Sensors and Materials, Vol. 31, No. 7 (2019) 2245–2262
MYU Tokyo

S & M 1928

Realization of Person Tracking and Gesture Recognition with a Quadrotor System

Neng-Sheng Pai, Yue-Han Zhou, Pi-Yun Chen,*
Wei-Lun Chen, and Shih-An Chen

Department of Electrical Engineering, National Chin-Yi University of Technology,
No. 57, Sec 2, Zhongshan Rd, Taiping Dist. Taichung 41170, Taiwan (ROC)

(Received November 29, 2018; accepted May 8, 2019)

Keywords: quadrotor, gesture recognition, tracking–learning–detection, fixed point cruising, fuzzy-PID controller, nonlinear support vector machine, extended Kalman filter

In this paper, the design of a quadrotor vehicle having a person-tracking and observation system, which uses human gesture recognition, is described. The system has three operating functions, namely, object tracking, human gesture recognition, and fixed-point cruising. The tracking–learning–detection (TLD) algorithm was used to enable the autonomous tracking of the object from images. An extended Kalman filter (EKF) provides an estimate of the current position of the quadrotor vehicle, and a fuzzy-proportional integral derivative (PID) controller provides position error compensation. The principle of the human gesture recognition system is as follows. A background model is first built from images using a Gaussian mixture model (GMM) to detect the foreground image. A nonlinear support vector machine (SVM) is then employed to recognize changes of gesture and establish interactivity between the vehicle and the user. The coordinates of the vehicle are marked using a GPS for fixed-point cruising. The coordinates and parameters of the points are set so that the quadrotor vehicle can follow them during cruising. Lastly, all of the functions are incorporated into the person-tracking and gesture-recognition system in the quadrotor. The experimental results show the feasibility of the above-mentioned methods, which can help us easily recognize the various gestures in this study.

1. Introduction

A UN World Population Ageing report[1] points out that the percentage of the population aged over 60 increased from 9.2% in 1990 to 11.7% in 2013, and is projected to surge to 21.1% by 2050. This means that there is an escalating need for elderly care. Many smart home care concepts[2] have been proposed; some use wheeled or even humanoid robots.[3–5] Of these, remote care needs more attention. A remote care system involves people who receive care, those who give care, and family members. The care receiver (elderly) side includes a certain interactive platform and related devices such as cameras. The platform provides care receivers with easy access to basic services that they need such as an app for interactive entertainment,

*Corresponding author: e-mail: chenby@ncut.edu.tw
https://doi.org/10.18494/SAM.2019.2211

ISSN 0914-4935 © MYU K.K.

and teleconversations. Cameras facilitate the remote care provided by family members and caregivers. Live streaming videos ensure safety and the availablity of help in an emergency.

In this study a quadrotor aerial drone was introduced into the domain of smart care. Object tracking and mid-air navigation were achieved by image tracking using machine vision technology. The interaction between humans and the drone, by employing gestures, was also found to be feasible. An aerial drone is highly maneuverable, allows many viewing angles of an area, and can cover blinds spots, which fixed cameras cannot. Drones, unlike wheeled robots, do not suffer from a lack of mobility on rough terrain. The integration of these highly maneuverable aerial vehicles into a smart care system introduces many innovative applications to the field of smart care. Most of the commercially available drones allow manual operation that gives them good response capabilities. However, to use a drone in health care, the people involved should be familiar with the operation interface that can be complex. Since the elderly are involved, the drone must be easy to operate and be capable of autonomous operation and tracking the person being cared for.

2. System Architecture

A care system capable of tracking an object, human gesture recognition, and waypoint navigation using a quadrotor aerial drone is proposed here. The system architectural diagram is shown in Fig. 1.

The process starts with the tracking of the current camera image using the tracking–learning–detection (TLD) algorithm. Subsequent learning and detection allow the bounding boxes to be updated and the displacement of objects is calculated to pinpoint the position of the

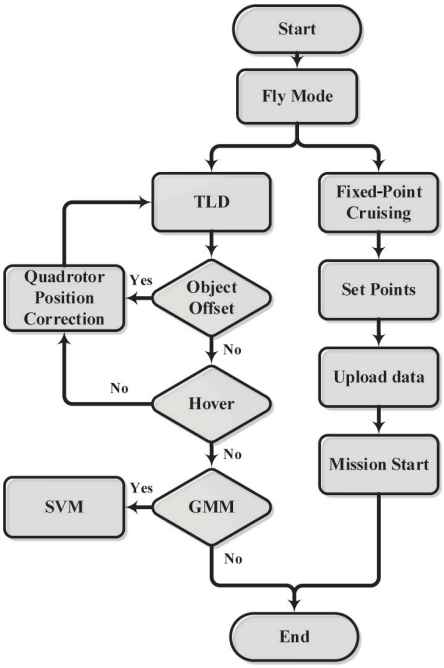


Fig. 1. (Color online) System architecture.

object in the image. A Gaussian mixture model (GMM) is established using the background of the image to allow the hovering drone to recognize human gestures using the support vector machine (SVM). In the waypoint navigation mode, the cruising points of the planned flight and related parameters must be set before navigation starts.

3. Tracking the Image of the Target Being Followed

The backgrounds of images from a stationery camera used for object tracking are usually still and stable, and a background subtraction method can be used to build the background model and obtain the foreground. However, the substantial changes in the background of a tracked object caused by the variations in the illumination, scale, and partial exclusion of the images from a drone-mounted camera must be considered. Therefore, the TLD algorithm[6] is used for object tracking. The TLD tracker algorithm uses the pyramid Lucas–Kanade (L–K) optical flow method[7] for tracking purposes. This method has the following advantages: there is no need for preliminary background modeling, it is more flexible than background subtraction methods, its use is not limited to a single scenario, and so forth. The flow chart of object tracking for this study is shown in Fig. 2.

3.1 TLD algorithm

The image of a tracked object may become distorted after tracking for a long time. This can be caused by the need for retracking after the object has been lost, which may cause tracking failure. The TLD algorithm delivers an outstanding performance in handling illumination changes, scale variations, and partial occlusion, and retracking of a lost target.

