IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 30, 2017, accepted July 3, 2017, date of publication July 19, 2017, date of current version November 14, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2728826

A Robust Vision-Based Skyline Detection Algorithm Under Different Weather Conditions

YUN-JIUN LIU, (Student Member, IEEE), CHUNG-CHENG CHIU, (Member, IEEE), AND JIA-HORNG YANG, (Member, IEEE)

Department of Electrical and Electronics Engineering, Chung Cheng Institute of Technology, National Defense University, Taoyuan 335, Taiwan

Corresponding author: Yun-Jiun Liu (rodliu0000@gmail.com)

ABSTRACT The skyline is a piece of important reference information in the automatic flight control system of an unmanned vehicle. Currently, most skyline detection algorithms assume that the skyline has distinct edge features that can be located in the image by edge detection. This method of detection, however, is only applicable when the skyline is obvious. Its detection result in images with an indistinct skyline is affected by the threshold value, such as on cloudy or heavily foggy days, when clear skyline features cannot be found. To find both distinct and indistinct skylines in the images, this paper proposes a vision-based skyline detection algorithm, in which the skyline is located by analyzing image brightness variations. This method is able to identify nonlinear skyline profiles in scenes with a clear skyline, and find the interface region between the sky and earth in scenes with an indistinct skyline, to estimate its location. According to the experimental results, the algorithm correctly finds the skyline profile in 97% of the test images, making it ideal to provide skyline reference information for the automatic flight control systems of unmanned vehicles.

INDEX TERMS Skyline, different weather and environmental conditions, real-time processing, skyline profile.

I. INTRODUCTION

In recent years, the development of unmanned aircrafts has received an increasing amount of attention. In particular, an automatic flight control system based on machine vision is a key project. When a camera is mounted in front of the aircraft, the captured image is equivalent to the first person perspective of the pilot. The flight control system can obtain environment information through image analysis, thus controlling the action of the aircraft, the same way the pilot uses vision to judge the current flight situation and control the aircraft, and to perform tasks such as aircraft stabilization, obstacle avoidance, and landing at the specified location.

The skyline provides important, robust reference information in visual flight control systems, and the acquisition of its information through image analysis is a critical step. The skyline can be used to determine whether an object in an image is located on the ground or in the sky. With the camera fixed on the aircraft in flight and the horizontal axis of its coordinates set parallel to the wing, the roll angle of the aircraft relative to the ground can be determined from the gradient of the skyline in the image. Through the intersection of the skyline and the vertical axis at the center of the image, the pitch angle of the aircraft can also be determined. This information can assist the flight control system to adjust the attitude of the aircraft and maintain its stability.

II. RELATED WORK

Among the methods to detect the skyline, some scientists use the difference between the sky and the ground. For example, Cornall *et al.* [1] thought that among the primary colors (RGB), blue would be suitable for distinguishing the sky from the ground. They firstly binarized the image with a threshold that emphasized the color blue to extract the boundary between the sky and ground regions, and then used Hough Transformation (HT) to obtain the skyline formed by the boundary. Using blue to distinguish sky from earth is practical when the weather is fair, because we can expect the sky to be blue then. However, on cloudy or rainy days, the sky is not necessarily blue, and using color information may not be sufficient to discern the difference between the sky and the ground. To enable the algorithm to find the skyline in cloudy or rainy days, scientists have switched to gather of pixel intensity, i.e. locating the skyline with the change in pixel intensity at the horizon line. Ettinger *et al.* [2] for example, performed a statistical analysis of pixel color distribution and found the skyline using a position of

maximum color intensity variation between the upper and lower portions of the image. Dong *et al.* [3] calculated the maximum grayscale complexity for an image region and multiplied it by a parameter which is less than one to obtain a threshold for image binarization. With this, they identified regions with relatively high complexity, such as the sea-to-sky convergence line, and used HT to find the skyline. Yuan *et al.* [4] proposed a technique for removing clouds and fogs, which is used to detect the skyline in images that appear cloudy. These methods are useful for the detection of the skyline when weather is not fair, but the statistical processing of pixel intensity takes a considerable amount of time to achieve real-time image processing. As the flight control system needs to monitor the flight continuously, a longer image analysis time leaves less reaction time for the system. The algorithm needs to provide timely information on skyline detection for the flight control system to have enough of a reaction time to maintain aircraft stability.

Other scientists have also attempted to use the classifier to discern skyline features. For instance, Fefilatyev *et al.* [5] used the classifier to divide the image pixels into sky and ground regions, locating their boundary, and applied HT to find the skyline formed by the boundary. McGee *et al.* [6] used a support vector machine (SVM) to classify the image pixels into sky and ground categories and binarized the image based on the classification results. The black/white interface after binarization could represent the skyline boundary, for which the Hough Transformation could be applied to obtain the skyline detection results. As skyline scenes need to be selected to train the classifier model prior to its use, the classifier will have a high accuracy in resolving skyline in images similar to the training scenes. However, as the characteristics of the scene are changed, the accuracy of the classifier will be affected. As the scene will definitely change during the flight of the aircraft, the algorithm must adapt to a variety of scenarios to meet the requirements of the flight control system.

Another, more intuitive method is to use the edge features of the skyline to search the skyline features. When the boundary between the sky and ground is very clear, the skyline will have strong edge features in the image. Bao *et al.* [7] used edge detection to look for the edge points in the image and calculated the cumulative projected values of the edge points in different directions to find the direction of the maximum cumulative value and the corresponding straight line as the skyline. Pereira *et al.* [8] used edge features to find the position of the skyline and the vibrating aircraft wing and analyzed the characteristics of continuous images to remove the edge features of the vibrating wing to eventually locate the skyline using its edge features. Dusha *et al.* [9] used morphology to find the edge points for the three channels of RGB. After integrating these three sets of edge points, he then used the Hough Transform to detect the long straight-line segments in the image, and finally filtered out the skyline with the optical flow method. As the processing speed for edge detection is very fast nowadays, the skyline can be detected quickly using edge features in most cases when it is distinct. However, when skyline edge features are not so clear, the edge detection threshold will directly affect the skyline detection results. A threshold that is too high cannot be used to find the edge points belonging to the skyline, while a value that is too low will accept a lot of noise that is not the skyline. As such, using the edge to determine the skyline, the algorithm may not be able to identify the right target when it is ambiguous in the image.

In all of the above studies, it is assumed that a long straight line in the image depicts the skyline. The feature points of the skyline are first identified, and then the skyline position is located using linear transformation. However, when objects such as buildings, mountains, and trees are present in the skyline, its profile will be a curve. The results of linear transformation therefore cannot show the contour of the skyline. To determine the skyline with mountains, woods, or buildings in the scene, Lie *et al.* [10] proposed connecting adjacent edge points in the image with dynamic programming to obtain the skyline. However, for skyline feature search using edge detection, a less-than-ideal representation of the skyline may be obtained when the weather changes, because non-skyline edge points will affect connection results. Chiu and Lo [11] found two peripheral blocks of the image with the largest variance and treated their connection line as the skyline. However, when the skyline was not distinct, the variance for the blocks containing the skyline was not great enough, and the skyline could not be accurately identified. To make the flight control system more stable, the algorithm must be applicable to a non-linear skyline.

To meet the above requirements, the algorithm proposed in this paper includes the steps to speed up processing. For an input image of fixed size, the algorithm can adjust the average processing speed to meet the demand of real-time computation. The edge detection threshold is avoided during the search of the skyline, and an analysis of the gradient change from the top to bottom of the image is instead used. This enables the algorithm to be applied to images of various brightness and contrast. The detected skyline will be close to the actual skyline profile, in that the curves of mountains and woods can be fully represented. The algorithm can rapidly find the skyline in the image under different scenarios and weather.

III. SKYLINE DETECTION ALGORITHM

The skyline detection algorithm proposed in this paper consists of four steps: namely pre-processing, selecting skyline candidate points, skyline candidate points filtering, and skyline candidate points connecting. Fig. 1 is the flowchart of the proposed algorithm. The first step is pre-processing, which is mainly the reduction of the computational workload of the algorithm to speed up the processing speed, so the algorithm can meet the demands of real-time computation. When the camera is fixed on an unmanned aircraft, the skyline in the image can be used to determine the attitude change of the aircraft. For a timely transmission of the skyline detection

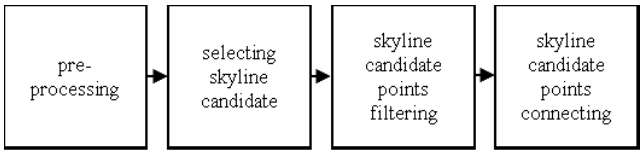


FIGURE 1. The flowchart of the proposed algorithm.

results to the flight control system, the processing speed of the algorithm must be less than the time interval of image input. In the NTSC standard, the speed of the algorithm to process a single image must be less than 33 milliseconds to fulfill real-time computing needs.

