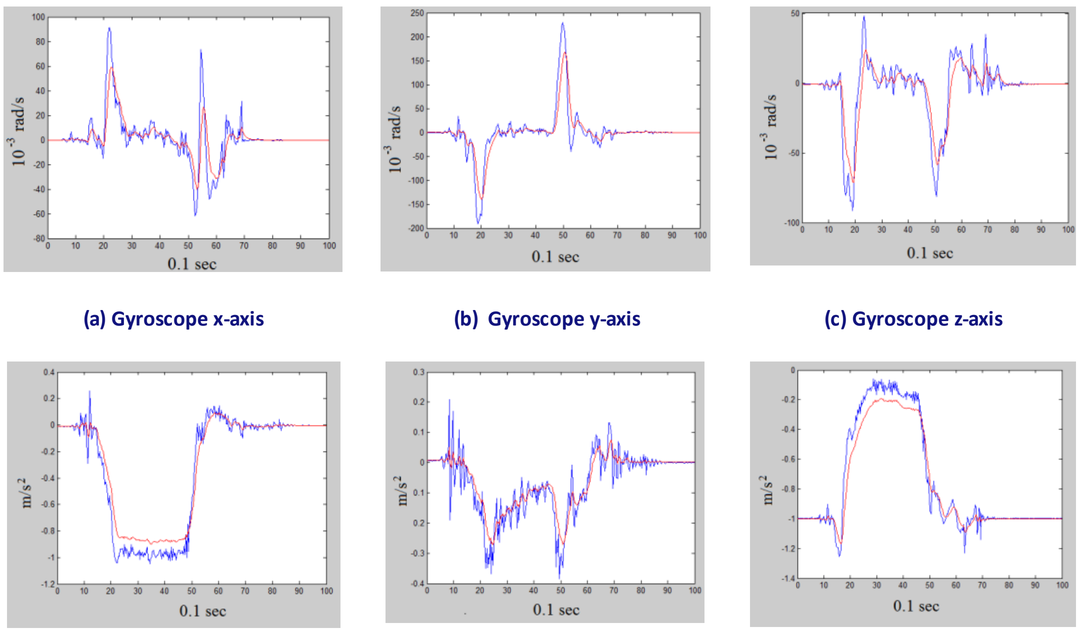


Figure 21. **Continuous Tracking.**

Figure 22. **Background Update.**

6.2 *Test of the Kalman Filter*

In this test, the quadrotor is powered on, and when the drone is hovering, each of its two Euler angles is changed by 90 degrees. The original 6-axis values and the output values of the Kalman filter are all taken, recorded and plotted using MATLAB, as shown in Figure 23 (a-f).



(a) Gyroscope x-axis **(b) Gyroscope y-axis** **(c) Gyroscope z-axis**

(d) Accelerometer x-axis **(e) Accelerometer y-axis** **(f) Accelerometer z-axis**

Figure 23. **The Results of the Kalman Filter.**

In Figure 23 (a-f), the Blue line represents the original reading of the sensor, and the Red line is the output of the Kalman filter. It can be seen that most of the ambient noise is filtered out, and the actual angular velocity and velocity change are retained. The design used in this study is not intended for stunt quadrotor drones. The pitch angle of the drone used in this study is normally smaller than 30 degrees and in practice rarely reaches that angle. Therefore, sudden and large angle changes have not been taken into consideration. Current filter performance was satisfactory under these conditions.

6.3 *Fuzzy-PID Simulation*

This is a MATLAB simulation of the Fuzzy-PID controlled sine wave tracking. Figure 24 shows the Simulink simulation graph. The input signal is a sine wave (Red) with amplitude 1 and 1.5 radian/sec frequency. The output displayed is Yellow. For parameter choices, the gains for P, I and D is 35, 25 and 10, respectively. The gains here are not the same as k_p, k_i and k_d, and are used to change the output range of the fuzzy PID. The k_p, k_i and k_d outputs of the fuzzy PID are variables and can adapt to large-scale changes of the angular velocity.