As shown in Fig. 3, TLD image tracking has three main components, i.e., tracking, learning, and detection, which all operate together. The pyramid L–K optical flow method[7] is used for tracking, while the detector is responsible for calculating the position of the tracked object in the image. The learning component carries out real-time error learning from the results of the tracker and detector to minimize the chance of tracking failure. The integrator combines and updates the bounding boxes of the tracker and detector.

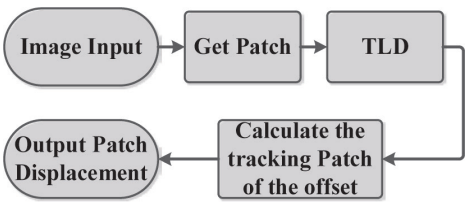


Fig. 2. (Color online) Target tracking flow chart.

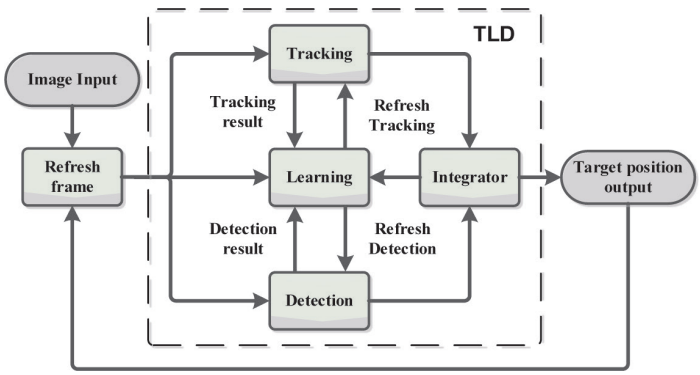


Fig. 3. (Color online) TLD flow chart.

3.1.1 Tracking

The TLD tracker algorithm employs the pyramid L–K optical flow method[7] for object tracking and also feeds back the result of object tracking using the forward–backward error,[8] as shown in Fig. 4. A comparison is then made between the forward–backward error and the Euclidean distance at the initial position, and tracking results with greater distances are discarded.

As shown in Fig. 4, the distance D between the forward and backward trajectories is the difference between the initial and end positions. The distance calculation is Euclidean. $S_{FB} = (H_t, H_{t+1}, \ldots, H_{t+n})$ represents the sequence of consecutive frames. H_t is the frame at time t and C_t is the position of C at time t. Forward tracking is conducted n times. The forward trajectory $T_F^n = (C_t, C_{t+1}, \ldots, C_{t+n})$ is obtained. n is the length of time and F is forward tracking. On the other hand, the backward trajectory $T_B^n = (\hat{C}_t, \hat{C}_{t+1}, \ldots, \hat{C}_{t+n})$ is obtained by backward tracking to the initial frame. B is backward tracking. Lastly, $\mathrm{FBE}(T_F^n | S_{FB}) = D(T_F^n, T_B^n)$ is obtained, where $D(T_F^n, T_B^n) = \left\| C_t - \hat{C}_t \right\|$.

The basic working principle of the pyramid L–K optical flow method is the detection of the changes of each pixel between two neighboring frames (using differentiation) to obtain the direction and speed of optical flow. It is assumed that a pixel K has displacement between two neighboring frames and so do the pixels q_n surrounding a pixel K with the same displacement. The optical flow equation is assumed to hold as well. The intensities of the pixel value on three dimensions, i.e., x, y, and time t are denoted as I_x, I_y, and I_t, respectively. The optical flow speeds between pixel K and the surrounding pixels q_n are V_x, V_y. The basic optical method is shown in the equation

$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n). \tag{1}$$

The matrix representation $\boldsymbol{AV} = \boldsymbol{B}$ is shown as

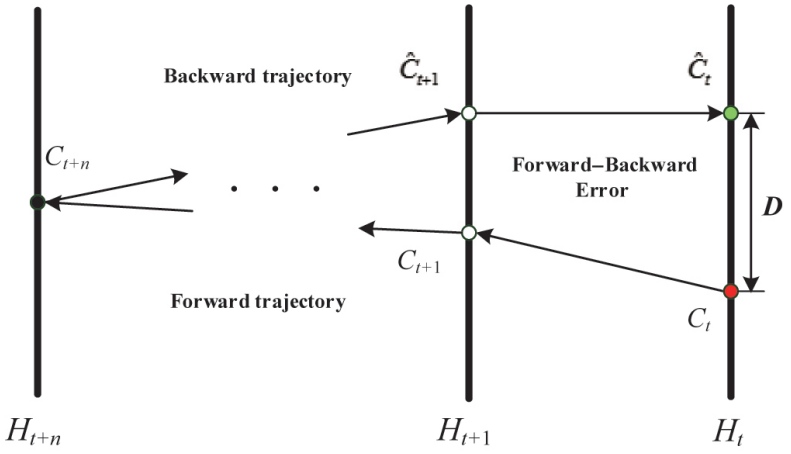


Fig. 4. (Color online) Forward–backward error.[9]

$$\boldsymbol{A}=\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix},\ \boldsymbol{V}=\begin{bmatrix} V_x \\ V_y \end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix} -I_y(q_1) \\ -I_y(q_2) \\ \vdots \\ -I_y(q_n) \end{bmatrix}. \tag{2}$$

The L–K optical flow method uses least squares to obtain approximate solutions, that is,

$$\boldsymbol{V}=\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B} \tag{3}$$

or

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B}\,. \tag{4}$$

Substitute Eq. (3) into Eq. (4) to obtain Eq. (5):

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix}=\begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(qq_i)I_y(k_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1}\begin{bmatrix} I_x(q_i)I_t(q_i) \\ I_y(q_i)I_t(q_i) \end{bmatrix}, \tag{5}$$

where = 1, 2, 3, ..., n. The optical flow direction is then obtained. The results obtained from optical flow estimation are passed to the integrator and tracker for evaluation. Then, the tracker is updated by the learning component.