The second step is to find the skyline candidate points, with the aim to locate the possible position of the skyline in the image. When detecting the skyline, effects of weather and scene must be considered. In general, when the weather is fair and the sky is free of clouds, prominent edge features will exist at the boundary of sky and earth, and edge detection can be used to locate the skyline. However, in cloudy, rainy, or foggy days, the skyline will become less obvious, and the threshold in edge detection will affect the skyline search results. A threshold that is too small will reduce the number of the edge points identified, and one that is too large will pick up too much non-skyline noise. If a better edge detection result is desired, we can dynamically calculate the threshold for edge detection by analyzing the characteristics of the image, when the image property for the whole skyline is consistent. However, when skyline regions with different image properties are present in the image, the dynamically calculated threshold cannot be applied to the whole image. For instance, if the image is located at the intersection of mountain and sea, the skyline between the mountain and sky is clear, but between the sea and sky is not so clear. A single edge detection threshold can only receive good detection results for a portion of the skyline region, and cannot be used for the whole skyline region. Even with thick clouds, when only a gradual change from bright to dark is seen between the sky and ground, the boundary is not obvious. There are no edge features at the boundary of the sky and the earth, and edge detection cannot be used to search for the skyline. From here, one can see that finding the skyline features during different weather and scenes is a complex issue, and no method previously existed could solve all of the above situations. As such, this paper presents a method to detect the skyline features for different weather and scenes without the need to set the edge detection threshold. This method can automatically find the skyline candidate position in the image.

The third step involves skyline candidate points filtering. The skyline candidate points found are evaluated to filter out the absurd ones. When collecting skyline candidate points, only the continuity of adjacent pixels in the vertical direction is considered for maximum inclusion of all possible skyline points, and the relationship between the image regions is omitted from the analysis. As such, this step will perform the extraction according to the characteristics of the sampling column of the candidate point, as well as its nearby regions, to preserve reasonable candidate points for subsequent analysis.

The last step is the connecting of skyline candidate points. The similarity between the sky and ground regions of adjacent candidate points is compared, and similar points are connected to infer the skyline. When non-straight line segments such as mountains and woods are present in the skyline of the image, the algorithm hereby proposed can present complex skyline profiles.

A. PREPROCESSING

There are two parts in pre-processing: selecting the sampling column and establishing the integral image. Before the detection of the skyline, the algorithm will select sampling columns at fixed intervals in the horizontal direction of the image to use as the analysis units for skyline detection. Each sampling column is a vertical column of pixels. In an image $f(x, y)$ with width w and height h, assuming the distance between neighboring sampling columns to be p, where p is in the range of $[1, w-1]$, under the condition that p is known, the x-coordinate of the pixels in the m-th sampling column can be expressed as $m \cdot p$, with m in the range of $[0, (w-1)/p]$. A greater distance p between the sampling columns results in fewer sampling columns. In the case of a pre-determined p and m, a column of pixels in the image can be represented by Equation (1):

When the skyline is detected for image $f(x, y)$, the processing speed of the skyline detection algorithm and the resolution of the skyline profile can be adjusted by the size of p. As the image width is fixed, a larger value of p means less sampling columns to be analyzed and a faster algorithm processing speed. On the other hand, skyline detection is based on the connections between the candidate points of the sampling columns. The number of sampling columns determines the resolution of the skyline profile. A larger p value therefore yields a skyline with a lower resolution in the algorithm. Taking two extreme examples: (1) at $p = 1$, the whole image is the scope of the analysis, so the processing time is the longest and the resolution of the skyline profile is the best; (2) at $p = w - 1$, sampling only occurs in the two columns of pixels to the leftmost and rightmost of the image. The left and right endpoints are used to determine the skyline. As such, the processing time will be the shortest, but the skyline profile will have the lowest resolution. In practical, present-day applications, the value of p will be adjusted according to the procession time of the skyline detection algorithm and requirements on the skyline profile resolution, which will be discussed in more detail in the experimental results section.

$$f(x, y) = f(m \cdot p, y), \quad y = 0, 1, 2 \ldots, h-1 \qquad (1)$$

The proposed algorithm only analyzes the pixels in the sampling column during its search for the skyline. As the purpose of finding the skyline is to obtain its overall trend of change in the image, and the skyline is a continuous curve,

reducing the horizontal resolution of the detection result will not affect the skyline profile as a whole. This algorithm analyzes the position of the skyline in the sampling column to preserve the vertical resolution of the detection result. The rapid acquisition of the skyline profile, by connecting its position in each sampling column, allows for the evaluation of trending skyline change and a reduction in the amount of data to be processed, thereby speeding up the algorithm.

Another part in pre-processing is to establish the integral image. As the algorithm needs to repeatedly calculate the regional intensity sum of the sampling column pixels in the vertical direction, the one-dimensional integral image for sampling column pixels in the vertical direction will first be established to accelerate the calculation speed. The integral image approach is the point-by-point addition of the grayscale intensity of a column of pixels from the top to bottom, with storage in a register. The integral image of each pixel represents the accumulated grayscale intensities of the above vertically pixels, plus that of the present pixel. In the m-th sampling column, $I(m \cdot p, y)$ is the integral image built on the sampling column pixel $f(m \cdot p, y)$, as shown in Equation (2). The range of y is $[0, h-1]$.

$$I(m \cdot p, y) = \sum_{k=0}^{y} f(m \cdot p, k), \quad k \in \mathrm{N} \tag{2}$$

When the algorithm needs to calculate the pixel sum for a section of arbitrary length in a column, the result can be obtained rapidly by subtracting the integral image value of that column. This greatly accelerates the processing speed of the algorithm. In Equation (3), if the sum from the $(r+1)$-th pixel to the s-th pixel in the sampling column m needs to be calculated, the subtraction of integral image $I(m \cdot p, r)$ from $I(m \cdot p, s)$ could be performed to quickly obtain the intensity sum of the pixel region.

$$\sum_{k=r+1}^{s} f(p \cdot m, k) = I(m \cdot p, s) - I(m \cdot p, r), \quad s > r, s \in \mathrm{N}, r \in \mathrm{N} \tag{3}$$

As can be seen in Equation (3), the calculation of the accumulated sum from the $(r+1)$-th pixel to the s-th pixel originally needs $s - r - 1$ addition operations, but through the integral image, only a subtraction is needed to get the same result. This speeds up the computation of the regional intensity sum of pixels in the vertical direction.

B. SELECTING SKYLINE CANDIDATE POINTS

The selection of skyline candidate points is divided into two steps: The first is to find the possible region containing the skyline by analyzing the change in the gradient of intensity of sampling column pixels. We call the region as the skyline candidate region. The second step is to find the candidate point of the skyline from the candidate region. At the same time, considering the influence of noise, methods to eliminate noise impact will be added to the above two steps, so that the overall algorithm is more robust.

In the first step, to analyze the change in the gradient of intensity of the sampling column, the vertical gradient will first be calculated for sample column pixels. The pixel gradient of the m-th sampling column in the image is defined here as $G_m(y)$, as shown in Equation (4):

$$G_m(y) = f(m \cdot p, y-1) - f(m \cdot p, y+1), \quad y = 1, 2, \ldots, h-2 \tag{4}$$

After calculating the vertical gradient of the sample column pixels, the property of "bright above and dim below" is used to determine a possible region for the skyline. A reasonable skyline has sky and clouds above it and earth or sea below, and as such, the pixel intensity near the skyline will appear brighter above and dimmer below. The calculated $G_m(y)$ will be greater than zero. Pixels in the sampling column can be divided into two types with the calculated result of Equation (4). The one greater than zero indicates that the area above that pixel is brighter and below it is darker. This pixel may be part of the skyline region. The one less than or equal to zero indicates that the pixel does not meet the criterion of being bright above and dark below, and is not inside the skyline region. To distinguish between these two types of pixels, the brightness analysis function of the sample column m is defined as a two-valued function, as shown in Equation (5):

$$\begin{cases} g_m(y) = 1, \textit{when} \quad G_m(y) > 0 \\ g_m(y) = 0, \textit{when} \quad G_m(y) \le 0, \end{cases} \quad y = 1, 2, \ldots, h-2 \tag{5}$$

With the brightness analysis function in Equation (5), possible skyline regions can be found in the sample columns. As the pixels around the skyline are brighter on top and dimmer below, continuous values of "1" for the $g_m(y)$ of all the sample column pixels imply the conformance of the region formed by these pixels to the brightness property mentioned above, and they could be in the area near the skyline. The skyline candidate region is defined here as formed by three or more adjacent pixels in the skyline sample column with $g_m(y)$ of 1. Each sample column could contain multiple skyline candidate regions, each with its own upper and lower boundaries, and they do not overlap. Figure 2 is an example explaining the use of $g_m(y)$ to find the skyline candidate region along with its upper and lower boundaries. $f(m \cdot p, y)$ represents the grayscale value of some of the pixels in the m-th sampling column. $G_m(y)$ and $g_m(y)$ are the gradient and brightness analysis function corresponding to $f(m \cdot p, y)$. After the values of $G_m(y)$ and $g_m(y)$ are calculated from $f(m \cdot p, y)$, the regions in $g_m(y)$ formed by continuous pixels of 1 are taken as the skyline candidate region. The intersections between 1 and 0 in $g_m(y)$ are the upper and lower bounds of the region. The portion of Figure 2 with continuous $g_m(y)$ values of 1 is the n-th skyline candidate region in the m-th sampling column, and is represented here as $B_m(n)$, in which the range of n is determined by the number of skyline candidate regions found in sampling column m. The value of n is thus different for each sampling column.

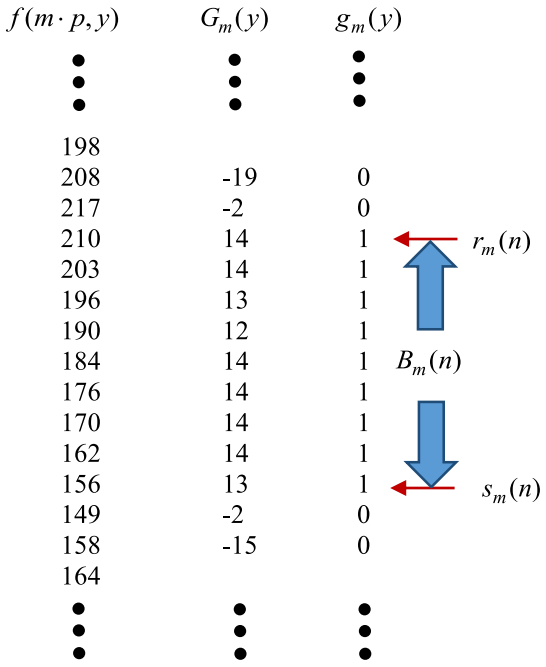


FIGURE 2. Using brightness change to find the skyline candidate region and its upper and lower boundaries.

