

Vision-Only Automatic Flight Control for Small UAVs

Chung-Cheng Chiu, *Member, IEEE*, and Ching-Tung Lo, *Student Member, IEEE*

Abstract—In this paper, a vision-based flight control system that uses a skyline-detection algorithm is developed for application to small unmanned aerial vehicles. The skyline-detection algorithm can detect straight or uneven skylines. The system integrates a remote controller, a remotely controlled airplane, a camera, a wireless transmitter/receiver, a ground control computer, and the proposed skyline-detection algorithm to achieve automatic control of flight stability. Static and dynamic tests are conducted to validate the system. In the static tests, the average accuracy rate for skyline detection is 98.62% based on five test videos. In the dynamic tests, straight and circular flights are used to verify lateral and longitudinal stability for the proposed flight control system. The experimental results demonstrate the performance and robustness of the algorithm and the feasibility and potential of a low-cost vision-only flight control system.

Index Terms—Automated vehicles, skyline detection, vision control.

I. INTRODUCTION

OVER the last decade, the potential of unmanned aerial vehicles (UAVs) has been demonstrated in D-cube missions that are identified as dangerous, dirty, or dull. Recently, considerable interest has arisen in small-scale lightweight low-cost unpiloted aircraft (known as small UAVs) designed to fly at low altitudes with limited payloads and power constraints. Aside from military deployment, small UAVs are widely applicable in civilian scenarios, because they are easy to build, carry, and launch. When equipped with vision sensors, small UAVs can execute numerous monitoring missions, including remote sensing, traffic monitoring, forest fire surveillance, accident reconnaissance, mapping, search, and rescue. Vision sensors are indispensable for carrying out these missions. In addition, a robust, reliable, and inexpensive autopilot system is necessary, and attitude determination and control are the key to achieving autonomy [1], [2].

Aircraft attitude is determined from the Euler angles, pitch, roll, and yaw [1]. To measure and control these three angular parameters, an attitude determination and control system is

essential. Traditionally, the attitude problem has been solved by using microelectromechanical systems (MEMS)-based inertial sensors, which offer the advantages of small size, light weight, and low power consumption [4], [5]. However, the sensors of MEMS are vulnerable to fabrication imperfections, temperature fluctuations, and calibration errors; therefore, the accuracy of MEMS data rapidly degrades over time due to sensor bias, drift, and noise [6]. Thus, the use of noninertial sensors such as a charge coupled device/complementary metal-oxide-semiconductor (CCD/CMOS) camera for obtaining valuable attitude information is regarded as an alternative approach in attitude determination and control research [7]–[11].

Recent studies have demonstrated the feasibility of evaluating the artificial horizon during autonomous flight by using an onboard imaging sensor without increasing the payload [12]–[22]. The two degrees of freedom critical for stability—the roll angle ϕ and the pitch angle θ —can be derived from the slope of the horizon and the distance from the central point of an image to the horizon, respectively. In several studies, the horizon is extracted under the assumption of a straight skyline. Accordingly, finding the best fit horizon is considered to be a maximum edge-detection problem. Bao *et al.* [12] use an orientation-projection technique applied to binary images to detect the horizon. The projection value is defined as the number of edge pixels along the selected direction of projection in a binary image. The horizon may then be obtained from the maximum projection values. Moreover, as the maximum votes in the Hough space can be represented as straight lines, the Hough transform is employed in [13], [14], [16], [21], and [22] to choose the maximum votes as the horizon. However, the skyline is not the horizon, and the skyline is not always straight; it becomes uneven when the contours of trees or buildings become a part of it. If the skyline is uneven or multiple straight lines such as roads or runways are present in the images, these methods may not extract the correct horizon from the maximum projection or peak value.

Based on the assumption of straight skylines, Cornall *et al.* [13] apply a circular mask to binary images to find a bisector line that divides the sky and the ground into two symmetric areas. Because bisector lines are perpendicular to the horizon, calculating the slope of the bisector passing through the centroids of the sky and the ground can be used to derive the inclination of the horizon. However, when an object, e.g., a propeller, appears on an image, it interferes with this type of detection. Thus, Pereira *et al.* [14] propose a propeller-removal algorithm for horizon detection. Because the position of a propeller varies in consecutive images, their technique involves subtracting consecutive images and using a logical operator to detect the

Manuscript received September 7, 2010; revised January 17, 2011 and April 5, 2011; accepted April 28, 2011. Date of publication May 27, 2011; date of current version July 18, 2011. This work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-606-020. The review of this paper was coordinated by Prof. S. Ci.

This paper has supplementary downloadable material available at <http://ieeexplore.ieee.org>, provided by the authors.

The authors are with the Department of Electrical and Electronics Engineering, Chung Cheng Institute of Technology–National Defense University, Taoyuan County 335, Taiwan (e-mail: davidchiu@ndu.edu.tw).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TVT.2011.2157545

propeller and remove it. The Hough transform is then employed to detect the horizon. Nevertheless, in addition to propeller interference, dark clouds or white buildings may affect horizon detection.

To make sky and ground segmentation more accurate, Fefilatyev *et al.* [15] have devised some classifiers to produce better binary images for horizon detection. To extract useful edge features in complex images, Dush *et al.* [16] propose an edge-based approach to morphologically utilize edge information from the red, green, and blue channels. After the edges from the different color channels have been extracted and correlated through logical operations, the Hough transform is again employed. McGee *et al.* [21] presented a sky segmentation method based on a support vector machine (SVM) to classify the sky and the nonsky regions. The border between the sky and the nonsky regions is found from the binary image that is segmented by the sky segmentation method. The Hough transform was employed to detect the horizon from the border image. Once the horizon has been found, all nonsky pixels above the horizon are regarded as obstacles in the path of the aircraft. However, the horizon detection may not detect the horizon when the horizon is not a straight line. Dong *et al.* [22] proposed a binarization method based on the local complexity of background to detect the sea-skyline. The Hough transform was again employed to extract the straight line as a sea-skyline from the binary image. In their study of the interference of haze over a horizon, Yuan *et al.* [17] apply a haze-removal technique to extract a horizon from foggy aerial images. Nonetheless, these methods sometimes produce a time-consuming bottleneck and, thus, present difficulties in real-time applications.

Although the aforementioned straight-line-based techniques have achieved specific goals, they cannot guarantee correct results when an uneven skyline appears. To solve the uneven-skyline-detection problem, several relevant studies are noted here. Lie *et al.* [18] propose an edge-linking method for detecting the skyline according to the adjacent characteristics of edge points that extend from one side of the image to the other. However, their technique is sensitive to unwanted edges, particularly when the edges appear in the neighborhood of the skyline. Because the skyline separates the image into sky and ground distributions, Ettinger *et al.* [19] utilize pixel-distribution information from color statistics to find a best fit horizon that maximizes the between-class variance of the two pixel regions. However, this color-based method may not be effective when the appearance of sky and ground do not conform to a Gaussian distribution. To ensure the performance of the technique, it is necessary to consider both color and texture features. Thus, Todorovic *et al.* [20] propose a block-based method to decompose an image into a number of subimages. Next, a vector descriptor is extracted from the different subimages through a technique similar to Ettinger's approach. The vector descriptors are then classified into different groups, and the maximum group is chosen to represent the skyline. Unfortunately, this technique is not computationally efficient for conducting a full search of every video frame. In addition, controlled by the full-search detection methods, the flying attitude may dramatically change while capturing subsequent noise-corrupted frames.

In light of the aforementioned studies, the following challenges to skyline extraction.

- 1) Uneven skylines cannot be neglected in real flight due to the low operational altitudes of small UAVs.
- 2) The processing images are complicated by varied lighting conditions, luminous intensity variations that result from the auto-white balance and auto-gain control of the camera, aircraft vibrations due to wind gusts, and noise interference in video signal transmissions.
- 3) The intensities of the sky and ground are not always homogeneous because of dark clouds or white buildings.

This paper introduces a vision-based flight control system for small UAVs using a novel skyline-detection algorithm. The skyline-detection algorithm is divided into the initial and the tracking stages. In the initial stage, because the skyline intersects the boundaries of the processed image at two points, the localized areas around the boundaries of the processed image are defined to be the region of interest (ROI). Accordingly, a search algorithm for the points of intersection in the new ROI is proposed. In the tracking stage, a tracking algorithm for the points of intersection is proposed to decrease both processing time and noise interference. Because the two terminal points of a skyline intersect the boundaries of an image, regardless of whether the skyline is straight or uneven, the regions around the boundaries of an image are the ROIs that detect the terminal points of a skyline. Compared with other global-search methods, the proposed local-search algorithm can focus on the ROI to detect and track the terminal points of a skyline to speed up the processing time. In addition, most edge textures in the image will not influence the detection results, because the ROI is confined to a local area.