3.1.2 Detection

The detector scans the input image frame through a scanning window and determines the presence or absence of the object for each patch. The detector shown in Fig. 5 is a cascade classifier.[6] Owing to the large number of frames to be processed, the classifier has three stages, namely, the patch variance, ensemble, and nearest-neighbor classifiers. The patches are

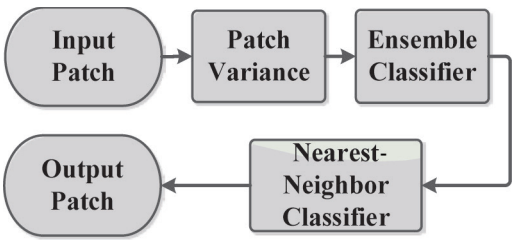


Fig. 5. (Color online) Schematic diagram of the detector.

first filtered by the patch variance and ensemble classifiers. The patches not rejected are kept and passed to the nearest-neighbor classifiers.

A. Patch variance classifier

During this stage, the patches with a gray-value variance below 50% are rejected. The gray-value variance equation for patch P is $E(P^2) - E^2(P)$ and the expected value $E(P)$ can be measured in real time using integral images. Typically, most of the nonobject patches are rejected during this stage. The variance threshold is preset to 50%, but can be manually adjusted if necessary.

B. Ensemble classifier

The ensemble classifier consists of m base classifiers. The patch is first applied with a Gaussian blur effect to increase the robustness to noise. Next, the pixels in each base classifier are compared and each comparison returns either 0 or 1. Take the comparison of arbitrary points A and B for example. The return value is 1 if the brightness of point A is greater than that of point B. Otherwise, 0 is returned. The results of the comparisons are entered as a binary code, which indexes to an array of posterior probabilities $P_j(y|x)$. The probability is estimated as $P_i(y \mid x) = \frac{\#p}{\#p + \#n}$, where $\#p$ and $\#n$ correspond to the numbers of positive and negative patches that were assigned the same binary code. The posterior probabilities of individual base classifiers are averaged. The ensemble classifier endures and only patches with posterior probabilities greater than 50% are passed to the next stage.

C. Nearest-neighbor classifier

The object model T is a collection of positive patches P_n^+ and negative patches P_m^-. It is a data structure that represents the object and its surrounding thus far observed, where P^+ and P^- represent the object and background patches, respectively. The object model is shown as

$$T = \left\{ P_1^+, P_2^+, \ldots, P_n^+, P_1^-, P_2^-, \ldots, P_m^- \right\}. \tag{6}$$

In this method, the spatial similarity of two bounding boxes is measured using the overlap, which is defined as the ratio of the intersection to the union. The shape of an object is represented by patch P. The similarity between the two patches P_j and P_k is defined as

$$S\left(P_j, P_k\right) = 0.5\left(NCC\left(P_j, P_k\right) + 1\right), \tag{7}$$

where NCC is a normalized correlation coefficient. Given an arbitrary patch P and the object model T, several similarity measures are defined for P–N learning[6]

(1) Similarity with the positive nearest neighbor:

$$S^+\left(P, T\right) = \max_{P_j^+ \in T} S\left(P, P_j^+\right) \tag{8}$$

(2) Similarity with the negative nearest neighbor:

$$S^{-}\left(P, T\right) = \max_{P_j^{-} \in T} S\left(P, P_j^{-}\right) \tag{9}$$

(3) Similarity with the positive nearest neighbor considering the 50% earliest positive patches:

$$S_{50\%}^{+}\left(P, T\right) = \max_{P_j^{+} \in T \wedge j < n/2} S\left(P, P_j^{+}\right) \tag{10}$$

(4) Relative similarity:

$$S^{r} = \frac{S^{+}}{S^{+} + S^{-}} \tag{11}$$

The relative similarity ranges from 0 to 1, where higher values mean a greater confidence that the patch depicts the object, i.e., the foreground.

(5) Conservative similarity: The conservative similarity ranges from 0 to 1.

$$S^{c} = \frac{S_{50\%}^{+}}{S_{50\%}^{+} + S^{-}} \tag{12}$$

A high value indicates a greater confidence that the patch resembles the appearance observed in the first 50% of positive patches. The preset threshold is $\theta_{NN} = 0.6$. A patch P is classified as a positive object if $S^{r}\left(P, T\right) > \theta_{NN}$.

3.1.3 Learning component

The learning component uses a semisupervised learning method.[6] The classification is analyzed by P- and N-experts, which estimate examples that have been classified incorrectly. Figure 5 shows the flow chart of P–N learning. The main task of P–N semisupervised learning is to give incorrectly classified positive and negative examples to the respective P- and N-experts for analysis. The P-expert analyzes samples incorrectly classified as negative, estimates false negatives, and adds them to the training set with a positive label. The N-expert analyzes examples classified as positive, estimates false positives, and adds them to the training set with a negative label.

Equations (13) and (14) show the numbers of examples corrected by the P- and N-experts, respectively.

$$n^{+}\left(i\right) = n_{C}^{+}\left(i\right) + n_{F}^{+}\left(i\right) \tag{13}$$

$$n^{-}(i) = n_C^{-}(i) + n_F^{-}(i) \tag{14}$$

Here, at the ith iteration of training, n^{+} is the number of examples relabeled positive by the P-expert. n_C^{+} and n_F^{+} are respectively the numbers of examples correctly and incorrectly relabeled as positive. n^{-} is the number of examples relabeled as negative by the N-expert. n_C^{-} and n_F^{-} are respectively the numbers of examples correctly and incorrectly relabeled as negative. $\alpha(i)$ and $\beta(i)$ are the numbers of positive and false negative errors, respectively. Their equations are shown below.

$$\alpha(i+1) = \alpha(i) - n_C^{-}(i) + n_F^{+}(i) \tag{15}$$

$$\beta(i+1) = \beta(i) - n_C^{+}(i) + n_F^{-}(i) \tag{16}$$

If $n_C^{-}(i) > n_F^{+}(i)$, i.e., the number of examples correctly relabeled as negative, is higher than the number of examples incorrectly relabeled as positive, then false positives $\alpha(i)$ will decrease, as shown in Eq. (15). Similarly, if $n_C^{+}(i) > n_F^{-}(i)$, the false negatives $\beta(i)$ will decrease, as shown in Eq. (16).

3.1.4 Integrator

The integrator combines the bounding boxes of the tracker and detector into a single bounding box output. The object information is passed to the learning component for classification purposes. If neither the tracker nor the detector outputs a bounding box, the object is declared invisible. The integrator outputs the maximally confident bounding box. Object tracking resumes as soon as the object is detected in the image again.