The upper and lower bounds of $B_m(n)$ are defined as $r_m(n)$ and $s_m(n)$. The range of skyline candidate region $B_m(n)$ is as shown in Equation (6):

$$B_m(n) = f(m \cdot p, y), \quad r_m(n) \le y \le s_m(n) \tag{6}$$

The influence of noise must be taken into account when searching for the upper and lower bounds of the skyline candidate region. Figure 3 illustrates the effect of noise on the brightness analysis function $g_m(y) \cdot f(m \cdot p, y)$ is the grayscale value for a series of pixels in the sampling column m gradually becoming dimmer from the top down. $G_m(y)$ and $g_m(y)$ are the corresponding gradient and brightness analysis functions.

When no noise is present, the pixels of $f(m \cdot p, y)$ satisfying the "bright above and dim below" criterion will have $G_m(y)$ greater than zero and $g_m(y)$ of all 1. In Figure 3(a), however, the bright noise with an intensity of 212 causes the $G_m(y)$ of the pixel above to be less than zero. In Figure 3(b), the dark noise with an intensity of 152 causes the $G_m(y)$ of the pixel below to be less than zero. This will change $g_m(y)$ from 1 to 0. The upper and lower bounds of the skyline candidate region are discerned by the intersections between 1 and 0 of $g_m(y)$. When $g_m(y)$ is changed from 1 to 0, error in discerning the position of the skyline region boundary could occur, affecting the accuracy in the subsequent search of skyline candidate points. To minimize the influence of noise on the skyline detection result, noise detection and correction methods will be implemented in the search for the skyline.

As can be seen from Figure 3, the bright or dark noise will only change the $g_m(y)$ value of one pixel to 0, so when noise occurs, the affected pixel will divide the block in which

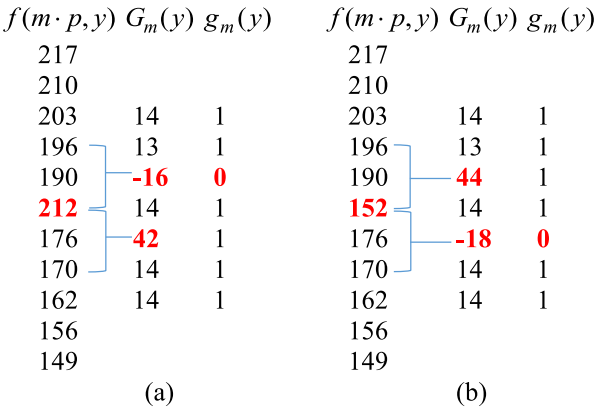


FIGURE 3. Effect of noise on the brightness analysis function. (a) Bright noise, (b) dark noise.

$g_m(y)$ is entirely 1 into two parts. According to this property, when looking for the skyline candidate region, if there is only one point of 0 in two blocks of consecutive 1 from top to bottom, that point can be identified as noise. The two blocks are then merged to eliminate the influence of noise in the determination of the skyline boundary.

After finding the skyline candidate region, the second step is to find the position of the skyline candidate point in the region. The scenario needs to be divided into two cases: the sky and earth having clear delineation between them, and having no clear delineation. In the first case, the pixel intensity will change drastically at the boundary, but not so much for adjacent pixels in the sky or earth region. The location in the skyline candidate region with the greatest gradient change is the boundary between sky and earth regions, and is the reasonable position of the skyline candidate point. $\lambda_m(n)$ is the y-coordinate of the pixel in $B_m(n)$ with the largest intensity gradient, as shown in Equation (7):

$$\lambda_m(n) = \underset{r_m(n) \le y \le s_m(n)}{\arg\max} \; G_m(y) \tag{7}$$

The maximum pixel intensity gradient can be used to locate the skyline candidate point only when the sky and earth regions have a distinct boundary. When this is not the case, such as on foggy days, the intensity gradient of adjacent pixels is very small for the skyline candidate region, and the whole area shows a gradual transition from bright to dim without clear edge features. The maximum pixel intensity gradient will be randomly distributed in the bright-to-dim region. If a point with the maximum pixel intensity gradient is selected as the skyline candidate, the connection of these random skyline candidates may result in error. The midpoint of the gradually changing region should be chosen as the skyline candidate point to outline a reasonable skyline profile. Thus, to find a reasonable skyline candidate point, we must first decide if a clear boundary exists between the sky and earth areas in the skyline candidate region.

To determine if a clear boundary exists between the sky and the earth regions, it is necessary to first compare the intensity

gradient of $\lambda_m(n)$ and the average intensity gradient of the pixels in$B_m(n)$. Definition of the average intensity gradient $\overline{G_m(n)}$ for pixels in $B_m(n)$ is shown in Equation (8):

$$\begin{aligned}&\overline{G_m(n)}\\&= \{G_m[r_m(n)] + G_m[r_m(n)+1] + \cdots + G_m[s_m(n)-1]\\&\quad + G_m[s_m(n)]\}/[s_m(n) - r_m(n) + 1]\\&= [f(m\cdot p, r_m(n)-1) + f(m\cdot p, r_m(n)) - f(m\cdot p, s_m(n))\\&\quad - f(m\cdot p, s_m(n)+1)]/[s_m(n) - r_m(n) + 1] \qquad (8)\end{aligned}$$

Here, q is set as the threshold which is greater than one. Assuming $G_m[\lambda_m(n)]$ is the maximum in pixel intensity gradient of the pixels in the $B_m(n)$, and $b_m(n)$ represent the y-coordinate of skyline candidate points in $B_m(n)$. If $G_m[\lambda_m(n)]$ is greater than $q\cdot\overline{G_m(n)}$ that indicates the distinct edge features exist at $\lambda_m(n)$, so $\lambda_m(n)$ is the y-coordinate of the skyline candidate point, $b_m(n)$. When $G_m[\lambda_m(n)]$ is less than or equal to $q\cdot\overline{G_m(n)}$, it is possible that pixels in the $B_m(n)$ have similar intensity gradients and the distinct edge features does not exist around $\lambda_m(n)$. In order to determine if a clear boundary exists around $\lambda_m(n)$, it is necessary to find the intensity gradient of the pixels around $\lambda_m(n)$. Here we define $G_m^{\min}[\lambda_m(n)]$ as the minimum of the intensity gradient for the two pixels above and below the $\lambda_m(n)$, as shown in Equation (9):

$$G_m^{\min}[\lambda_m(n)] = \min\{G_m[\lambda_m(n)-1], G_m[\lambda_m(n)+1]\} \quad (9)$$

Here, $\overline{G_m(n)}$ and $G_m^{\min}[\lambda_m(n)]$ are compared to determine the existence of a clear boundary around $\lambda_m(n)$. When $G_m^{\min}[\lambda_m(n)]$ is smaller than $\overline{G_m(n)}$, the intensity gradient of pixels near $\lambda_m(n)$ suddenly drops from the maximum value of $G_m[\lambda_m(n)]$ to below the average value $\overline{G_m(n)}$. This means a significant intensity gradient change is present around $\lambda_m(n)$, i.e. distinct edge features exist at the boundary between sky and earth, and $\lambda_m(n)$ is the y-coordinate of the skyline candidate point, $b_m(n)$. When the intensity gradient for the pixels above and below $G_m[\lambda_m(n)]$ is not less than the average value $\overline{G_m(n)}$, the intensity gradient of $\lambda_m(n)$ and its surrounding pixels is small, and no distinct edge exists around the maximum gradient $G_m[\lambda_m(n)]$. The whole skyline candidate region thus has gradual variation in brightness, and $b_m(n)$ should be taken at the midpoint of the region. The determination of $b_m(n)$ through the presence or absence of a clear boundary between the sky and earth is shown in Equation (10):

$$b_m(n) = \begin{cases}\lambda_m(n), when\{G_m^{\min}[\lambda_m(n)] < \overline{G_m(n)}\}\\ \quad ||\{G_m[\lambda_m(n)] > q\cdot\overline{G_m(n)}\}\\ (r_m(n)+s_m(n))/2, \quad others\end{cases} \quad (10)$$

In this study, q is set to be *2*. As seen in Equation (10), when determining the presence of a clear boundary in the skyline candidate region, $\lambda_m(n)$, the maximum intensity gradient for that region is an important parameter. However, noise could cause error in the search for $\lambda_m(n)$ in the candidate region. For example, in Figure 3(a), the bright noise with intensity of 212 changes the $G_m(y)$ of the pixel below to be 42, producing a large positive intensity gradient for the pixel below. In Figure 3(b), the dark noise with intensity of 152 changes the $G_m(y)$ of the pixel above to be 44, producing a large positive intensity gradient for the pixel above. These two positive intensity gradients are erroneous information representing noise, but could be wrongly judged as the maxima of the regional intensity gradient. As such, the impact of noise must be considered during the search of skyline candidate points. To avoid taking the erroneous information caused by noise as the maximum of the regional intensity gradient, the error in the intensity gradient caused by noise must be eliminated when comparing the maximum of regional intensity gradient. Based on the aforementioned detection of noise position with the change in $g_m(y)$, the influence of noise on the maximum of the regional intensity gradient could be removed in the following manner: when $g_m(y)$ is found to be 0 because of the presence of intense noise in the sampling column, comparison between the pixel with $g_m(y)$ of 0 and the intensity of pixels above and below it could be used to determine if noise is bright or dark. In Figure 3(a), the pixel with $g_m(y)$ of 0 has an intensity of 190, and the intensities for the pixels above and below it are 196 and 212, respectively. The intensity for the pixel below does not conform to the ''bright above and dim below'' criterion, and could therefore be determined as bright noise. In Figure 3(b), the pixel with $g_m(y)$ of 0 has an intensity of 176, and the intensities for the pixels above and below it are 152 and 170, respectively. The intensity for the pixel above does not conform to the ''bright above and dim below'' criterion, and could therefore be determined as dark noise. For bright noise, the intensity gradient for a distance of two pixels below the pixel with $g_m(y)$ of 0 must be neglected. For dark noise, the intensity gradient for a distance of two pixels above the pixel with $g_m(y)$ of 0 must be neglected. This is so because the large positive intensity gradient is formed by the noise. The real regional maximum is found from the remaining pixels to detect the presence of distinct skyline in the region.