Designing a vision-based flight control system is an elaborate undertaking. It involves the systematic integration of the airplane, skyline-detection algorithm, and controller for real-time applications. The most important issues in designing a vision-based system are real-time constraints, robustness to noise, and reliable detection. Hence, this paper proposes a vision-based flight control system with the following features.

- 1) The hardware integration of the system can be implemented in a low-cost radio-controlled aircraft, and its feasibility is therefore demonstrated.
- 2) Compared with other existing algorithms, the skyline-detection algorithm exhibits superior accuracy with regard to complex images.
- 3) The algorithm can detect both straight and uneven skylines.

The remainder of this paper is organized as follows. Section II describes the hardware integration of the proposed system. A detailed discussion of the skyline-detection algorithm is presented in Section III. Comparative experimental results from actual flight tests are provided in Section IV. Section V contains concluding remarks and directions for future research.

II. HARDWARE INTEGRATION OF THE PROPOSED SYSTEM

The vision-based flight control system comprises a UAV subsystem and an image-processing and transmitter subsystem, as shown in Fig. 1.

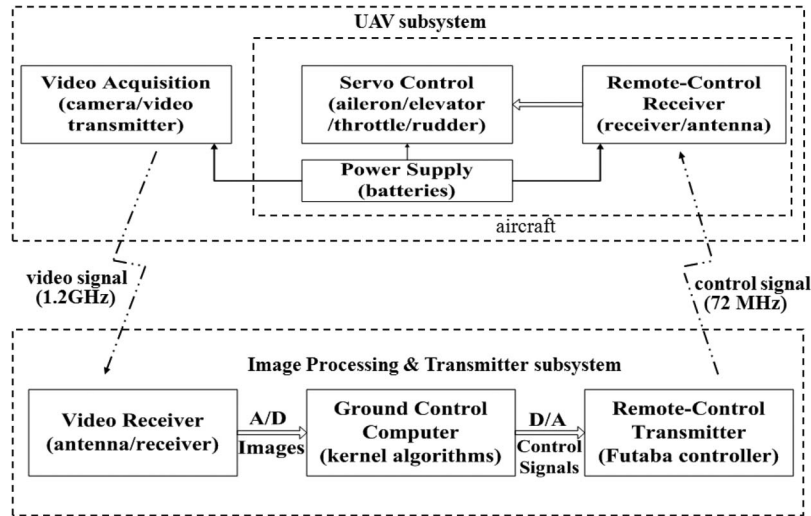


Fig. 1. Proposed system.

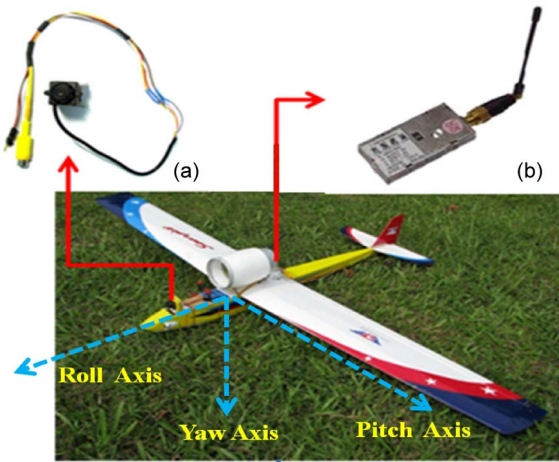


Fig. 2. UAV subsystem. (a) CMOS camera. (b) Video transmitter.

A. UAV Subsystem

The UAV subsystem integrates a video acquisition module, a remote control receiver module, a servo control module, and a power supply module on a radio-controlled aircraft. The aircraft is a SOARJET EP-19 (fuselage length: 87 cm), high-wing (wingspan: 140 cm), four-channel (controlling rudder, ailerons, elevator, and throttle), battery-powered, single-motor aircraft. The subsystem is shown in Fig. 2. The aircraft weighs about 600 g and carries the four modules, which have a combined weight of roughly 187 g. The video acquisition module consists of a 1/3-in CMOS camera that is $20 \times 20 \times 20$ mm in size and 15 g in weight, together with a 1.2-GHz wireless transmitter and antenna.

To make the aircraft coordinate system coincident with the camera coordinate system and to avoid encroachments of the aircraft body in the image frame, the camera is mounted on top of the nose of the aircraft, and video images are transmitted by the video transmitter. The optical axis of the camera is parallel to the roll axis of the aircraft, and the vertical and horizon lines of the image are parallel to the yaw and pitch axes, respectively. Because the camera is mounted on the aircraft, the variations of the roll and pitch angles of the aircraft can be estimated from

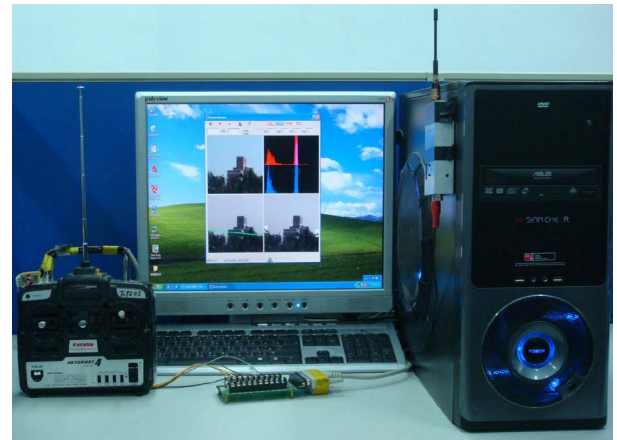


Fig. 3. Image-processing and transmitter subsystem.

the roll and pitch changes of the horizon in the image. Video images that are received by a ground-based receiver may be subject to interference caused by wind gusts, engine vibrations, and electromagnetic noise. Therefore, the proposed skyline-detection algorithm is divided into two stages (the initial and the tracking stages) to decrease the interference. Once the skyline-detection algorithm has been executed, roll and pitch data are computed, and control signals are transmitted to the remote control receiver in the aircraft. The ailerons and the elevator are controlled by these signals. Because aircraft stability mainly depends on roll and pitch angles, this paper uses the roll and pitch data that are detected by the skyline-detection algorithm to control the aileron and elevator channels, whereas the other two channels are controlled by an experienced pilot. Using two 12-V Li-Poly batteries, the power module provides all the necessary power for the UAV subsystem. One battery is used for the engine, remote control receiver, and servos, and the other battery is used for the video-acquisition module.

B. Image-Processing and Transmitter Subsystem

The image-processing and transmitter subsystem comprises a video receiver module, a ground control computer (GCC)

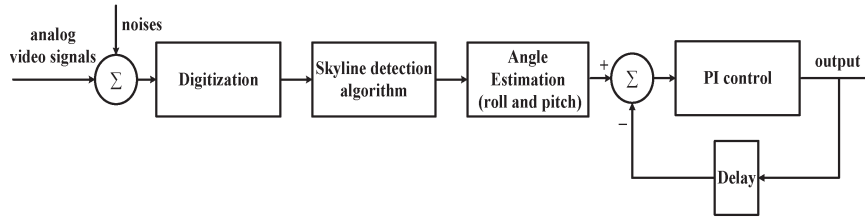


Fig. 4. Diagram of the tasks performed by the image-processing kernel.

module, and a remote control transmitter module, as shown in Fig. 3. The video receiver module is responsible for receiving the analog video signals from the video transmitter of the UAV subsystem. When the video signals are received by the video receiver, they are sent to a frame-grabber card in the GCC. The images are digitalized by the grabber card and then processed by the GCC, which is equipped with a Pentium IV 2.6-GHz processor and 1-GB random access memory (RAM).

The tasks that are performed by the GCC include digitalizing the video signals, detecting the skyline, estimating the attitude of the aircraft, and sending the control signals to the remote control transmitter. A flowchart is shown in Fig. 4, and the procedure is discussed in detail in Section III. The control signals from the GCC are sent to the remote control transmitter through an analog/digital converter and an RS232 connector.