Figure 25 shows the input signals of the fuzzy controller. They are; k_p (Red), k_i (Yellow) and k_d (Blue). For the ease of observation, the plots are created without gain being added. By comparing this to Figure 26, adjustment of the fuzzy PID was made to k_p, k_i and k_d for the sine wave to be seen.

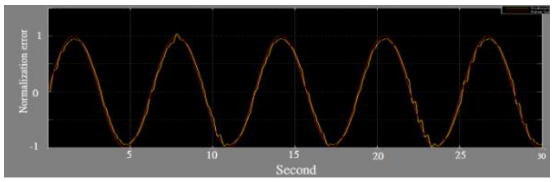


Figure 24. The Sine Wave Response of the Fuzzy Controller.

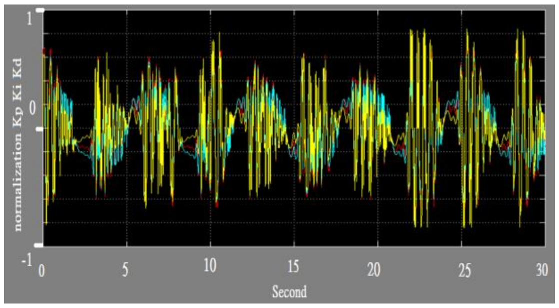


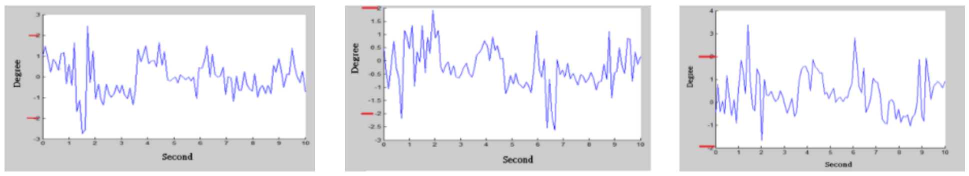
Figure 25. The Output of the Fuzzy Controller.

6.4 Posture Control Test

In the Fuzzy controller posture control test, Figure 26 (a-c) shows the errors that occurred in ϕ, θ and ψ during hovering. The error range (marked by the Red lines) is ± 2 degrees. The overall error was within three degrees.

6.5 Tracking the System Interface

The tracking system operational interface was implemented as a smartphone APP. On the operational interface, shown in Figure 27, the user can choose which object to track in Block A. The button in Block B is used to switch the current drone operational mode between manual control and automatic tracking. Buttons C and D are used for manual take-off and landing. Button E is used to abort current activity and force the drone to hover.



(a) Angle Error ϕ (b) Angle Error θ (c) Angle Error ψ

Figure 26. The Hover of the Angle Error.

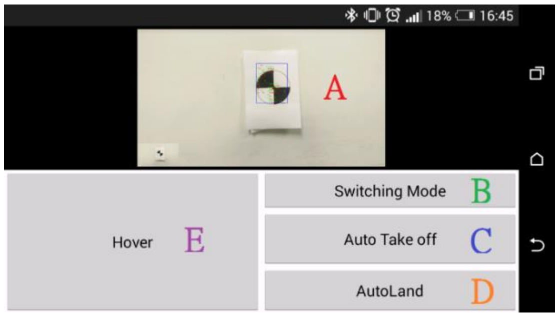


Figure 27. The User Interface.

7 CONCLUSION

THE purpose of this paper was the design of an autonomous tracking system for a quadrotor drone based on machine vision. The system carries out the image processing, quadrotor drone posture estimation and drone control. The image processing system, in addition to the TLD algorithm for tracking, maps the position of the object in 3D space using the IPM method to calculate the relative distance between the object and the drone body. The Kalman filter was applied for posture estimation using the original values from the gyroscope and its use effectively reduced the amount of the sensor noise. The posture estimation was achieved using quaternion representation by converting the gyroscope readings into three-axis angles of the drone's coordinate system. For the position estimation, the displacement of the drone was calculated using the accelerometers only. Steady hovering of the drone was not possible due to the existence of small errors; however, it was capable of fulfilling a tracking mission. The fuzzy-PID control was used for compensation and the problems of difficult parameter adjustment and the adaptability was solved through posture control. The portable ground control terminal was implemented as a smartphone APP, and the smartphone and remote controller being linked by a transmission channel. The user can launch the autonomous tracking system proposed in this paper through simple interaction.