C. FILTERING OF SKYLINE CANDIDATE POINTS

In the step of skyline candidate points filtering, the brightness change between the image blocks above and below the candidate point is calculated. It used to remove those points having only the ''bright above and dim below'' feature with neighboring pixels, but are not on the boundary between bright and dark regions. As the reasonable skyline has a la4rge bright region above it and a dim region below it, the right skyline candidate point has a higher pixel intensity in the region above it. However, only the pixel intensity change between the skyline candidate points and the neighboring pixels are considered during the search, and not the interfacial features between the bright and dim regions. Some candidate points, though having the ''bright above and dim below'' feature with their neighboring pixels, do not possess the same property for pixel intensity of the regions above and below it, and are therefore not the right skyline position.

Through a comparison of the brightness change of the regions above and below the candidate points, we remove those having only the "bright above and dim below" feature with neighboring pixels, but are not on the boundary between bright and dark regions, and keep the reasonable points.

During the extraction, the average pixel intensities of three regions are used to determine if the candidate point should be kept. The y-coordinate of the skyline candidate point in the n-th region of the m-th sampling column is set to be $b_m(n)$. Here the average image intensity for the region above the candidate point is defined as $U_m(n)$, for the region below defined as $V_m(n)$, and for the local region centered at the candidate point as $A_m(n)$. The average pixel intensity $U_m(n)$ for the region above $b_m(n)$ is the average pixel intensity between the point above $b_m(n)$ and the previous skyline candidate point. The average pixel intensity $V_m(n)$ for the region below $b_m(n)$ is the average pixel intensity between the point below $b_m(n)$ and the next skyline candidate point. With $m = i$ and $n = j$ for $b_m(n)$, and p as the spacing between adjacent sampling columns, the calculation for $U_i(j)$ and $V_i(j)$ is shown in Equation (11):

$$\begin{cases} U_i(j) = \sum_{y=b_i(j-1)}^{b_i(j)-1} f(i \cdot p, y) \Big/ b_i(j) - b_i(j-1) \\ V_i(j) = \sum_{y=b_i(j)+1}^{b_i(j+1)} f(i \cdot p, y) \Big/ b_i(j+1) - b_i(j) \end{cases} \tag{11}$$

Here, $A_m(n)$ is defined as the average pixel intensity for a region within the range η above and below the candidate point $b_m(n)$. With $m = i$ and $n = j$ for $b_m(n)$, the computation for $A_i(j)$ is as shown in Equation (12):

$$A_i(j) = \frac{\sum_{y=b_i(j)-\eta}^{b_i(j)+\eta-1} f(i \cdot p, y)}{2 \cdot \eta} \tag{12}$$

As $A_i(j)$ is the local characteristic around $b_i(j)$, the range of η cannot be too small. Here, the reasonable vertical distance between two adjacent skyline profiles is defined as TH. In some images, there are more than two skyline profiles that are "bright above and dim below," such as overlapping mountains. If the vertical distance between two adjacent skylines is greater than TH, they are considered independent and treated as likely skyline segments. If the distances between $b_i(j)$, the previous candidate point $b_i(j-1)$, and the next candidate point $b_i(j+1)$ are all less than TH, the distances between $b_i(j)$ and candidate points above and below it are too short, and η is set to be TH. Otherwise, η is the larger of the distances to the skyline candidate points above and below, as shown in Equation (13):

$$\begin{cases} if \quad |b_i(j-1) - b_i(j)| < TH \ and \ |b_i(j) - b_i(j+1)| \\ \qquad < TH, \eta = TH \\ else \quad \eta = \max(|b_i(j-1) - b_i(j)|, |b_i(j) - b_i(j+1)|) \end{cases} \tag{13}$$

In this study, TH is set to be $h/8$, in which h is the image height. Here $\delta_m(n)$ is defined as a record of the filtering result for skyline candidate point. Each skyline candidate in $b_m(n)$ has a corresponding $\delta_m(n)$ record, with initial value of "0". When the location of $b_m(n)$ is a reasonable skyline candidate point, the local region $A_m(n)$ centered at $b_m(n)$ will contain sky and earth pixels, and will therefore be dimmer than the $U_m(n)$ of sky region and brighter than the $V_m(n)$ of earth region. $\delta_m(n)$ will be set to "1" to indicate the conformance of the pixel intensity in regions above the below $b_m(n)$ to the "bright above and dim below" requirement, as shown in Equation (14). $b_m(n)$ is therefore a reasonable skyline candidate point.

$$if \ U_m(n) > A_m(n) \ and \ V_m(n) < A_m(n), \ \delta_m(n) = 1 \tag{14}$$

In the subsequent step to connect skyline points and obtain a skyline profile, only those candidate points with a $\delta_m(n)$ of 1 will be evaluated. Candidate points with a $\delta_m(n)$ of 0 are removed. This reduces the computational workload of the algorithm and avoids connecting the wrong line segments.

D. CONNECTING SKYLINE CANDIDATE POINTS

The skyline candidate points in the sampling columns are connected horizontally to obtain the complete skyline profile. This includes two steps: connecting similar skyline candidate points and merging regional skyline segments. In the first step, the candidate points of adjacent sampling columns are connected to obtain the regional skyline segments. If two candidate points of adjacent sampling columns belong to the same skyline, the image features around them will be similar, and the distance will be in a reasonable range. Candidate points of adjacent sampling columns can therefore be connected based on the proximity of the distance between them and similarity in image features. The continuous line segments from left to right between the candidate points can be used to draw the skyline segment profile.

Assuming that the i-th and $(i+1)$-th sampling columns are adjacent, three rules are first applied to find the skyline candidate point pairs in the columns with similar image features. Based on the similarity of the point pairs, the pairs with a higher similarity will be selected and connected to obtain regional skyline segments for the adjacent sampling columns. $b_i(l)$ is defined as the y-coordinate of the l-th candidate point in the i-th sampling column, and $b_{i+1}(t)$ is the y-coordinate of the t-th candidate point in the $(i+1)$-th sampling column. Based on candidate point filtering and the condition that $\delta_i(l)$ and $\delta_{i+1}(t)$ are both 1, the skyline candidate points at $b_i(l)$ and $b_{i+1}(t)$ must satisfy the three rules below to be considered as candidate point pairs with similar image features:

1) RULE 1: THE VERTICAL DISTANCE BETWEEN $b_i(l)$ AND $b_{i+1}(t)$. IS LESS THAN THE PRESET VALUE

Here, α is set as the threshold for the vertical distance between $b_i(l)$ and $b_{i+1}(t)$. When the skyline candidate points at $b_i(l)$ and $b_{i+1}(t)$ are on the same skyline, the absolute value of the difference between $b_i(l)$ and $b_{i+1}(t)$ must be less than α. α can be adjusted for different situations. The α for this experiment

is set to be $3 \cdot p$, and p is the spacing between two adjacent sampling columns.

2) Rule 2: THE REGIONS ABOVE AND BELOW $b_i(l)$ AND $b_{i+1}(t)$ NEED TO SATISFY THE CROSS COMPARISON REQUIREMENT

When the candidate points are on the skyline, the pixel brightness for the region above will be greater than the region below. Here, $U_i(l)$ and $V_i(l)$ calculated in Equation (11) are used as the average pixel intensity for the regions above and below $b_i(l)$, and $U_{i+1}(t)$ and $V_{i+1}(t)$ are used for $b_{i+1}(t)$. When the candidate points at $b_i(l)$ and $b_{i+1}(t)$ are on the same skyline, $U_i(l)$ should be greater than $V_{i+1}(t)$ and $U_{i+1}(t)$ should be greater than $V_i(l)$.

3) RULE 3: THE AVERAGE PIXEL INTENSITY MUST BE CLOSE FOR THE REGIONS ABOVE AND BELOW $b_i(l)$ AND $b_{i+1}(t)$

When the two candidate points at $b_i(l)$ and $b_{i+1}(t)$ are on the same skyline, the regions above them are continuous. Therefore, the average pixel intensity $U_i(l)$ and $U_{i+1}(t)$ for the regions above the two candidate points should be close. For the same reason, the average pixel intensity $V_i(l)$ and $V_{i+1}(t)$ for the regions below the two candidate points should also be close. $r_i(l)$ and $r_{i+1}(t)$ represent the upper bound of the skyline candidate region, $B_i(l)$ and $B_{i+1}(t)$, defined in Equation (6), and $s_i(l)$ and $s_{i+1}(t)$ represent the lower bound of the $B_i(l)$ and $B_{i+1}(t)$. When the two candidate points at $b_i(l)$ and $b_{i+1}(t)$ are on the same skyline, $U_i(l)$ will be in the pixel intensity range of $r_{i+1}(t)$ to $b_{i+1}(t)$, and $U_{i+1}(t)$ in $r_i(l)$ to $b_i(l)$. $V_i(l)$ will be in the pixel intensity range of $b_{i+1}(t)$ to $s_{i+1}(t)$, and $V_{i+1}(t)$ in $b_i(l)$ to $s_i(l)$.