III. SKYLINE-DETECTION ALGORITHM

The video images that are captured by a stationary charge-coupled device (CCD) camera contain both the background and moving object images. Because most of the background image is motionless, the color components red (R), green (G), and blue (B) of the background pixels should not change from one frame to the next during the background extraction. Unfortunately, several background pixels change in an image sequence because of moving objects that pass through these background pixels and the variation in illumination. Therefore, each pixel in an image sequence will have as many different colors as the candidates for a background pixel. For any given pixel of an image sequence, the probability of the background color will be higher than the probabilities of the colors counted when the moving objects pass through in the image sequence. Previous studies [1], [21], [22] used this concept to develop background extraction algorithms. However, the memory consumption and processing time required to obtain the initial background image are major drawbacks, because these probability-based algorithms use an image sequence in a fixed-time duration to count the probabilities of each color for each pixel. The algorithm described in this paper resolves the problems with regard to the large-memory requirement and processing time.

In general, skylines are straight, as long as aircraft fly at high altitudes. Because of their limited payloads and power constraints, small UAVs are designed to fly at low altitudes so that the uneven contours of buildings or forests become part of the skyline. Therefore, skylines can be either straight or uneven for small UAVs. Compared with straight skylines, uneven skylines make it more difficult to extract the horizon. Hence, a novel algorithm is proposed to detect and track a skyline based on the terminal points of the skyline.

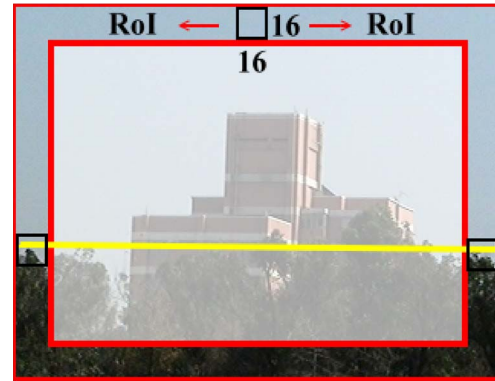


Fig. 5. ROI in the initial stage.

Captured images are first transferred from the color plane (RGB) to the gray-scale plane, and the ROI is defined at the boundaries of the captured images. In this paper, the ROI's determination can be divided into the following two parts: 1) the initial stage and 2) the tracking stage. In the initial stage, the ROI is determined on the whole regions that surround the image boundaries, as shown in Fig. 5. In addition, the ROI is divided into blocks of size $s \times s$ pixels, which overlap the neighboring blocks to a width of $s/2$ pixels. In the initial stage, each block of the ROI is classified as a sky or a ground block by a thresholding technique based on discriminant analysis. Then, a ground block that is adjacent to a sky block can be regarded as a candidate block where the terminal point is located. However, the adjacency characteristic is not reliable, because the threshold is sensitive to illumination changes, which can result in the incorrect classification of sky or ground blocks. Because the candidate block contains both sky and ground regions, the variance of the intensity in the candidate block is large. Therefore, the variance of the candidate block is compared with a variance-based threshold to confirm the candidate block. Finally, the intersection of the sky and the ground is extracted from the candidate block to complete the skyline-detection process.

In normal flight attitude, the sky and ground pixels in an image are distributed on the upper and lower sides of the skyline, respectively. Provided that there is no occlusion of propellers or fuselage near the boundaries of the image, the two terminal points of a skyline composed of obvious edge points can be found on opposite boundaries of the image frame. Once these two terminal points have been detected, a virtual horizon is created to provide the following two important parameters for flight control: 1) the roll angle and 2) the pitch distance.

The proposed skyline-detection algorithm is composed of the initial and the tracking stages. The initial stage is utilized to detect the initial terminal points of the skyline along the ROI of

the images, and the tracking stage is used to track the terminal points in the current frame from the detected skyline of the previous frame. After the small hand-launched UAV takes off, it is manually operated by an experienced pilot. When the UAV reaches the appropriate altitude, the initial stage is activated until the control mode is switched from manual to autonomous, at which time the detection algorithm is switched from the initial stage to the tracking stage.

A. Detection Algorithm for the Initial Stage

The detection algorithm for the initial stage uses image blocks of $s \times s$ pixels, which overlap half the width of the neighboring blocks for increased accuracy along the ROI while searching the candidate blocks (where the terminal points of the skyline may be located).

Let the gray-level distribution of block bl range from 1 to m . The probability distribution of gray level i in a single block bl is denoted by

$$p_{b_i} = \frac{C_i}{\sum_{j=1}^m C_j}, \text{ where } i = 1, 2, \dots, m, \text{ and } \sum_{i=1}^m p_{b_i} = 1. \quad (1)$$

The parameters C_i and $\sum_{j=1}^m C_j$ denote the number of pixels with gray level i and the total number of pixels in a given block, respectively. The mean and variance of bl are given by

$$\mu_{bl} = \sum_{i=1}^m i p_{b_i} \quad (2)$$

$$\sigma_{bl}^2 = \sum_{i=1}^m (i - \mu_{bl})^2 p_{b_i}. \quad (3)$$

To distinguish whether a block belongs to the sky or the ground, the algorithm compares the mean value to a given threshold through the following equation:

$$Label_{bl} = \begin{cases} 1, & \text{if } \mu_{bl} > T_1 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $Label_{bl}$ represents either a sky or a ground block, which is identified by 1 or 0, respectively.

To select the threshold T_1 from the gray levels of all the blocks, a pixel-based thresholding technique based on discriminant analysis is adopted. Suppose that the pixel values of a given ROI, which covers the region of all the blocks, with gray level $\{1, 2, \dots, L\}$ can be divided into the following two classes: 1) $C_{\text{sky}} = \{1, 2, \dots, k\}$ and 2) $C_{\text{ground}} = \{k+1, k+2, \dots, L\}$. Here, k is the threshold value. Given that the probabilities of the pixel values of the two classes are $C_{\text{sky}} : \{p_1, p_2, \dots, p_k\}$ and $C_{\text{ground}} : \{p_{k+1}, p_{k+2}, \dots, p_L\}$, the probabilities of the classes are

$$\omega_{\text{sky}}(k) = \sum_{i=1}^k p_i \quad (5)$$

$$\omega_{\text{ground}}(k) = \sum_{i=k+1}^L p_i \quad (6)$$

where p_i is the probability distribution of gray level i in all the blocks.

The mean values of the two classes and the total mean value μ_T in the ROI are computed as follows:

$$\mu_{\text{sky}}(k) = \sum_{i=1}^k i \times \frac{p_i}{\omega_{\text{sky}}(k)} \quad (7)$$

$$\mu_{\text{ground}}(k) = \sum_{i=k+1}^L i \times \frac{p_i}{\omega_{\text{ground}}(k)} \quad (8)$$

$$\begin{aligned} \mu_T &= \omega_{\text{sky}}(k) \times \mu_{\text{sky}} + \omega_{\text{ground}}(k) \times \mu_{\text{ground}} \\ &= \sum_{i=1}^L i \times p_i. \end{aligned} \quad (9)$$

Based on the aforementioned statistical formulations, the between-class variance and the total variance in the ROI are given by

$$\sigma_B^2(k) = \omega_{\text{sky}} \times (\mu_{\text{sky}} - \mu_T)^2 + \omega_{\text{ground}} \times (\mu_{\text{ground}} - \mu_T)^2 \quad (10)$$

$$\sigma_T^2 = \sum_{i=1}^L (i - \mu_T)^2 p_i. \quad (11)$$

To measure the separability between the sky and the ground, a separable factor is applied to the algorithm. The factor is defined by

$$SF(k) = \frac{\sigma_B^2(k)}{\sigma_T^2} \quad (12)$$

where σ_T^2 serves as a normalization factor, and $SF(k)$ is a normalization value. Accordingly, an optimal threshold for distinguishing a sky block from a ground block can be obtained when the separable factor is maximized. Hence, the threshold T_1 is obtained from

$$T_1 = \arg \max_{1 < k < L} \{SF(k)\}. \quad (13)$$

Candidate blocks for skyline detection are then chosen from all the labeled blocks. A ground block that is adjacent to a sky block can be regarded as a candidate block. Intuitively speaking, if the sky blocks in the sky regions are correctly labeled, a ground block next to a sky block is the most likely candidate to contain a piece of the skyline. However, the adjacency characteristic is not reliable, because the aforementioned thresholding technique is sensitive to illumination changes, which can result in the incorrect classification of sky or ground blocks (e.g., dark clouds in the sky region may be classified as ground blocks, and bright buildings or light reflection in the ground region may be classified as sky blocks). Thus, the difficulties involved in detecting the correct skyline will be increased by the possible misclassification of candidate blocks.