4. Human Gesture Recognition

To recognize human gestures, a background model is first built using the GMM method. The foreground (i.e., human gestures) is detected. Human gestures are divided into upper- and lower-body gestures. Lower-body gestures include left leg up, right leg up, standing on both legs, and kneeling. Upper-body gestures include right hand up, left hand up, both hands down, both hands flat, both hands holding head, and so forth. The recognition techniques in this study will focus on the human's full-body gestures when falling and upper-body gestures when standing. The SVM[9–13] algorithm is used in this study to recognize human gestures. It is used to train and build the model with sample data. The trained model is used later in data classification and regression.

4.1 GMM

The GMM of the background image is constructed using multiple Gaussian models with similar background color distribution densities and its mathematical equation is shown below:

$$P(X_t) = \sum_{i=1}^{k} \omega_{i,t} \cdot \eta(X_t | \mu_{i,t}, \sigma_{i,t}), \tag{17}$$

$$\begin{cases} \omega_{k,t} = (1-\gamma)\omega_{i,t-1} + \alpha(M_{k,t}), \\ \mu_t = (1-\rho)\mu_{t-1} + \rho X_t, \\ \sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^2, \\ \rho = \gamma\eta(X_t \mid \mu_k, \sigma_k), \end{cases} \tag{18}$$

where $\omega_{i,t}$ represents the weight of the ith Gaussian distribution η. X_t is a random variable. The average of Gaussian distributions is $\mu_{i,t}$. The standard deviation is $\sigma_{i,t}$. Equation (18) is the update equation for the Gaussian background where γ represents the learning speed. $M_{k,t}$ is the matched Gaussian distribution. If the current pixel value matches the Gaussian distribution, $M_{k,t}$ is 1 and the average and standard deviation are updated. Otherwise, $M_{k,t}$ is 0 and no update is performed.

4.2 SVM

The SVM algorithm is composed of two parts. The first part is the analysis of linear systems. Nonlinear system analysis is performed through the nonlinear mapping of the nonseparable low-dimensional examples into high-dimensional feature spaces. Nonlinearly inseparable examples are thus changed to linearly separable ones. The above method allows the linear and nonlinear systems to use the same method for analysis and processing. The second part of the SVM algorithm is the structural risk minimization (SRM) in feature spaces to build an optimal support hyperplane separation so that the expected risk of the sample space satisfies the upper limit with optimal probability and the overall system is optimized. The goals of the SVM algorithm are to build an object function by SRM and to separate the two types of model optimally.

4.2.1 Nonlinear SVM algorithm

If the optimal hyperplane is constructed from training data using a linear approach, the final classification error is huge and data points are difficult to separate. In this situation, a nonlinear method must be used to separate the data points. Boser *et al.*[12] proposed the use of a nonlinear function to separate data points. Function $\varphi(x_i)$ is used to map input data to a feature space of a higher dimension as shown in Fig. 6.

The equation is rewritten as Eq. (19) on the basis of $\varphi(x_i)$.

$$\begin{aligned} &\text{minimize} \quad J_p(w, \xi) = \frac{1}{2} w^{\mathrm{T}} \cdot w + c\sum_{i=1}^{n} \xi_i \\ &\text{subject to} \quad \begin{cases} y_i\left(w^{\mathrm{T}} \cdot \varphi(x_i) + b\right) \geq 1 - \xi_i, & i = 1, 2, \ldots, n \\ \xi_i \geq 0 \end{cases} \end{aligned} \tag{19}$$

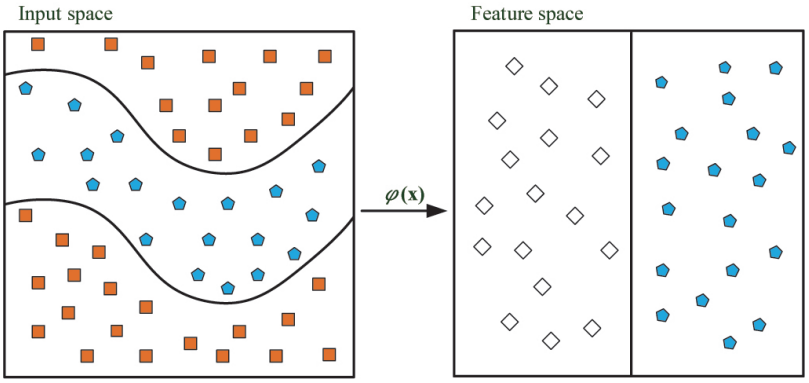


Fig. 6. (Color online) Schematic diagram of space transformation.

Equation (19) is then rewritten as Eq. (20) by the Lagrange multiplier method.[12]

$$\begin{aligned} L(w,b,\xi,a,v) &= J(w,\xi) - \sum_{i=1}^{n} a_i \left(y_i \left(w^{\mathrm{T}} \varphi(x_i) + b \right) - 1 + \xi_i \right) - \sum_{i=1}^{n} v_i \xi_i \\ &= \frac{1}{2} w^{\mathrm{T}} \cdot w + c \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} a_i \left(y_i \left(w^{\mathrm{T}} \varphi(x_i) + b \right) \right) + \sum_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i \xi_i - \sum_{i=1}^{n} v_i \xi_i \end{aligned} \tag{20}$$

Here, $a_i > 0$ and $v_i > 0$ are both Lagrange multipliers. To find the smallest L, Eq. (21) is obtained by considering the partial derivative of Eq. (20).

$$\begin{cases} \dfrac{\partial L}{\partial w} = 0 \rightarrow w = \displaystyle\sum_{i=1}^{n} a_i y_i \varphi(x_i) \\ \dfrac{\partial L}{\partial b} = 0 \rightarrow \displaystyle\sum_{i=1}^{n} a_i y_i = 0 \\ \dfrac{\partial L}{\partial \xi_i} = 0 \rightarrow c - a_i - v_i = 0 \end{cases} \tag{21}$$

Substitute Eq. (21) into Eq. (19) to obtain the following equation.