After skyline candidate points with similar image features are found with the three rules above, their similarity is compared. $b_i(l)$ and $b_{i+1}(t)$ are defined as the y-coordinate of two candidate points with similar image features. The difference function $\varphi(i, l, t)$ is used to represent the degree of similarity between candidate points at $b_i(l)$ and $b_{i+1}(t)$. $\varphi\ (i, l, t)$ is defined as the sum of the absolute values of the pixel intensity difference for the regions above and below the skyline candidate points, as shown in Equation (15). A smaller value of $\varphi\ (i, l, t)$ indicates a higher similarity between $b_i(l)$ and $b_{i+1}(t)$.

$$\varphi(i, l, t) = |U_i(l) - U_{i+1}(t)| + |V_i(l) - V_{i+1}(t)| \quad (15)$$

After the degree of similarity is calculated for the skyline candidate point pairs, the candidate points of adjacent sampling columns are connected to give a reasonable skyline of continuous curve in the image. The candidate points forming the skyline in adjacent sampling columns should have a one-to-one relationship, and the line segments of different candidate point pairs should not cross. As only the similarity between left and right skyline candidate points are considered during the search for candidate point pairs with similar image features, two cases must now be considered among the candidate point pairs. The first is the one-to-many situation, in which one candidate point of the left sampling column and two or more points of the right sampling column meet the criterion of having similar image features. The second is the many-to-one situation, in which two or more candidate points of the left sampling column and the same point of the right sampling column meet the criterion of having similar image features. To solve these two cases, after finding all of the candidate point pairs of adjacent sampling columns with similar image features, the occurrence of the one-to-many situation is first checked for the candidate points in the left sampling column. If present, the difference function between that candidate point and each candidate point in the right sampling column having similar image features is computed as in Equation (15). A smaller difference function $\varphi(i, l, t)$ implies a higher degree of similarity for skyline candidate point pairs and a greater possibility of being the actual skyline profile. Thus, only the candidate point pairs with the smallest difference function are kept, and others removed, to ensure the one-to-one relation of the remaining candidate point pairs. After checking for all the skyline candidate points of the left sampling column, the many-to-one relation is checked for the candidate points of the right sampling column. If it is present, the candidate point pairs with the smallest difference function calculated with Equation (15) are kept, and the others removed. After the check is completed for left and right sampling columns, the candidate point pairs that remain should all have a one-to-one relation. However, another anomaly still requires attention, and it is the crossover between line segments of different candidate point pairs. Here, the difference function of each candidate point pair is again computed as in Equation (15), and the pair with the smallest $\varphi\ (i, l, t)$ is retained, while other crossed line segments are eliminated. The remaining skyline candidate point pairs are then connected.

The second step in connecting the skyline is the joining of regional skyline segments, which is done by comparing the image features at the endpoints of regional skyline segments. The line segments are merged to obtain the complete skyline profile. In the first step of connecting similar skyline candidate points, a complete skyline profile may not be achieved. A complete skyline could include different regions such as mountains, sea, and woods, which change the image property of the skyline in these regions. The similar candidate points could become disjointed at the border of skyline segments with different properties, such as at the intersection of mountains and sea, and merging is needed to obtain a complete skyline profile. As the regional skyline segments are formed by connecting skyline candidate points, the left and right endpoints of each regional skyline segment can be expressed with the coordinates of candidate points. Furthermore, only the right end of the left line segment and the left end of the right line segment need be compared. Other candidate points of the regional line segment can be excluded from this step.

Before the merge, skyline segments consisting only of two skyline candidate points are removed, as only line segments formed by more than three skyline candidate points are regarded as effective regional skyline segments. Two rules are

used to find all possibly merging regional skyline segments. Based on the similarity between the endpoints of two regional skyline segments, those with high degree of similarity are merged. The two rules used here are Rules 1 and 2 in connecting skyline candidate points. Considering two regional skyline segments in the image, the y-coordinate of the right end of the left segment is $b_i(l)$, while that for the left end of right segment is $b_j(t)$. Under the condition of i being smaller than j, the endpoints at $b_i(l)$ and $b_j(t)$ must satisfy Rule 1 and 2 for the two segments to be merged. In the Rule 1, the α should be adjusted for the different distance between i and j. For example, α is $3 \cdot p$ in the original Rule 1, but could be set as $3 \cdot (j-i) \cdot p$ in the adapted Rule 1. According to Rule 2, the average pixel intensity $U_i(l)$ for the region above the right end of the left segment is larger than $V_j(t)$t, the average pixel intensity for the region below the left end of the right segment, and the average pixel intensity $U_j(t)$ for the region above the left end of the right segment is larger than $V_i(l)$, the average pixel intensity for the region below the right end of the left segment.

After all possibly merging regional skyline segments are identified with Rule 1 and 2, the segments are merged according to the degree of similarity between their endpoints. The process of merging regional skyline segments is very similar to connecting skyline candidate points. As the skyline is a continuous curve in the image, the endpoints of regional line segments must satisfy the one-to-one relationship during merging. During the search of possibly merging regional skyline segments, the case of one-to-many could occur at the right end of segments, while many-to-one could occur at the left end. The connection between endpoints could also cross. The solution to the above problems is also similar to the step taken for connecting skyline candidate points of adjacent sampling columns, i.e., Equation (15) is used to calculate the difference function of candidate points, to select the endpoint pair of regional skyline segments with the greatest similarity.

After finding all the possibly merging regional skyline segments, their right ends are checked for the case of one-to-many. If this case is present, this endpoint pair with the greatest similarity according to Equation (15) is retained. The same is performed for the left ends of all the regional skyline segments. Equation (15) is next used to remove any cross connection between the ends of skyline segments. The remaining endpoint pairs of the regional skyline segments are joined to obtain the skyline detection result.

This paper defines the reasonable skyline profile. In this, the distance between the two endpoints and the image frame must be less than 5 percentage of image width. Sometimes an image will have more than two reasonable skyline profiles, such as in the scenario of overlapping mountains. This algorithm can be used to find all the possible skyline profiles at the same time. In the experiment results section below, the uppermost skyline profile is set to be the first choice in detected skyline results. The other profiles can be displayed as needed by setting appropriate parameters.

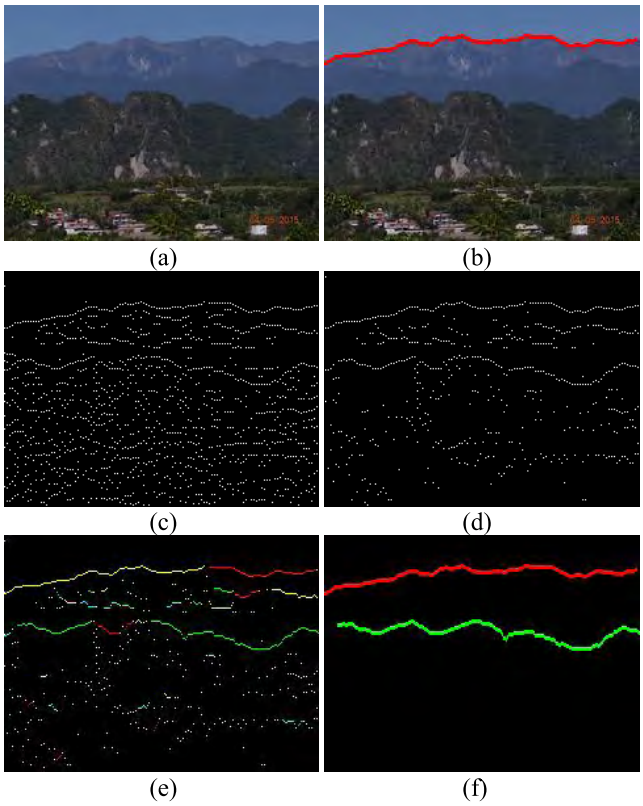


FIGURE 4. The detection result and the processing of the proposed algorithm in an experimental image, image resolution: 240 × 180. (a) The original image. (b) The skyline detection result in the original image. (c) The skyline candidate points before filtering. (d) The skyline candidate points after filtering. (e) The regional segments of the skylines generate by connecting the candidate points. (f) The reasonable skyline profiles generate by connecting the regional segments.

The six images in Fig. 4 illustrate the detection result and the processing of the proposed algorithm in an experimental image. Fig. 4 (a) is the original image and the resolution is 240 × 180. Fig. 4 (b) presents the skyline detection result with a red line in the original image. The white pixels in Fig. 4 (c) are the skyline candidate points selected in the sample columns based on the methods in Section II-B, and white pixels in Fig. 4 (d) are the skyline candidate points after filtering by the method in Section II-C. Fig. 4 (e) shows the regional segments of the skyline, that are connected the candidate points by the algorithm in this paper. In Fig. 4 (f), the reasonable skyline profiles are generated by connecting the regional segments above.