To confirm the candidate blocks, the algorithm compares the block variance with a variance-based threshold using the following criterion:

$$Candidate_{bl} = \begin{cases} \text{true}, & \text{if } \sigma_{bl}^2 > T_2 \\ \text{false}, & \text{otherwise} \end{cases} \quad (14)$$

where $Candidate_{bl}$ represents the chosen candidate block. Because the candidate block contains both sky and ground regions, the variance of the intensity in the candidate block is large. Let the total number of blocks in the ROI be N . The threshold T_2 is then defined by

$$T_2 = \frac{1}{N} \sum_{bl=1}^N \sigma_{bl}^2. \quad (15)$$

Finally, the intersection of the sky and the ground is extracted from the candidate block to complete the skyline detection. Based the aforementioned algorithm, the details of the proposed algorithm for the initial stage are given as follows.

- Step 1) Transform the captured color image into a gray-level image $g(x, y)$, where $0 \leq x < m_W$, and $0 \leq y < n_H$.
- Step 2) Partition the ROI into blocks of size $s \times s$ pixels, which overlap the neighboring blocks to a width of $s/2$ pixels. The total number of blocks in the ROI is N . The block size $s \times s$ is set equal to 16×16 .
- Step 3) Compute the block mean μ_{bl} and block variance σ_{bl}^2 of each block through (2) and (3), where $\{l \in Z : 1 \leq l \leq N\}$.
- Step 4) Compare the block mean μ_{bl} with the threshold T_1 determined by (13). If $\mu_{bl} > T_1$, mark the block as a sky block and set $label_{bl} = 1$; otherwise, mark it as a ground block and set $label_{bl} = 0$.
- Step 5) Search for two candidate blocks in both the left- and right-hand directions, starting from one of the sky blocks. Compare the ground-block variance σ_{bl}^2 with the threshold T_2 determined by (15). If $\sigma_{bl}^2 > T_2$, mark the ground block as a candidate block; otherwise, relabel the ground block as a sky block. Repeat the processing until two candidate blocks are found.
- Step 6) Segment the pixels of the two candidate blocks into sky and ground regions using the mean values of the candidate blocks. In the candidate blocks, the boundaries between the sky and ground regions define the edge points of the skyline. The edge points intersect the image frame at the terminal points of the skyline.

B. Detection Algorithm for the Tracking Stage

In the tracking stage, a new ROI is confined to the estimated region of the terminal points of the skyline to avoid image interference and increase the processing speed. The linear Kalman filter is employed to track the terminal points from consecutive frames. The new ROI is the search area of size $r \times r$ pixels surrounding the estimated point from the linear Kalman filter. Here, r is set equal to 3^c , and the initial value of c is set equal to 2. Because the Kalman filter can estimate the locations of the terminal points, the size of the search area (ROI) is reduced to 9×9 ($c = 2$). When the terminal point is not detected from the search area, the parameter c is increased by 1 to enlarge the search area. Therefore, the ROI of the tracking stag can

extend the search area from 9×9 ($c = 2$) to 27×27 ($c = 3$) according to the detection result.

Because the time interval between two consecutive frames is minimal, the estimation can be accomplished using equations of motion without acceleration, i.e.,

$$\begin{cases} x_{f+1} = x_f + \dot{x}_f \cdot \Delta t \\ y_{f+1} = y_f + \dot{y}_f \cdot \Delta t \end{cases} \quad (16)$$

where x_f and y_f are the coordinates of the terminal points in the f th frame, and Δt is the time interval. The linear Kalman filter for tracking the terminal points from consecutive frames is modeled by

$$X_{f+1} = \Phi \cdot X_f \quad (17)$$

where the state vector is defined by

$$X_{f+1} = [x_{f+1}, y_{f+1}, \dot{x}_{f+1}, \dot{y}_{f+1}]^T \text{ and } X_f = [x_f, y_f, \dot{x}_f, \dot{y}_f]^T \quad (18)$$

and the state transition matrix that is derived from (16) is given by

$$\Phi = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (19)$$

After applying the Kalman filter, the estimated coordinates are regarded as the central point of an $r \times r$ search area. The terminal point search is conducted in this area.

To extract the edge features of the search area, the algorithm compares the magnitude of the gradient with a given threshold through the following equation:

$$Edge_{point} = \begin{cases} true, & \text{if } mag(\nabla g(x, y)) > T_3 \\ false, & \text{otherwise} \end{cases} \quad (20)$$

where the gradient vector of an image $g(x, y)$ is defined as

$$\nabla g = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} \quad (21)$$

and the magnitude of the gradient vector is given by

$$mag(\nabla g) = [G_x^2 + G_y^2]^{1/2}. \quad (22)$$

For computational efficiency, the gradient operational equation that is applied to approximate (22) has the form

$$\begin{cases} G_x(x, y) = mask_x * l(x, y) \\ G_y(x, y) = mask_y * l(x, y) \end{cases} \quad (23)$$

where $*$ denotes convolution, $l(x, y)$ is a neighborhood of (x, y) , and the two orthogonal masks are defined by

$$mask_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, \quad mask_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}. \quad (24)$$

The threshold value T_3 is defined as

$$T_3 = \begin{cases} 50, & \text{if } \beta \times \mu_r > 50 \\ \beta \times \mu_r, & \text{if } \beta \times \mu_r \leq 50 \end{cases} \quad (25)$$

where the mean value of the local gradient image is given by

$$\mu_r = \frac{1}{R} \sum_{x=0}^{r-1} \sum_{y=0}^{r-1} \text{mag}(\nabla g(x, y)) \quad (26)$$

and R is the number of $\text{mag}(\nabla g) \neq 0$. The weight coefficient β is set equal to 0.35.

The detected edge points may include unnecessary points, e.g., clouds, terrain, or noise. To determine the candidates for the terminal points of the skyline from among the edge points, the algorithm compares the difference of the sky and the ground with a given threshold T_4 through the following criterion:

$$\text{Edge}_{\text{terminal}} = \begin{cases} \text{true}, & \text{if } |\mu_{\text{up}} - \mu_{\text{down}}| \geq T_4 \\ \text{false}, & \text{otherwise} \end{cases} \quad (27)$$

where μ_{up} and μ_{down} are the mean values of the upward and downward vertical pixels of an edge point, respectively. A terminal point of the skyline is found when $|\mu_{\text{up}} - \mu_{\text{down}}|$ is maximized.

Details of the proposed algorithm for the tracking stage are given as follows.

- Step 1) Create a search area of size $r \times r$ pixels using the estimated point from (17). Here, r is set equal to 3^c , and the initial value of c is set equal to 2.
- Step 2) Compute the gradient values in the search area based on (23) and threshold the gradient values to obtain the edge points using (20).
- Step 3) Detect the candidates for terminal points through (27). Here, T_4 is set equal to 30.
- Step 4) If all edge points in the search area are labeled $\text{Edge}_{\text{terminal}} = \text{false}$, increase the parameter c by 1 to enlarge the search area. When the parameter c is greater than 3, decrease the threshold T_4 by 5. Then, go to step 2. If not all edge points in the search area are labeled $\text{Edge}_{\text{terminal}} = \text{false}$, go to Step 5.
- Step 5) Choose the candidate with maximum difference $|\mu_{\text{up}} - \mu_{\text{down}}|$ as the terminal point.

Once the two terminal points of the skyline have been detected, the virtual horizon that connects the terminal points can be used to estimate the roll angle (ϕ) and pitch distance (η) as the flight attitude parameters, as shown in Fig. 6.

The roll angle can be derived from the inverse tangent of the slope of the virtual horizon using

$$\phi = \arctan\left(\frac{j_r - j_l}{i_r - i_l}\right) \times \frac{180}{\pi}, \text{ where } -90^\circ < \phi < 90^\circ. \quad (28)$$

For small UAVs, the pitch angle cannot directly be measured from the virtual horizon of an image, because the pitch angle is related to the altitude of the UAV and the distance to the virtual horizon. Therefore, the distance \overline{MD} from the central

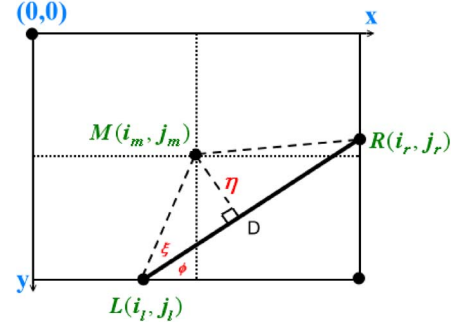


Fig. 6. Roll angle and the pitch distance.

point of the image to the virtual horizon, which is known as pitch distance η , is substituted to estimate the pitch angle, i.e.,

$$\overline{MD} = |\overline{LM}| \cdot \sin(\xi) \quad (29)$$

where

$$\xi = \arccos(\overline{LM} \cdot \overline{LR} / (|\overline{LM}| \cdot |\overline{LR}|)).$$

IV. EXPERIMENTAL RESULTS

In this section, we use static and dynamic tests to demonstrate that the proposed algorithm is effective to stabilize the flight of a UAV. The system integration and static tests are first described in detail. Then, the experimental results for the skyline-detection algorithm are presented based on test videos in various environments. The test videos indicate that the algorithm can correctly identify the skyline more than 98% of the time. Finally, in the dynamic tests, the algorithm is employed to control the stability of a small airplane, and the flight data show that the attitude of the airplane remains stable during autonomous control.