8 REFERENCES

Bouguet, J-Y. (2001). "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation Microprocessor Research Labs*. 1-10.

Castillo, P., Dzul, A. and Lozano, R. (2004) "Real-time stabilization and tracking of a four-rotor mini rotorcraft," *IEEE Transactions on Control Systems Technology*. 12 (4) 510-516.

Choi, O. K., Kim, J. and Lee, J. S. (2015) "BIBO Stability Analysis of TSK Fuzzy PI/PD Control Systems," *Intelligent Automation & Soft Computing*. 21(4) 645-658.

Dang, C. T., Pham, H. T., Pham, T. B. and Truong, N. V. (2013) "Vision based ground object tracking using AR Drone quadrotor," *International Conference on Control, Automation and Information Sciences*. 146-151.

Diebel, J. (2006) "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors," *Stanford University*.

Engel, J., Sturm, J. and Cremers, D. (2012). "Camera-based navigation of a low-cost quadrocopter," *IEEE International Conference on Intelligent Robots and Systems*. 2815-2821.

Huang, T., Huang, D. and Luo, D. (2018) "Attitude tracking for a quadrotor UAV based on fuzzy PID controller," *2018 5th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*.19-24.

Kalal, Z., Mikolajczyk, K. and Matas, J. (2012) "Tracking-learning-detection," *IEEE Transactions in Pattern Analysis and Machine Intelligence*. 1409-1422.

Liu, C. and Prior, S. D. (2015). "Design and implementation of a mini quadrotor control system in GPS denied environments," *IEEE International Conference on Unmanned Aircraft Systems*.462-469.

Low, C. B. and Wang, D. (2008). "GPS-based tracking control for a car-like wheeled mobile robot with skidding and slipping," *IEEE Transactions on Mechatronics*.480-484.

Ludeman, L. C. (2003) "Random Processes: Filtering, Estimation, and Detection," John Wiley & Sons.

Muad, A. M., Hussain, A., Samad, S. A., Mustaffa, M. M., and Majlis, B. Y. (2004). "Implementation of Inverse Perspective Mapping Algorithm for the Development of an Automatic Lane Tracking System," *IEEE Region 10 Conference TENCON 2004*. 207-210.

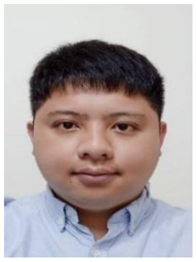
Sato, M. and Sato, Y. (1995) "On A General Fuzzy Additive Clustering Model," *Intelligent Automation & Soft Computing*. 1(4) 439-448.

Vetrella, A. R., Savvaris, A., and Fasano, G. (2015) "RGB-D camera-based quadrotor navigation in GPS-denied and low light environments using known 3D markers," *IEEE International Conference on Unmanned Aircraft Systems*. 185-192.

Valenci, D., and Kim, D. (2018) "Quadrotor Obstacle Detection and Avoidance System Using a Monocular Camera," *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems*. 78-81.

9 NOTES ON CONTRIBUTORS

Pi-Yun Chen received Ph.D. in the Graduate School of Engineering Science and Technology from National Yunlin University of Science & Technology, Taiwan, in 2011. She is now an associate professor in the Department of Electrical Engineering, National Chin-Yi University of Technology, Taiwan. Her current research interests include fuzzy systems and control systems.



Guan-Yu Chen received Master's degree in Electrical Engineering from the National Chin-Yi University Technology, Taiwan, in February 2017.