IV. EXPERIMENTAL RESULTS AND ANALYSES

The experimental results are divided into three parts. The first part is testing under different weather and scenes in order to assess the ability of the algorithm to correctly identify the skyline profile in images of various weather and scenes. Actual images are used as examples to illustrate the processing steps of the algorithm. The second part involves generating the statistics of the percentage of correctness of the algorithm. Here, the definition of ''accurate skyline detection result''

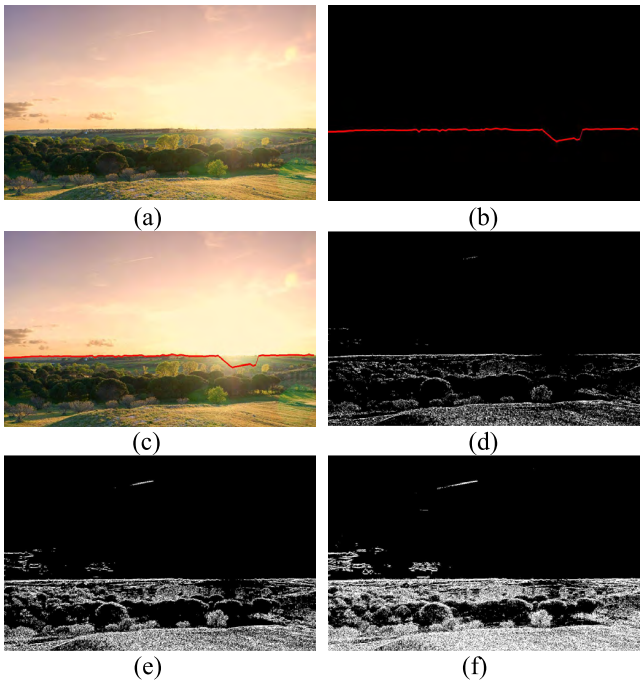


FIGURE 5. Plain-sunset, image resolution: 960 × 600. (a) The original image. (b) The reasonable skyline profiles. (c) The skyline detection result in the original copy. (d) SOBEL with threshold = 40. (e) SOBEL with threshold = 20. (f) SOBEL with threshold = 10.

will be given, followed by the statistics of the algorithm in successfully identifying the skyline in images of different weather and scenes. The third part is an analysis of the algorithm processing speed. With the image size fixed, the average processing speed of the algorithm can be accelerated by adjusting the interval between sampling columns, thus meeting the demands of real-time computation. The method to adjust the processing speed, and its effect on the results of skyline detection, are described here.

A. TESTING UNDER DIFFERENT WEATHER AND SCENES

In testing under different weather conditions and scenes, dozens of test images are selected for each of the following: different weather, such as sunny, overcast, or cloudy days, and different scenes such as mountain, sea, or forest, to assess the robustness of the algorithm. A total of 200 images are tested. Among them, the characteristics of the skyline, such as the brightness, contrast, and edge strength are all different. Profiles of mountains or forests are also included as part of the skyline in some images so that the skyline contour appears as a random curve. Examples are given below, illustrating the skyline detection of the algorithm in images of different weather and scenes.

Fig. 5 to 10 present the skyline detection results of the algorithm in six images of different scenes and weather, and compare the differences between the skyline profile obtained with our algorithm and skyline edge features found by edge detection. In the images, (a) is the original copy, and (b) illustrates the reasonable skyline profiles obtained by

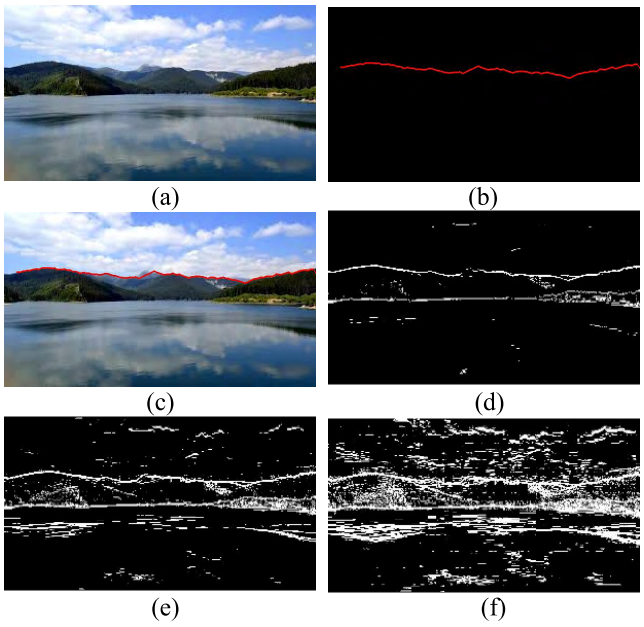


FIGURE 6. Lakeside-cloudy, image resolution: 600 × 336. (a) The original image. (b) The reasonable skyline profiles. (c) The skyline detection result in the original copy. (d) SOBEL with threshold = 40. (e) SOBEL with threshold = 20. (f) SOBEL with threshold = 10.

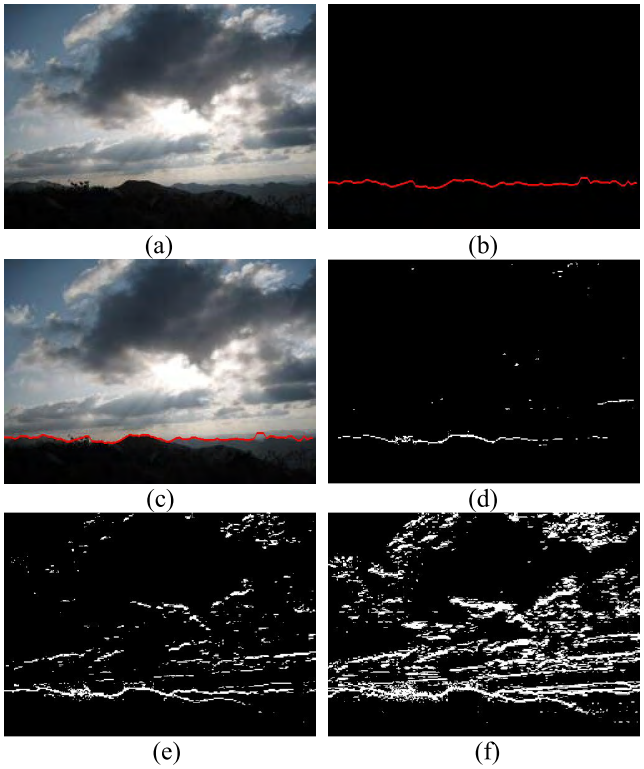


FIGURE 7. Mountain-cloudy, image resolution: 500 × 360. (a) The original image. (b) The reasonable skyline profiles. (c) The skyline detection result in the original copy. (d) SOBEL with threshold = 40. (e) SOBEL with threshold = 20. (f) SOBEL with threshold = 10.

the algorithm. In Figure (c), we present the skyline detection result in the original copy using a red line. Images (d) to (f) are edge detection results on the original image using

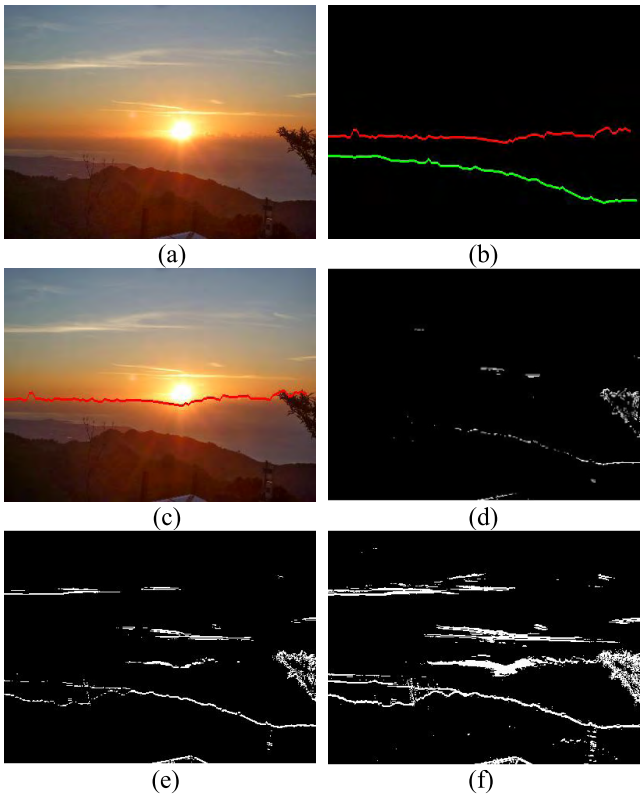


FIGURE 8. Mountains-sunset, image resolution: 400 × 300. (a) The original image. (b) The reasonable skyline profiles. (c) The skyline detection result in the original copy. (d) SOBEL with threshold = 40. (e) SOBEL with threshold = 20. (f) SOBEL with threshold = 10.

a 3 × 3-level SOBEL mask with threshold values of 40, 20, and 10, respectively, and the edge points are shown in white. Since the original resolution of the six images is not necessarily the same, the detection results will be displayed here with the proportion taken by the skyline in the original image, and the line width of the detection results in the images will thus vary.

Fig. 5 shows the sunset image of a plain with a resolution of 960 × 600. The skyline is clear and linear in the image, except for some interference from the setting sun on the right side. As can be seen from Fig. 5 (d), there is no obvious edge feature in the vicinity of the setting sun, and edge points not belonging to the skyline are also detected. In Fig. 5 (e) and (f), the skyline at the setting sun is found by lowering the threshold, but more edge noise that is not part of the skyline is also identified. This could make it difficult to find the correct skyline in Fig. 5 (e) and (f) using straight line detection. In Fig. 5 (c), one can see that our algorithm accurately identifies most of the skyline profile. Although the regional segments of the skyline are interfered at the setting sun, the full skyline profile can be obtained by merging them.