A. System Integration and Static Tests

Because the roll angle and the pitch distance are detected from the virtual horizon by the GCC, the flight control signals of the airplane are analyzed by the GCC and are transmitted through a remote controller. To manage the interface between the GCC and the remote controller, the aileron and elevator input-output (I/O) circuit of the remote controller is modified to connect to the analog-to-digital/digital-to-analog (AD/DA) board mounted on the GCC, as shown in Fig. 7. In addition, a switch is installed on the remote controller to switch the control mode between manual and automatic.

When the switch is set to the manual mode, the remote controller is operated by the pilot. When the switch is set to the automatic mode, the remote controller is controlled by the GCC. Because the remote controller only accepts voltage signals, the control signals from the GCC must be converted to voltage signals according to the roll angle and the pitch distance. The input voltage of the remote controller ranges from 0 V to 5 V. In this paper, the ranges of the roll angle and the pitch distance are defined as $-90^\circ \sim 90^\circ$ and -120 pixels ~ 120 pixels, respectively. Because the servo that is installed in the airplane controls the ailerons and elevator through the

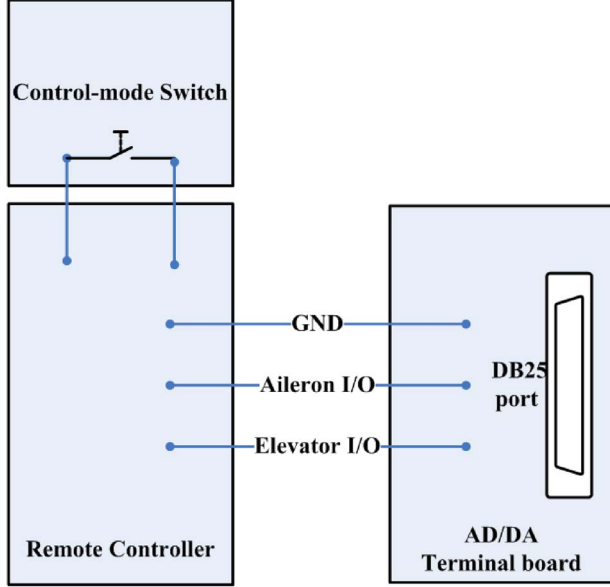


Fig. 7. Modification of the remote controller.

TABLE I
CONVERSION TABLE

Parameters	Range	Conversion of the parameter and the voltage V (volts)
Roll angle ϕ (degrees)	$-90^\circ < \phi < 90^\circ$	$V(\phi) = 0.0278\phi + 2.5$
Pitch distance η (pixels)	$-120 \leq \eta \leq 120$	$V(\eta) = 0.0208\eta + 2.5$
Servo angle ω (degrees)	$-52^\circ \leq \omega \leq 52^\circ$	$V(\omega) = 0.0481\omega + 2.5$

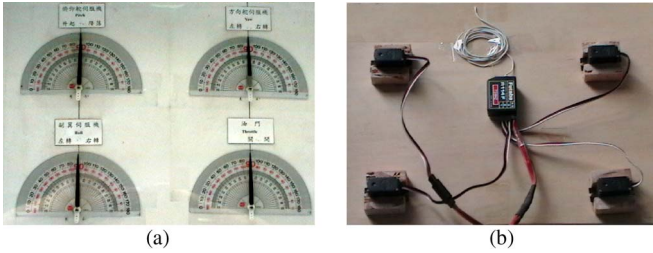


Fig. 8. Homemade servo system. (a) Front side. (b) Back side.

signals from the remote controller, conversion between servo angles and the input voltages of the remote controller is also necessary and is included in Table I. Table I lists the conversions between the flight parameters (including the roll angle, the pitch distance, and the servo angle) and the voltage signals of the remote controller.

A homemade servo system that consists of four protractors and four servos (as shown in Fig. 8) is employed to verify that the modification of the remote controller is functional and the conversions between the parameters and the controller voltage are appropriate.

To enhance the stability of the UAV, a proportional–integral (PI) controller is included in the proposed flight control system, as shown in Fig. 9.

In Fig. 9, the parameter $y_{sp}(m)$, $m = \phi$, or η is the set point of the desired roll angle or pitch distance, $y^j(m)$ is the detected

roll angle or pitch distance of the j th frame, and $V_o^j(m)$ is the output voltage of the j th frame to the remote controller. To stabilize the airplane, $y_{sp}(\phi)$ and $y_{sp}(\eta)$ are set at 0° and zero pixels, respectively. The parameters $V^j(m)$ and $V_{sp}(m)$ are the voltages that correspond to $y^j(m)$ and $y_{sp}(m)$, respectively. We propose a simple PI controller, which is described by the following equation:

$$V_o^j(m) = K_P(m) \cdot \Delta V^j(m) + K_I(m) \cdot \sum_{i=j-29}^j \Delta V^i(m) + V_{OS}(m) \quad (30)$$

where $K_P(m)$ is the proportional gain, $K_I(m)$ is the integral gain, $\Delta V^j(m)$ is the measured difference given by $\Delta V^j(m) = V_{sp}(m) - V^j(m)$, and $V_{OS}(m)$ is the offset voltage of the servo. In this paper, we adopt a trial-and-error approach to adjust the gains for the desired response, because there is no precise mathematical model of the flight dynamics of a small UAV. Using the results of several test flights, the gains K_P and K_I are set at 1 and 0.07, respectively.

Because the detection rate is around 30 Hz, the variation of the detected skyline between two successive images is small. To avoid a crash that is caused by erroneous detection of the skyline, the effective ranges of the roll angle and the pitch distance for two successive images are set within $\pm 20^\circ$ and ± 40 pixels, respectively. When a detected value violates these limitations, the previous value is substituted for the detected value.

B. Skyline-Detection Results

Before actual test flights are conducted, five test videos are recorded under varying weather and illumination conditions to evaluate the accuracy of the proposed algorithm. Test videos 1–3 are captured with an onboard camera, and test videos 4 and 5 are downloaded from the websites [http://vimeo.com/2014108 and http://vimeo.com/9314939] to enrich the test sample. The skyline-detection accuracy for a test video is defined as

$$\text{Accuracy} = \frac{N_C}{N_T} \times 100\% \quad (31)$$

where N_C is the number of correctly detected frames, and N_T is the total number of frames in the video. By definition, the skyline is correctly detected when the deviation between the terminal points that are detected by the proposed algorithm and the human eye is less than five pixels. Table II shows the test results of the initial and tracking algorithms. Based on the test results summarized in Table II, the average accuracy rate of the tracking algorithm is 98.62%.

To continue the discussion, factors that affect the accuracy are divided into the following four categories:

- 1) the shape of skyline, e.g., straight or uneven;
- 2) varying illumination, e.g., sunrise, sunny, and overcast days;
- 3) complex textures, e.g., clouds, buildings, and roads;

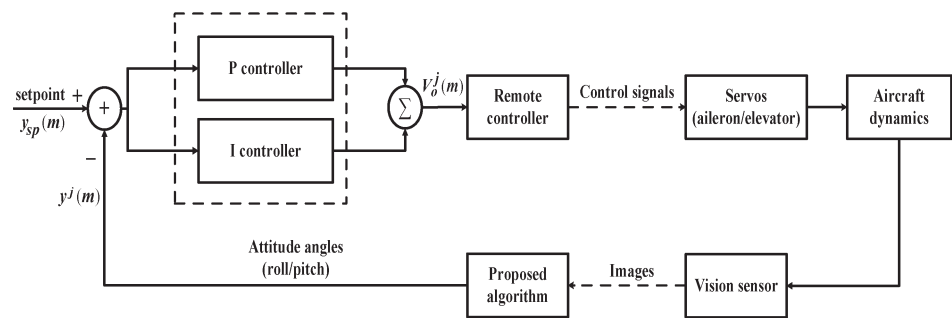







Fig. 9. PI control block diagram for the proposed system.