$$\begin{aligned} &\text{maximize} \quad J_D(a) = -\frac{1}{2} \sum_{i,l=1}^{n} a_i a_l y_i y_l \varphi(x_i)^{\mathrm{T}} \varphi(x_l) + \sum_{i=1}^{n} a_i \\ &\text{subject to} \quad \begin{cases} \displaystyle\sum_{i=1}^{n} a_i y_i = 0 \\ 0 \le a_i \le c, \quad i = 1, 2, \ldots, n \,. \end{cases} \end{aligned} \tag{22}$$

Here, $\varphi(x_i)^{\mathrm{T}}\varphi(x_l)$ is defined as the kernel function,[13] which is shown as

$$K(x_i, x_l) = \varphi(x_i)^{\mathrm{T}} \cdot \varphi(x_l). \tag{23}$$

According to the literature,[13] the kernel function satisfies Mercer's condition, as shown in the equation

$$\int K(x_i, x_l) g(x_i) g(x_l) dx_i dx_l \geq 0, \tag{24}$$

where $g(x)$ is an integrable function and can be chosen as a kernel function if it satisfies Mercer's condition. Common kernel functions include the polynomial kernel, multilayer perception, and radial basis function, which are shown in Eqs. (25), (26), and (27), respectively.
(1) Polynomial kernel

$$K(x_i, x_l) = \left(\gamma\left(x_i^{\mathrm{T}} \cdot x_l + 1\right)\right)^d \tag{25}$$

(2) Sigmoid kernel

$$K(x_i, x_l) = \tanh\left(\gamma x_i^{\mathrm{T}} \cdot x_l + d\right) \tag{26}$$

(3) Radial basis function

$$K(x_i, x_l) = e^{-\gamma\|x_i - x_l\|^2} \tag{27}$$

The choice of a kernel function depends on the classification problem to be solved. Different results are obtained depending on the parameters used. The radial basis function, the most often used kernel function for the SVM algorithm, is used in this paper.

5. Modeling and Control of a Quadrotor Drone

In this section, the mathematical model and control method of a quadrotor drone are presented. The flow chart of drone attitude control is shown in Fig. 7. The inertial sensors used

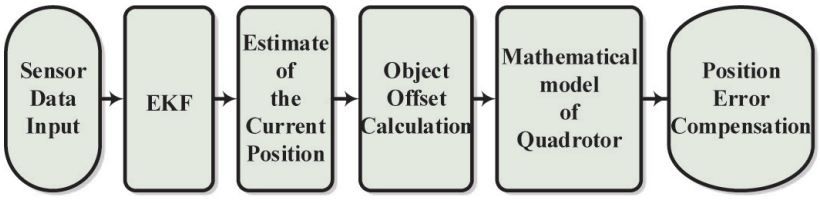


Fig. 7. (Color online) Flow chart of drone attitude control.

in this study include both gyroscopes and accelerometers. The Euler angles are first obtained by taking the integral of the angular velocities measured using the gyroscope and converting them into body coordinates. The drone position is obtained by double-integrating the accelerometer output. The extended Kalman filter (EKF) is used to filter the noise from the three-axis accelerometer data, after which the current attitude of the drone is estimated. The position error of the tracked object is calculated and used by fuzzy-proportional integral derivative (PID) control to compensate for the drone data, which is then input to the mathematical model. The magnitude of compensation is calculated to adjust the attitude of the drone.

5.1 EKF

To estimate the drone attitude, the angular velocities of the pitch ϕ, roll θ, and yaw ψ axes on the body coordinate system are measured using the gyroscope. That is, the angle on each axis is obtained by integrating the derivative of the angle over time. However, this method is subject to error that grows over time. This problem can be solved using the Kalman filter.

In the prediction state, the evolution function of the state estimate $\hat{\boldsymbol{x}}_{k,k-1}$ is shown in Eq. (28), where $\boldsymbol{\Phi}_{k-1}$ is the state transition matrix. $\boldsymbol{Q}_{k-1}$ is the state noise covariance matrix. The covariance matrix equation $\boldsymbol{P}_{k,k-1}$ is shown in Eq. (29).

$$\hat{\boldsymbol{x}}_{k,k-1} = \boldsymbol{\Phi}_{k-1}\boldsymbol{x}_{k-1,k-1} \tag{28}$$

$$\boldsymbol{P}_{k,k-1} = \boldsymbol{\Phi}_{k-1}\boldsymbol{P}_{k-1,k-1}\boldsymbol{\Phi}_{k-1}^{\mathrm{T}} + \boldsymbol{Q}_{k-1} \tag{29}$$

In the update state, the measured value is $\boldsymbol{z}_k$. The state estimate at k is shown in Eq. (30). The Kalman gain $\boldsymbol{K}_k$ can be obtained as in Eq. (31). $\boldsymbol{H}_k$ is the measurement module matrix as shown in Eq. (32). $\boldsymbol{R}$ is the measurement noise covariance matrix. The covariance matrix function from time k−1 to k $\boldsymbol{P}_{k,k-1}$ is shown in Eq. (33).

$$s\hat{\boldsymbol{x}}_{k,k} = \boldsymbol{x}_{k,k-1} + \boldsymbol{K}_k\left(z_k - \hat{z}_k\right) \tag{30}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k,k-1}\boldsymbol{H}_k^{\mathrm{T}}\left(\boldsymbol{H}_k\boldsymbol{P}_{k,k-1}\boldsymbol{H}_k^{\mathrm{T}} + \boldsymbol{R}_k\right)^{-1} \tag{31}$$

$$\boldsymbol{H}_k = \left.\frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k} \tag{32}$$

$$\boldsymbol{P}_{k,k} = \left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right)\boldsymbol{P}_{k,k-1} \tag{33}$$

5.2 Fuzzy-PID controller of the quadrotor drone

The mathematical model of the drone is shown in Eq. (34).

$$\begin{aligned} m_Q \ddot{x}_b &= -u \sin\theta \\ m_Q \ddot{y}_b &= u \cos\theta \sin\phi \\ m_Q \ddot{z}_b &= u \cos\theta \cos\phi - m_Q g \\ \ddot{\phi} &= \tilde{\tau}_\phi \\ \ddot{\theta} &= \tilde{\tau}_\theta \\ \ddot{\psi} &= \tilde{\tau}_\psi \end{aligned} \tag{34}$$

The attitude can be calculated using the obtained coordinate systems. The error calculated from the tracked object is used to compensate for and correct the attitude of the drone using the fuzzy-PID controller, which is introduced below.