The skyline in Fig. 6 is a curve of the outline of a mountain, with a resolution of 600 × 336 and on a cloudy day. A comparison of Fig. 6 (d), (e), and (f) shows clear edge features at the mountain outline, but lowering of the threshold findsmuch

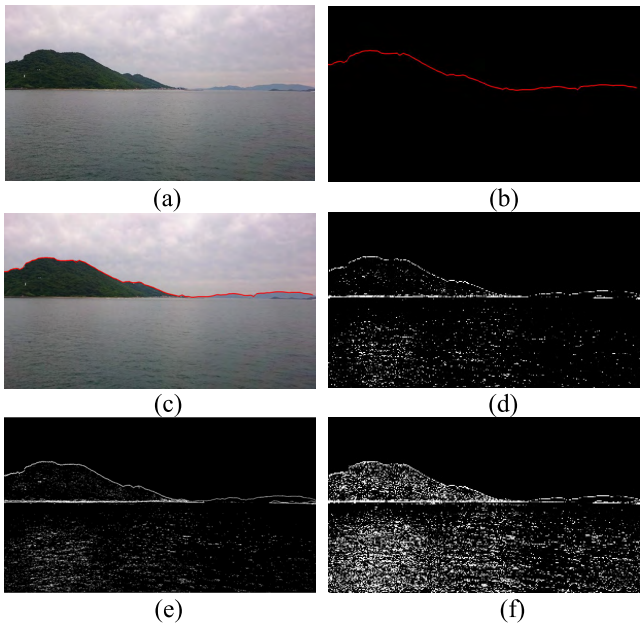


FIGURE 9. Seaside-rainy, image resolution: 1600 × 900. (a) The original image. (b) The reasonable skyline profiles. (c) The skyline detection result in the original copy. (d) SOBEL with threshold = 40. (e) SOBEL with threshold = 20. (f) SOBEL with threshold = 10.

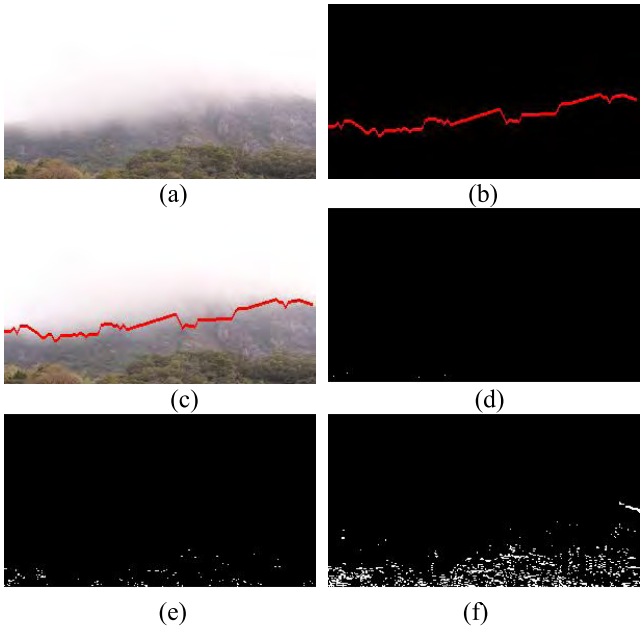


FIGURE 10. Mountain region-heavy fog, image resolution: 300 × 168. (a) The original image. (b) The reasonable skyline profiles. (c) The skyline detection result in the original copy. (d) SOBEL with threshold = 40. (e) SOBEL with threshold = 20. (f) SOBEL with threshold = 10.

non-skyline edge noise in Fig. 6 (e) and (f). In Fig. 6 (d), one can also see the clear edge features at the intersection of the mountain and lake. Since the line segment at the mountain/lake boundary is closer to a straight line than the mountain outline, a straight line detection on the edge points of Fig. 6 (d) will yield the straight line at the intersection

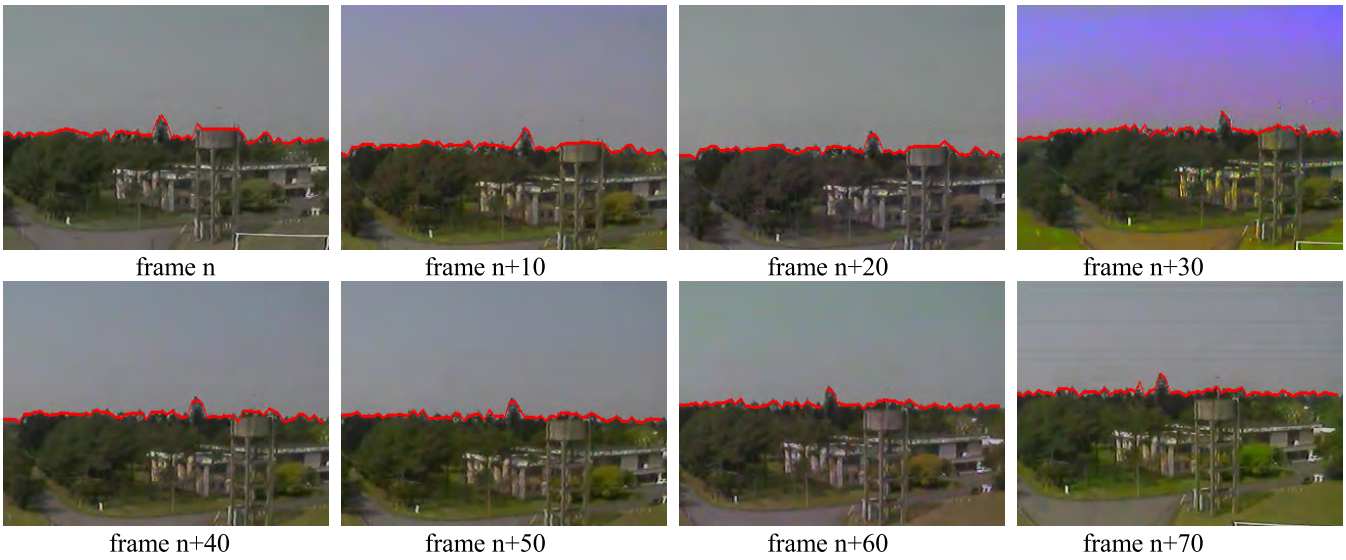


FIGURE 11. Skyline detection with ground objects in continuous image set 1, image resolution: 320 × 240.

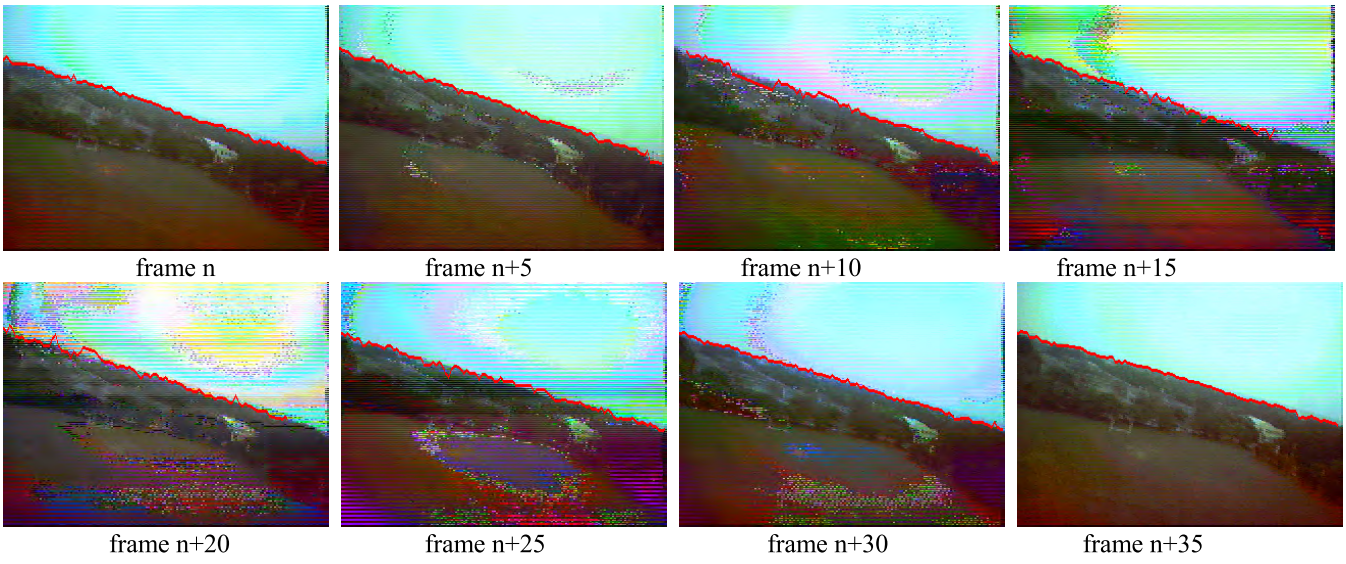


FIGURE 12. Skyline detection with varying noise in continuous image set 2, image resolution: 320 × 240.

of the mountain and lake, and not the skyline made of the mountain outline. In Fig. 6 (c), our algorithm finds a complete outline of the mountain. Since the mountain/lake boundary does not fulfill the criterion of "bright above and dim below," it will not generate skyline candidate points. The final skyline profile is almost identical to the mountain outline.