TABLE II
ACCURACY FOR THE TEST VIDEOS

Videos (weather)	Description (H:High,M:Medium,L:Low)				Total frames	Accuracy	
	Brightness	Contrast	Texture	Noise		Initial algorithm	Tracking algorithm
1(sunny) 	M	H	L	L	15231	97.2%	100%
2(sunny) 	M	H	M	M	17694	90.4%	98.3%
3(cloudy) 	L	M	M	M	18347	86.2%	97.8%
4(cloudy) 	M	M	H	L	5998	87.5%	98.8%
5(cloudy) 	H	H	H	L	5821	93.3%	98.4%

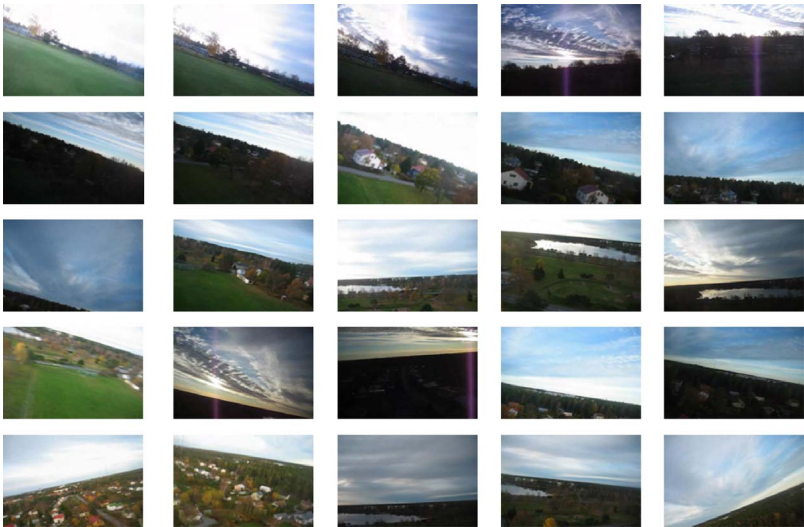


Fig. 10. Images that were sampled from test video 4.

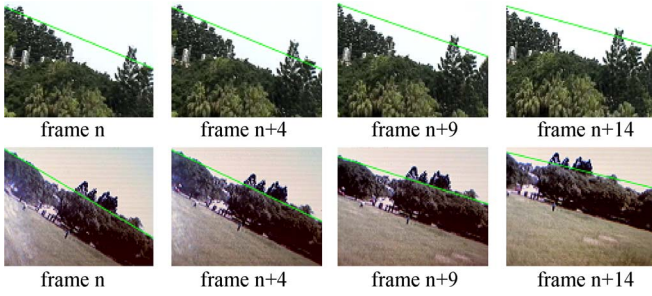


Fig. 11. Skyline detection with varying skyline geometries.

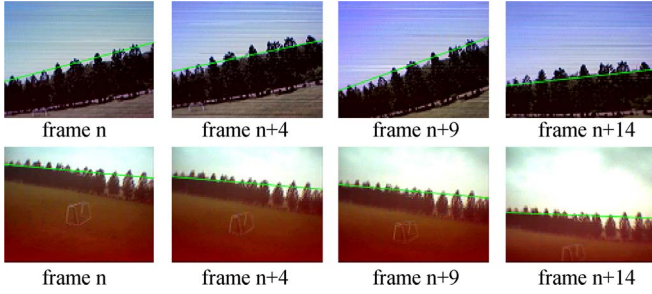


Fig. 12. Skyline detection with varying illumination.

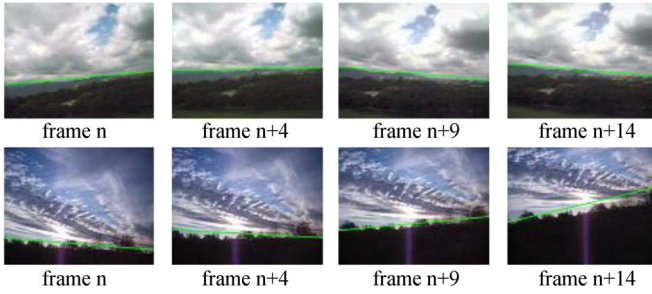


Fig. 13. Skyline detection with varying cloud cover.

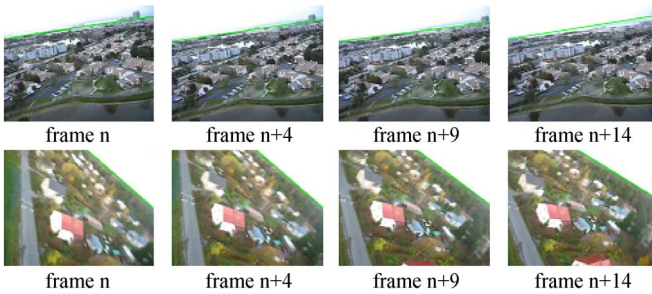


Fig. 14. Skyline detection with varying ground objects.

- 4) noise interference that results from flight vibrations and electromagnetic interference.

In Table II, the accuracy of the tracking results is better than the results of the initial algorithm, because the detection results of the initial algorithm are influenced by the noise interferences and complex textures, e.g., clouds, buildings, and roads. In addition, the ratio of the images with varying cloud cover is around 0.04% (266/5998) in test video 4. Hence, the average accuracy of test video 4 is 87.5% processed by the initial algorithm. To provide more details of test video 4, some images with different views are shown in Fig. 10.

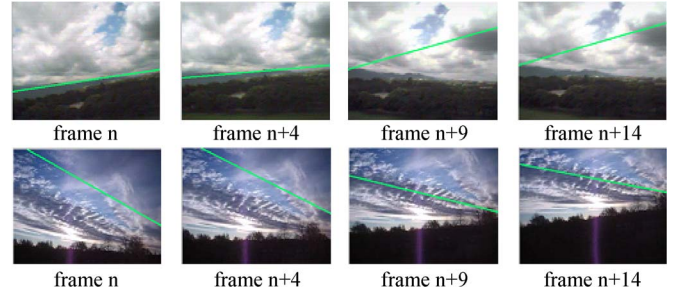


Fig. 15. Detection results of the original images in Fig. 13 by the initial algorithm.

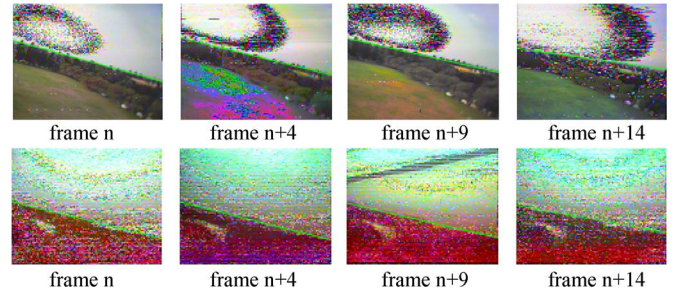


Fig. 16. Skyline detection with varying noise.

TABLE III
PROCESSING TIME FOR EACH STEP OF THE PROPOSED ALGORITHM

Stage	Average time (milliseconds)
Image acquisition	29.7
Initial detection	2.1
Tracking detection	0.7
Attitude computation	0.002
PI control	0.04
Voltage output	0.8
Total	33.342

TABLE IV
PROCESSING TIMES OF DIFFERENT SKYLINE-DETECTION ALGORITHMS

Algorithms	Processor	Frame size	Time (ms)
Proposed method	P IV, 2.6GHz	320 × 240	0.7
Propeller removal[14]	P IV, 2.4GHz	320 × 240	25
Pixel-statistical[19]	P III, 750MHz	320 × 240	30
MLDA[20]	P IV, 3.33GHz	128 × 128	33
Edge-based Hough[16]	P IV, 900MHz	352 × 288	67
Projection[12]	P III, 3.0GHz	352 × 288	200

The following figures show the test results of the tracking algorithm when the initial skyline is correct. Fig. 11 shows that both straight and uneven skylines can successfully be detected. Thus, it is clear that geometric variations due to altitude changes will not hinder skyline detection. Moreover, although the number of edge points of a skyline may change under different lighting conditions, the terminal points can still be located (as Fig. 12 indicates), as long as the skyline is distinguishable by visual inspection.