In this study, each fuzzy-PID controller[14] considers the error e and the change in error, de, as input variables. The output variables are k_P, k_I, and k_D. The input and output membership functions are defined as shown in Fig. 8. The adjustment of k_P can raise the proportional gain of the control system, shorten the response time of the system, and reduce the steady-state error. However, a proportional gain that is very high may cause system instability. k_I is then used to eliminate the steady-state error of the system. A large k_I means that the steady-state error of the system will be eliminated faster. k_D improves the system error in dynamic responses and suppresses the change in error in the response process.

The fuzzy-PID controller deals mainly with three cases. In case 1, when $|e|$ is large, a larger k_P and a smaller k_D are preferred, and k_I must be as close to zero as possible so that the error can be rapidly eliminated and the system response can also be shortened. In case 2, $e \cdot de > 0$. When $|e|$ is large, a larger k_P, an appropriate k_D, and a smaller k_I are preferred. Otherwise, an appropriate k_P, a smaller k_D, and a larger k_I are preferred to prevent oscillation and increase system stability. In the last case, $e \cdot de < 0$. If $|e|$ is large, appropriate k_P and k_D, and a smaller k_I are preferred. If $|e|$ is small, smaller k_P and k_D, and a larger k_I are preferred to increase system stability.

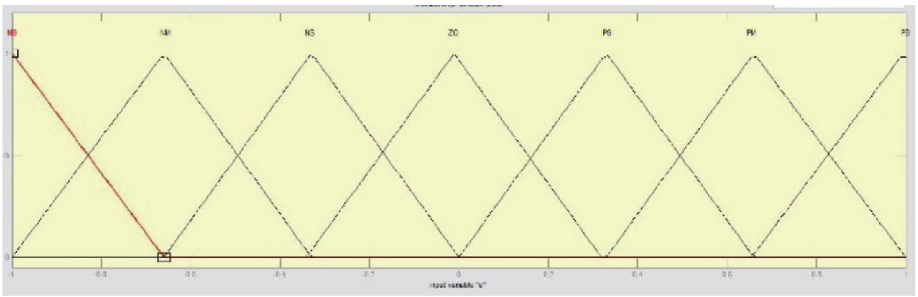


Fig. 8. (Color online) Schematic diagram of input membership function.

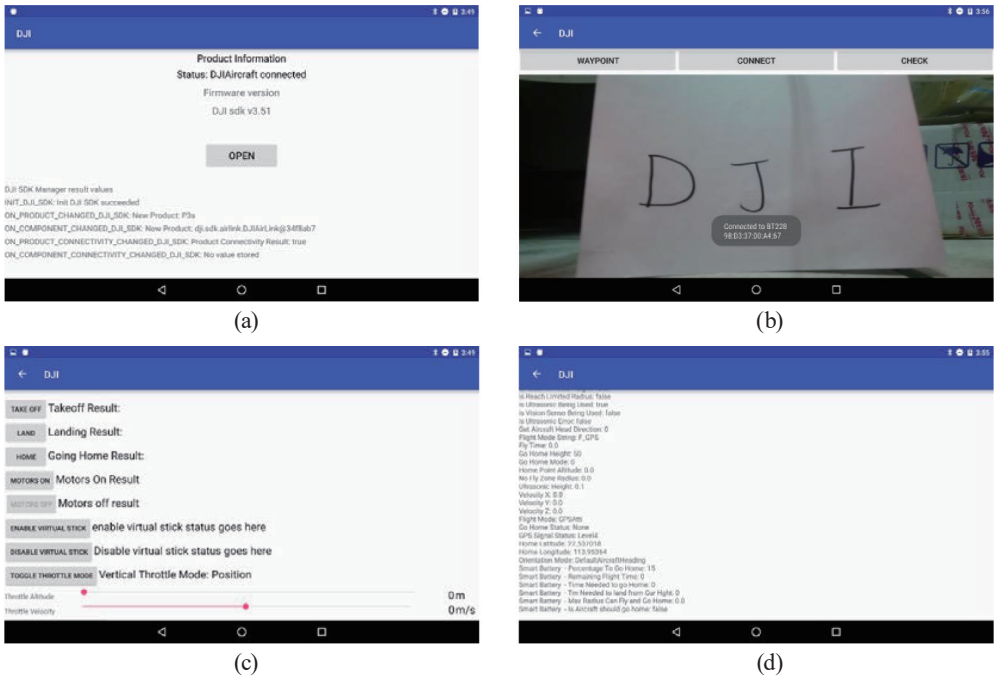
6. Experimental Results

A tracking care system, based on the experimental results presented above, was implemented using a DJI quadrotor drone. Figure 9(a) shows the screen of a tablet PC displaying initialization information when the drone and RF remote controller are connected. Figure 9(b) shows the drone operating interface on the tablet PC, which is used to switch the flight mode and confirm flight images and Bluetooth connection. Figures 9(c) and 9(d) show the control interface and related information.

As soon as the drone arrives at a preset position, the ground end begins to process the images received and TLD object tracking is launched. The drone can then carry out object tracking on the basis of the calculated tracking error. Figure 10(a) shows the ground end marking the bounding box. Figures 10(b), 10(d), 10(f), 10(h), and 10(j) show the tracked object moving first to the left and then to the right. Figures 10(c), 10(e), 10(g), and 10(i) show the view angle from behind the quadrotor drone. It can be seen that autonomous tracking was achieved.

When the tracked object stops, the drone will hover and the human gesture recognition system will begin to operate. Foreground detection is achieved through GMM background modeling and human gesture recognition is implemented through the SVM.[15] The main function being to help the caregivers on the ground better understand user gestures and needs so that further actions can be taken. Figures 11(a)–11(d) are screens that show drone hovering, human gesture recognition system activation, and gesture recognition performance.

Lastly, the drone was switched to the waypoint navigation mode, performed waypoint navigation care, and monitored the area at all times to ensure the safety of the care receiver. Figures 12(a)–12(d) show the screens displaying waypoint navigation care flight.



(a) (b) (c) (d)

Fig. 9. (Color online) Operating interface of the quadrotor drone.