Given that skylines are detected from edge features, the proposed algorithm is sensitive to complex textures in images,

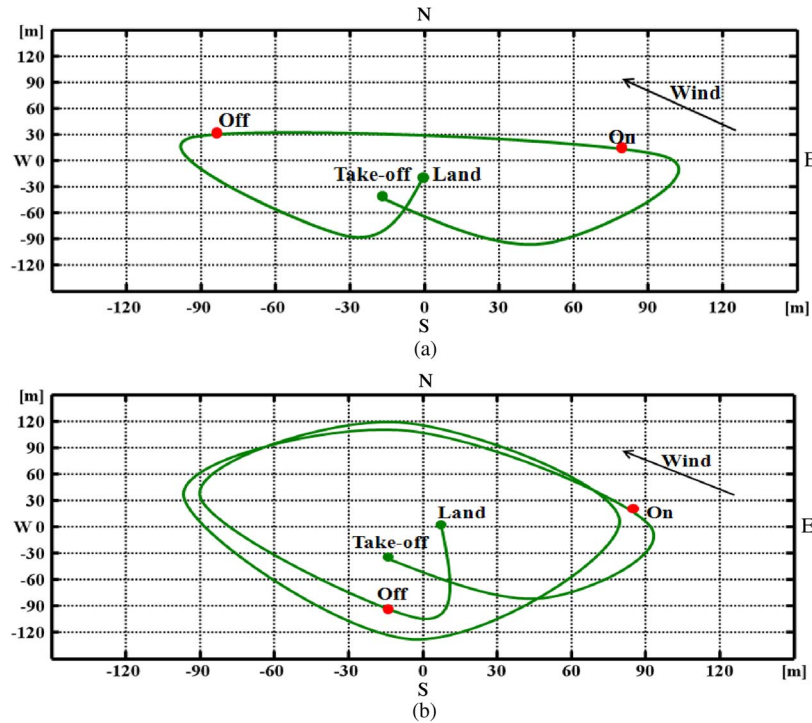


Fig. 17. Trajectory of the test flight. (a) Straight flight path. (b) Circular flight path.

particularly when the gradient value of a background image is greater than the gradient value of a terminal point of the skyline. Some detection errors may result from clouds in the sky or buildings on the ground. Nevertheless, most edge textures in the image will not influence the detection results, because the ROI is confined to a local area. Therefore, the positions of erroneous detections caused by complex textures are generally very close to the correct locations of the terminal points, and errors can be corrected within a few frames, as shown in Figs. 13 and 14. To show the advantage of the tracking algorithm, the original images in Fig. 13 are detected by the initial algorithm. The detection results are shown in Fig. 15.

Noise-corrupted images may cause serious errors in vision-based methods. Fortunately, compared with other global detection methods, the proposed algorithm restricts the errors to a limited area that surrounds the previously detected terminal points. Thus, flight attitude may not dramatically change due to error detection. Moreover, scene sequences may rapidly change with forward flight velocity. Using the tracking algorithm, the errors may also be recovered in a few frames, as shown in Fig. 16.

For performance analysis, Table III lists the required processing time for every step of the proposed algorithm, which is coded in Microsoft Visual C++ 2005 and operated with an image size of 320×240 pixels on a Pentium IV 2.6-GHz processor with 1 GB of memory. In the tracking stage, the total processing time from image acquisition to voltage output is around 31.2 ms.

Considering only skyline-detection algorithms, Table IV presents a performance comparison of the present technique and various published studies on visual skyline detection. Clearly, the proposed algorithm greatly reduces the computational burden through its local processing strategy.

C. Results of the Test Flight

After the series of ground tests, a test flight is conducted to evaluate the lateral and longitudinal stability of the proposed vision-based flight control system in the real world. At the beginning of the test, the UAV is hand launched and remotely controlled in the manual mode by an experienced pilot until it reaches a specific altitude of about 60 m above the ground. Once the desired altitude and the correct initial skyline have been attained, the mode is switched to automatic, and the control of the ailerons and elevator is handed over to the GCC (although the rudder and throttle are still operated by the pilot). Before landing, the control mode is switched back to the manual mode, and the UAV is again controlled by the pilot. A GlobalSat DG-100, a Global Positioning System (GPS) device that weighs 115 g, is carried onboard to record the trajectory during the test flight.

To demonstrate lateral and longitudinal stability, the following two test scenarios are provided: 1) straight flight and 2) circular flight. Fig. 17 shows a GPS plot of the flight path for the two scenarios. The axes denote the north–south (N/S) and east–west (E/W) cardinal directions and are delineated in meters.

During the autonomous flight, the airspeed is about $18 \sim 20$ m/s, with a wind blowing from the southeast at about $2 \sim 4$ m/s. For the straight flight, the rudder and throttle settings are held constant. Confined to the limitations of the testing field, the maximum distance covered in a straight flight is around 160 m. The variations in the roll angle and pitch distance are shown in Fig. 18.

According to the experimental results shown in Fig. 18(a) and (b), the variations of the roll angle and pitch distance during the autonomous straight flight are within $\pm 5^\circ$ and ± 10 pixels,

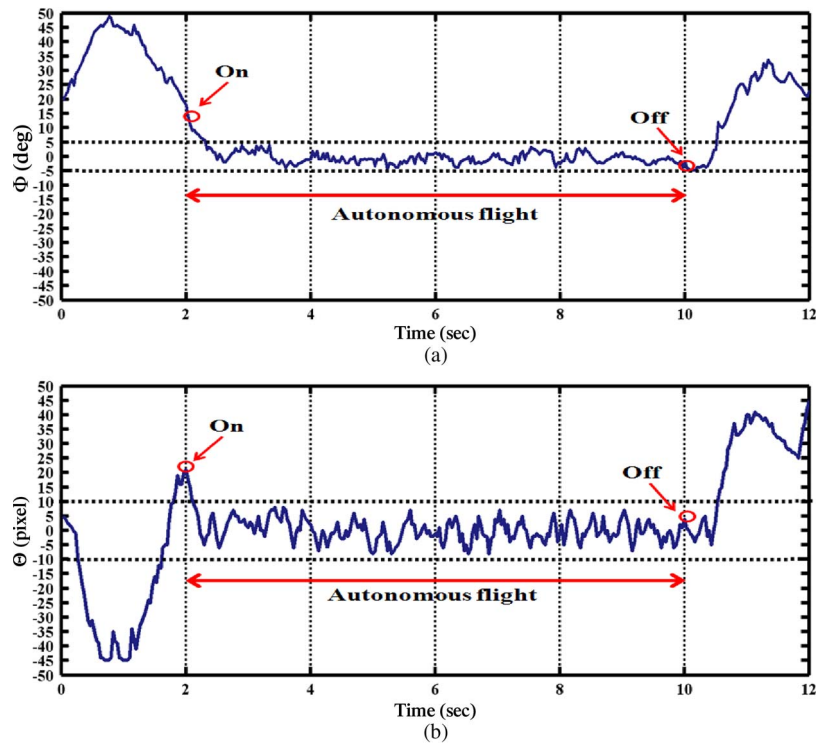


Fig. 18. Variations of the roll angle and pitch distance in the straight flight. (a) Roll angle. (b) Pitch distance.

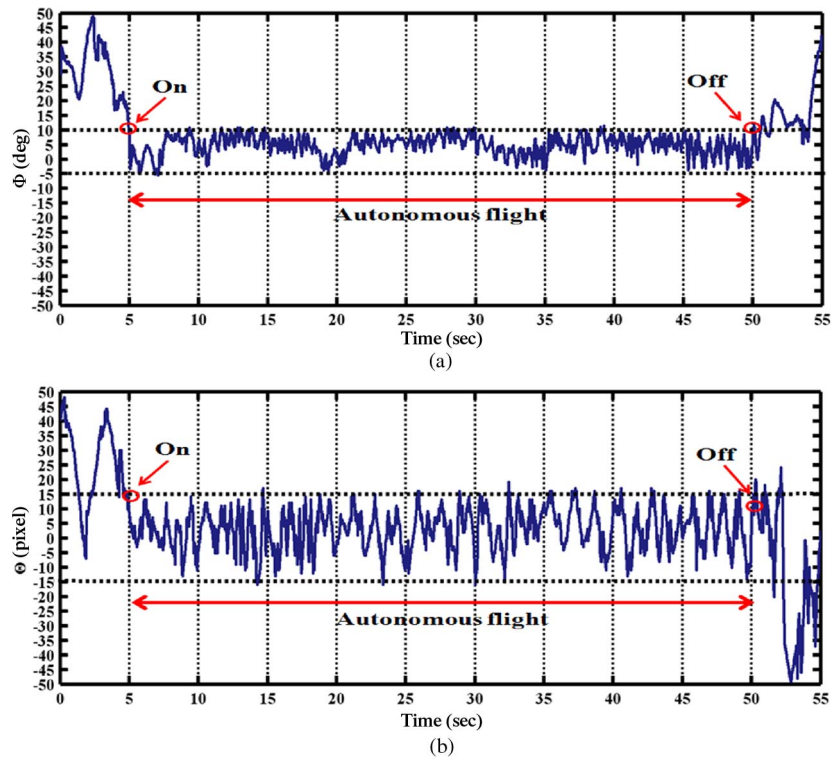


Fig. 19. Variations of the roll angle and pitch distance in the circular flight. (a) Roll angle. (b) Pitch distance.

respectively. Thus, stability during the straight flight is clearly validated.

Fig. 19 shows the variations of the roll angle and pitch distance during the circular flight. For the circular flight, the rudder and throttle are manually controlled by the pilot.

Due to yaw-roll coupling, roll offset occurs during the rudder-only turning flight, as shown in Fig. 19(a). Although the Dutch roll exists, the oscillation of the roll angle is between -5° and $+10^\circ$. The variation of the pitch distance is shown in Fig. 19(b). The oscillation of the pitch distance is within ± 15

pixels. Lateral and longitudinal stability can be inferred from this test flight.

Referring to Figs. 18 and 19, the roll angle and pitch distance can be shifted toward the set points for the desired roll angle and pitch distance a short time after switching to the automatic mode.

V. CONCLUSION

This paper has introduced a vision-based flight control system for small UAVs using a skyline-detection algorithm. The system integrates a remote controller, a remotely controlled airplane, a camera, a wireless transmitter/receiver, a GCC, and the proposed skyline-detection algorithm to achieve automatic control of flight stability. As the experimental results indicate, this low-cost flight control system offers the advantages of real-time constraint, robustness to noise, and reliable detection. Furthermore, the proposed skyline-detection algorithm can detect both straight and uneven skylines, with an average accuracy rate of 98.62%, based on the five test videos. The flight test results demonstrate that the system controls and stabilizes a UAV with vision-only flight control. The limitation of the proposed algorithm is that the skyline must remain in the image frame at all times. In future research, we will add rudder and throttle controls and transmit the GPS signal to the GCC to fully control the automatic flight of a UAV.

REFERENCES

- [1] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 340–345.
- [2] D. B. Kingston and R. W. Beard, "Real-time attitude and position estimation for small UAVs using low-cost sensors," presented at the AIAA 3rd Unmanned Unlimited Technical Conf. Workshop, Chicago, IL, Sep. 2004 Paper AIAA 2004-6488.
- [3] L. Wang, S. Xiong, Z. Zhou, Q. Wei, and J. Lan, "Constrained filtering method for MAV attitude determination," in *Proc. IEEE Instrum. Meas. Technol. Conf.*, Ottawa, ON, Canada, May 2005, vol. 2, pp. 1480–1483.
- [4] H. Chao, Y. Cao, and Y. Q. Chen, "Autopilots for small fixed-wing unmanned air vehicles: A survey," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Aug. 2007, pp. 3144–3149.
- [5] J. S. Jang and D. Liccardo, "Automation of small UAVs using a low-cost MEMS sensor and embedded computing platform," in *Proc. IEEE/AIAA 25th Digital Avionics Syst. Conf.*, Oct. 2006, pp. 1–9.
- [6] O. J. Woodman, "An introduction to inertial navigation," Univ. Cambridge, Cambridge, U.K., Tech. Rep. 696, Aug. 2007.
- [7] A. A. Proctor, E. N. Johnson, and T. B. Apker, "Vision-only control and guidance for aircraft," *J. Field Robot.*, vol. 23, no. 10, pp. 863–890, 2006.
- [8] C. Wagter, A. Proctor, and E. Johnson, "Vision only aircraft flight control," in *Proc. Digital Avionics Syst. Conf.*, Oct. 2003, vol. 2, pp. 2–11.
- [9] C. D. Wagter, B. Bijmens, and J. A. Mulder, "Vision-only control of a flapping MAV on Mars," presented at the AIAA Guidance, Navigation Control Conf. Exh., Hilton Head, SC, Aug. 2007 Paper AIAA 2007-6853.
- [10] S. G. Fowers, D. J. Lee, B. J. Tippetts, K. D. Lillywhite, A. W. Dennis, and J. K. Archibald, "Vision aided stabilization and the development of a quad-rotor micro UAV," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, Jacksonville, FL, Jun. 2007, pp. 143–148.
- [11] T. P. Webb, R. J. Prazenica, A. J. Kurdila, and R. Lind, "Vision-based state estimation for autonomous micro air vehicles," *J. Guid. Control Dyn.*, vol. 30, no. 3, pp. 816–826, 2007.
- [12] G. Q. Bao, S. S. Xiong, and Z. Y. Zhou, "Vision-based horizon extraction for micro air vehicle flight control," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 3, pp. 1067–1072, Jun. 2005.
- [13] T. D. Cornall, G. K. Egan, and A. Price, "Aircraft attitude estimation from horizon video," *Electron. Lett.*, vol. 42, no. 13, pp. 744–745, Jun. 2006.
- [14] G. A. S. Pereira, P. Iscold, and L. A. B. Torres, "Airplane attitude estimation using computer vision: Simple method and actual experiments," *Electron. Lett.*, vol. 44, no. 22, pp. 1303–1304, Oct. 2008.
- [15] S. Fefilatyev, V. Smarodzinava, L. O. Hall, and D. B. Goldgof, "Horizon detection using machine learning techniques," in *Proc. 5th Int. Conf. Mach. Learn. Appl.*, Dec. 2006, pp. 17–21.
- [16] D. Dusha, W. Boles, and R. Walker, "Fixed-wing attitude estimation using computer-vision-based horizon detection," in *Proc. 12th Aust. Int. Aerosp. Congr.*, Melbourne, Australia, Apr. 2007, pp. 1–19.
- [17] H. Z. Yuan, X. Q. Zhang, and Z. L. Feng, "Horizon detection in foggy aerial images," in *Proc. IEEE Int. Conf. Image Anal. Signal Process.*, Zhejiang, China, Apr. 2010, pp. 191–194.
- [18] W. N. Lie, T. C. I. Lin, T. C. Lin, and K. S. Hung, "A robust dynamic programming algorithm to extract skyline in images for navigation," *Pattern Recognit. Lett.*, vol. 26, no. 2, pp. 221–230, Jan. 2005.
- [19] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Vision-guided flight stability and control for micro air vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, Oct. 2002, pp. 2134–2140.
- [20] S. Todorovic and M. C. Nechyba, "A vision system for intelligent mission profiles of micro air vehicles," *IEEE Trans. Veh. Technol.*, vol. 53, no. 6, pp. 1713–1725, Nov. 2004.
- [21] T. G. McGee, R. Sengupta, and K. Hedrick, "Obstacle detection for small autonomous aircraft using sky segmentation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 4679–4684.
- [22] Y. F. Dong, Y. F. Zhang, C. Zhu, and A. B. Wang, "Extracting sea-skyline based on improved local complexity," in *Proc. IEEE Int. Conf. Comput., Mechatronics, Control Electron. Eng.*, Aug. 2010, pp. 82–85.



Chung-Cheng Chiu (M'05) received the B.S. and M.S. degrees in electrical engineering from Chung Cheng Institute of Technology–National Defense University, Taoyuan, Taiwan, in 1990 and 1993, respectively, and the Ph.D. degree from the National Chiao Tung University, Hsinchu, Taiwan, in 2004.

Since 1993, he has been a Lecturer with the Department of Electrical and Electronics Engineering, Chung Cheng Institute of Technology–National Defense University, where he became an Associate Professor in 2005. His research interests include image

recognition, image compression, document segmentation, and computer vision and their applications to intelligent transportation systems.

Dr. Chiu is the recipient of the 2003 Dragon Golden Paper Award from the Acer Foundation and the Silver Award of Technology Innovation Competition from AdvanTech.



Ching-Tung Lo (S'11) received the B.S. and M.S. degrees in computer science and electrical engineering in 1997 and 2004, respectively, from Chung Cheng Institute of Technology–National Defense University, Taoyuan, Taiwan, where he is currently working toward the Ph.D. degree in electrical engineering with the Department of Electrical and Electronics Engineering.

His research interests include computer vision and image recognition and their applications to vision-based flight control systems.