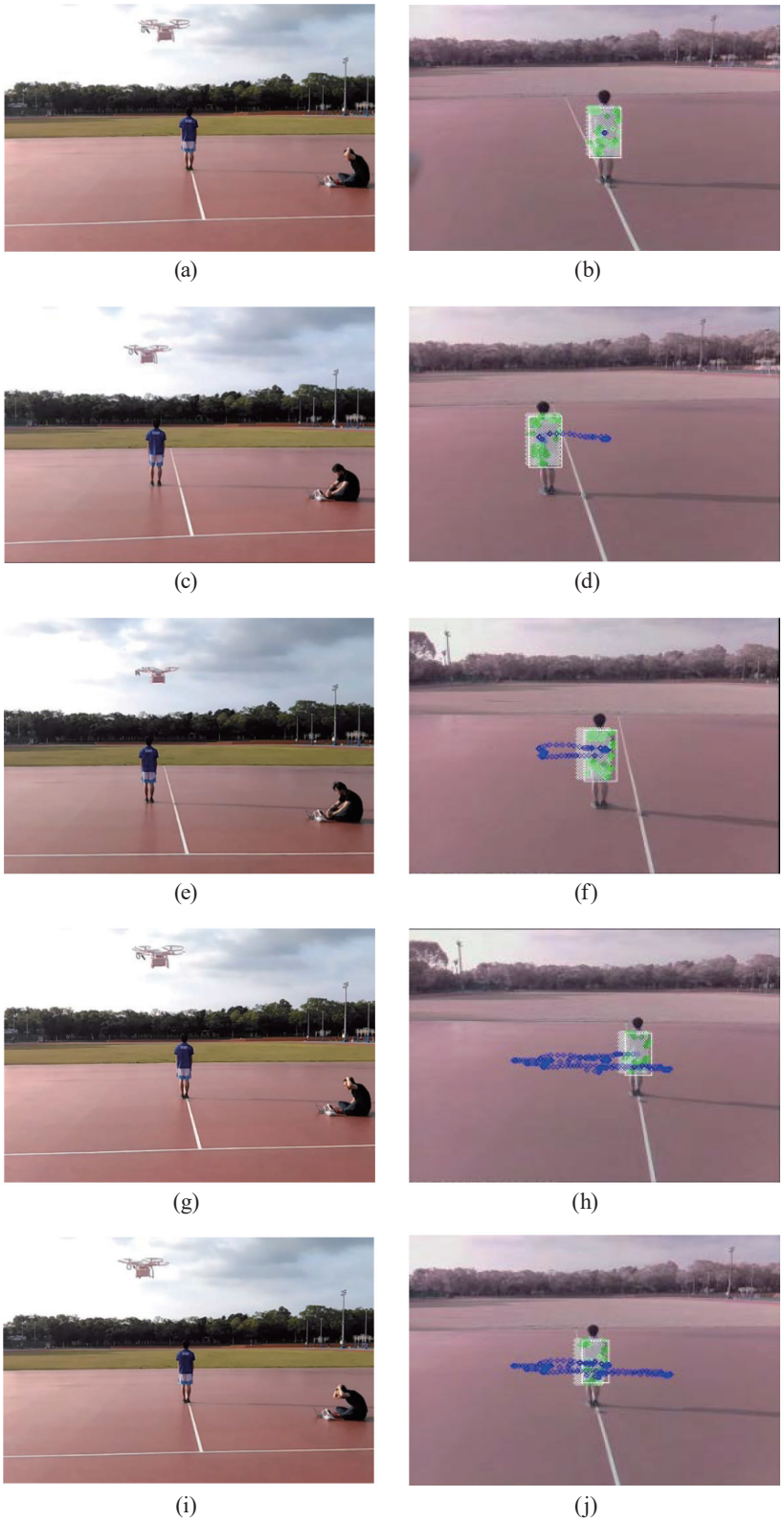
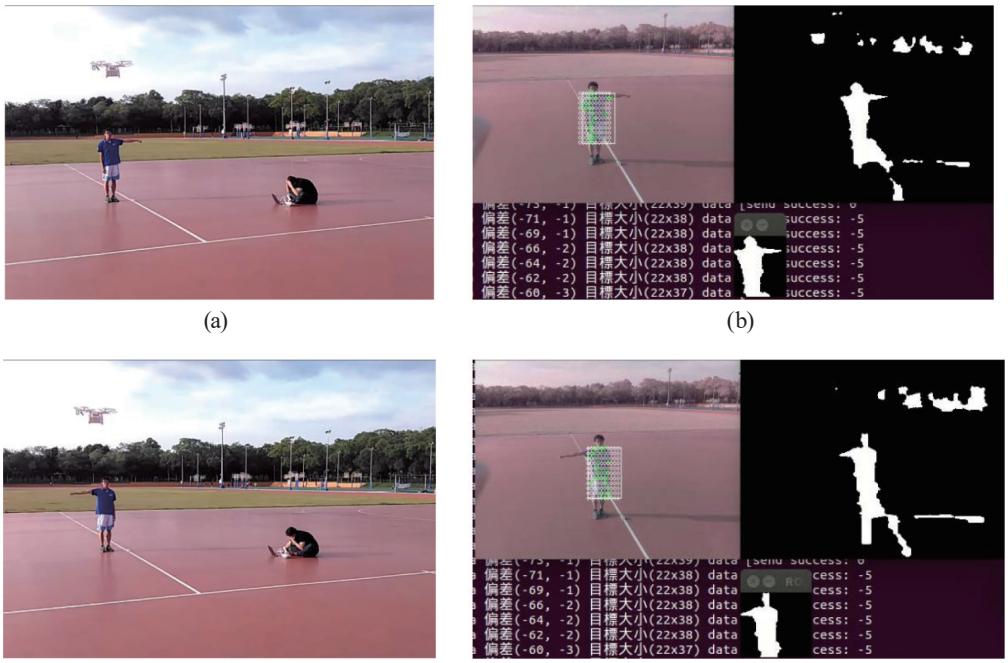


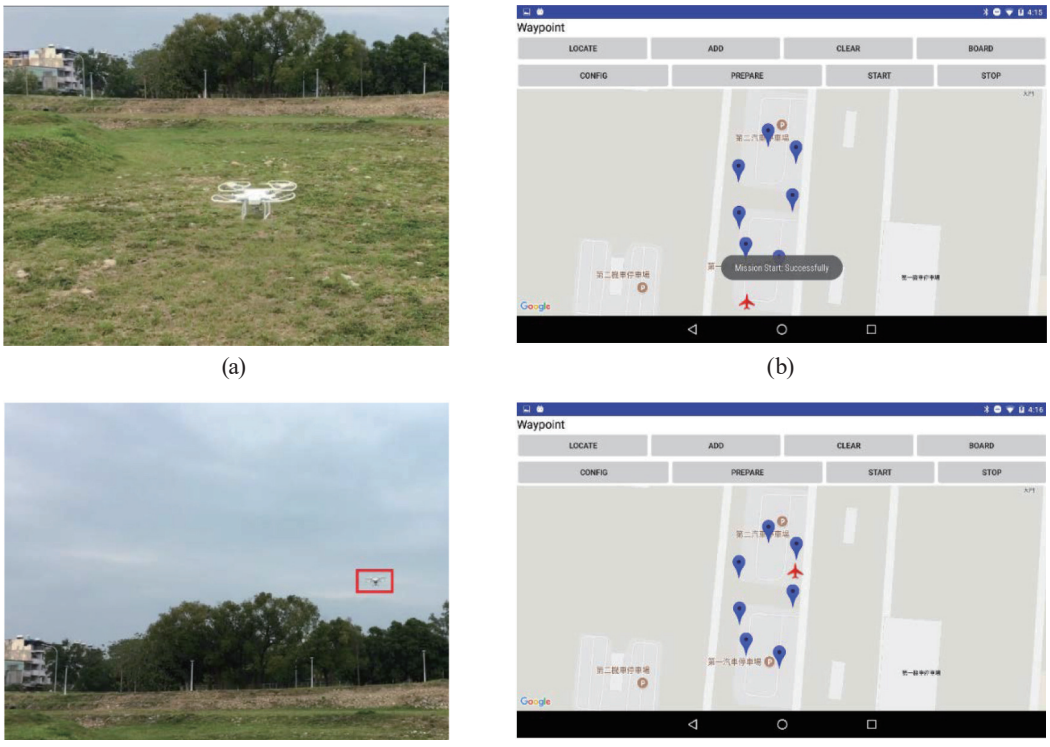
Fig. 10. (Color online) Autonomous tracking system.



(a) (b)

(c) (d)

Fig. 11. (Color online) Human gesture recognition system.



(a) (b)

(c) (d)

Fig. 12. (Color online) Waypoint navigation care system.

7. Conclusion

In this study, a quadrotor drone was used as a platform for the design of an autonomous tracking care system. The system has three functional aspects, namely, object tracking, human gesture recognition, and waypoint navigation. The TLD image tracking algorithm, which is good for dealing with background changes, was used for object tracking. The TLD algorithm has many advantages. For example, in a case where the tracked object is lost, the detector will recover and resume tracking. The learning component of the TLD algorithm improves tracking accuracy. A Kalman filter was used to estimate the current attitude of the drone, and displacement was calculated using the position of the tracked object received from the drone. Error compensation was implemented using a fuzzy-PID controller and autonomous object tracking was achieved.

To implement human gesture recognition, the images and GMM were used to build a background model and detect the foreground. Although the GMM method requires much computation, it is better for handling small background changes, such as those created by vegetation. The SVM is used to recognize human gestures by identifying the body motion of the tracked person.

Waypoint navigation is carried out using a smartphone app we developed. The coordinates of the quadrotor drone are set first and the navigation-related coordinates and parameters, which are required as waypoint navigation instructions, are then entered and uploaded to the drone. Lastly, a quadrotor drone tracking care system was implemented using the methods mentioned above. This system can be used at nursing homes, by home care providers, and in any place where there are people requiring remote care.

References

1 Science & Technology Policy Research and Information Center: World Population Prospects: The 2015 Revision. https://portal.stpi.narl.org.tw/index/ article/10250
2 S. Majumder, E. Aghayi, M. Noferesti, H. Memarzadeh-Tehran, T. Mondal, Z. Pang, and M. J. Deen: Sensors **17** (2017) 2496. https://www.mdpi.com/1424-8220/17/11/2496
3 M. H. Honarvar, T. Suzuki, S. Sato, and Y. Nakamura: Proc. 2nd RSI/ISM Int. Conf. Robotics and Mechatronics (IEEE, 2014) 684. https://doi.org/10.1109/ICRoM.2014.6990982
4 X. Zhao, A. M. Naguib, and S. Lee: Proc. 23rd IEEE Int. Symp. Robot and Human Interactive Communication (2014) 525. https://doi.org/10.1109/ROMAN.2014.6926306
5 T. Tabata, Y. Kobayashi, and Y. Kuno: Proc. IECON 2013 - 39th Annu. Conf. IEEE (2013) 8312. https://doi.org/10.1109/IECON.2013.6700525
6 C. T. Dang, H. T. Pham, T. B. Pham, and N. V. Truong: Proc. 2013 Int. Conf. Control, Automation and Information Sciences (2013) 146. https://doi.org/10.1109/ICCAIS.2013.6720545
7 J. Y. Bouguet: Intel Corporation (2001) 1.
8 Z. Kalal, K. Mikolajczyk, and J. Matas: Proc. 2010 20th Int. Conf. Pattern Recognition (ICPR) (IEEE 2010) 2756. https://doi.org/10.1109/ICPR.2010.675
9 R. Rifin and A. Klautau: J. Mach. Learn. Res. **5** (2004) 101.
10 U. H. G. Kreβel: Pairwise Classification and Support Vector Machines (MIT Press, Cambridge, MA, 1999) p. 255.
11 C. Cortes and V. Vapnik: Machine Learning **20** (1995) 273.
12 B. E. Boser, I. M. Guyon, and V. N. Vapnik: Proc. Fifth Annual Workshop on Computational Learning Theory (1992) 144.
13 B. Scholkopf and A. J. Smola: Learning with Kernels (MIT Press, Cambridge, MA, 2001).

14 G. Y. Chen: Master Thesis, Department of Electrical Engineering, National Chin-Yi University of Technology (2016).
15 J. K. Wu: Master Thesis, Department of Electrical Engineering, National Chin-Yi University of Technology (2015).