In Fig. 7, the skyline is a combination of outlines of mountains at different distances, with a resolution of 500 × 360. Since the mountains forming the skyline at the left and right are at different distances, the skyline at left has greater edge strength than that at right. There is also interference of the sunlight and clouds from above. As can be seen in Fig. 7 (d), a threshold that is too high will not be able to detect the skyline at right. If the threshold is reduced as in Fig. 7 (e) and (f), the noise from the sunlight and clouds above is also identified. In Fig. 7 (c), our algorithm finds the skyline regional segments both on the left and right, and the full skyline profile is obtained by connecting the regional segments of the skyline.

Fig. 8 has two skyline profiles: one in the cloudy region with weak edge features, and the other at the boundary between the mountains and sky. The resolution of the image is 400 × 300.

From Fig. 8 (d) and (e), one can see that edge detection can only give the edge profile at the boundary between the mountains and sky, and not the skyline features in the cloudy region. In Fig. 8 (b), our algorithm finds the outline at both the cloudy region and the mountains. Although the regional

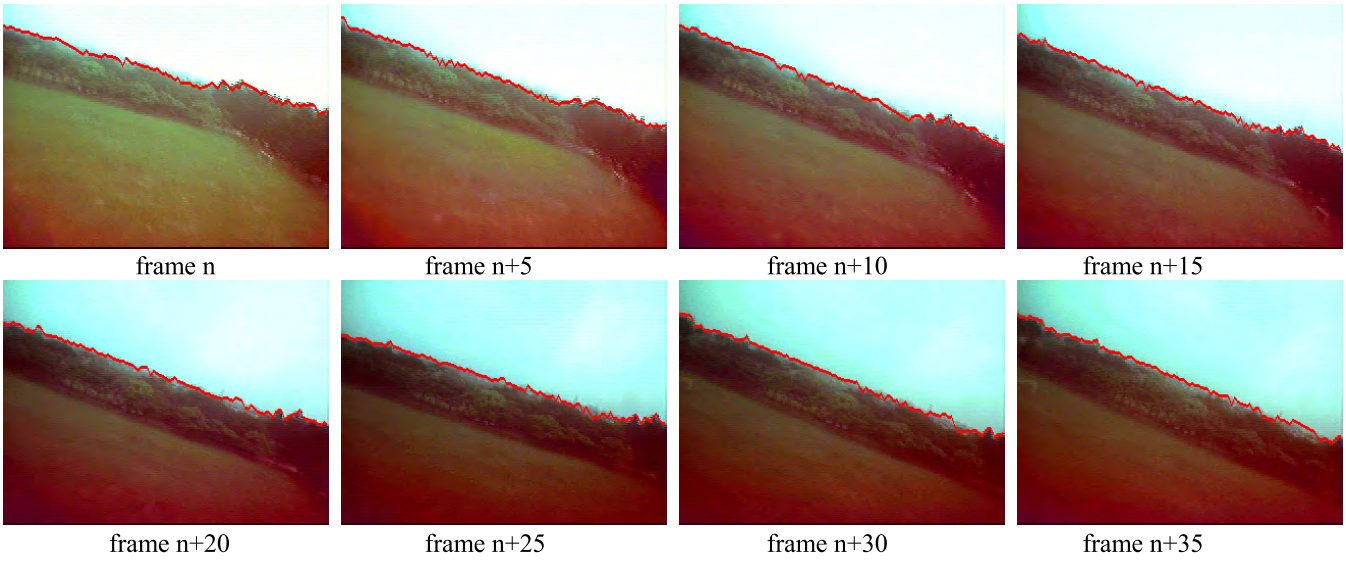


FIGURE 13. Skyline detection with varying illumination in continuous image set 3, image resolution: 320 × 240.

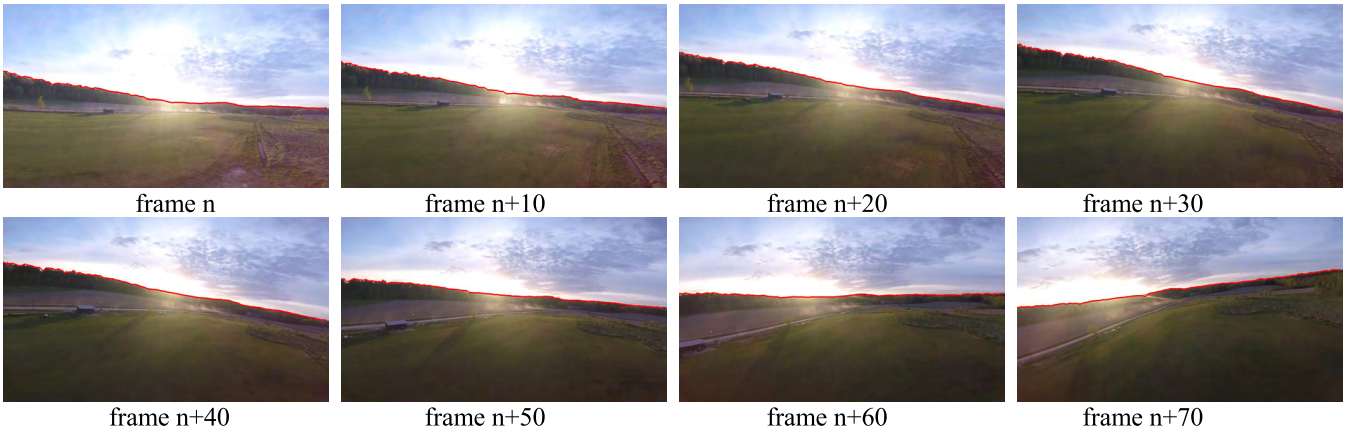


FIGURE 14. Skyline detection in sunny weather in continuous image set 4, image resolution: 1280 × 720. The camera faces the direction of sunset.

segments of the skyline in the cloudy region are interfered by the setting sun, a complete profile can still be obtained by connecting the regional segments of the skyline. Since the current algorithm sets the skyline profile at the upper portion of the image as the detection result, Fig. 8 (c) only shows the skyline contour in the cloudy region.

In Fig. 9, the skyline is composed of the outlines of two mountains, at a resolution of 1600 × 900. Since the two mountains are at different distances and the view is poor on rainy days, the image characteristics of the left mountain outline are very different from those of the right. As can be seen from Fig. 9 (d), it is not possible to detect the edge features of the outlines for the left and right mountains at the same time. If the threshold value is tuned down, as shown in Fig. 9 (e) and (f), the noise of the mountain and sea regions will also be found, and the straight line segments at the boundary between the mountains and sea will be more pronounced. If straight line detection is used on Fig. 9 (e), the straight line formed by the intersection of the two mountains and the sea will be identified, not between the sky and mountains. As seen in Fig. 9 (c), our algorithm finds the outlines of both the left and right mountains and the complete skyline profile can be obtained by connecting the outlines.

Fig. 10 is a shot in heavy fog, with a resolution of 300×168. From Fig. 10 (d), (e), and (f), it can be seen that the boundary between sky and mountain contains almost no edge features.

Thus, edge detection cannot be used to locate the skyline. Our algorithm, on the other hand, uses the ''bright above and dim below'' property at the sky/mountain boundary to find the skyline candidate points, and connects them to obtain the skyline profile, as shown in Fig. 10 (c).

B. STATISTICAL ANALYSIS OF THE ACCURACY OF THE ALGORITHM

The proposed algorithm can be applied to scenes with distinct and indistinct skylines. This has not been discussed

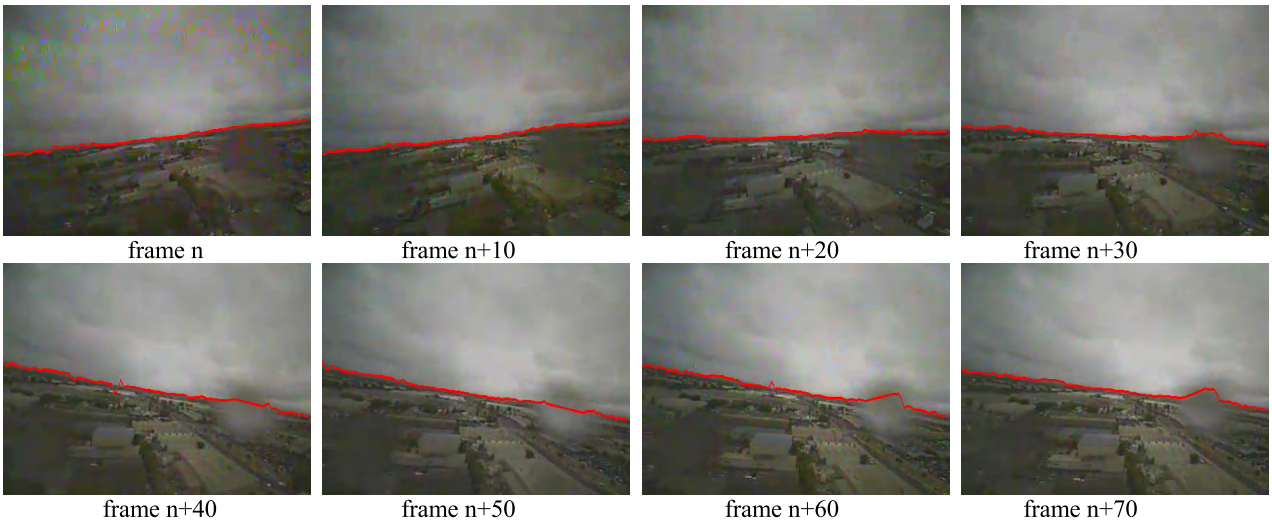


FIGURE 15. Skyline detection in rainy weather in continuous image set 5, image resolution: 400 × 300. There is the water drop attached to the camera lens.

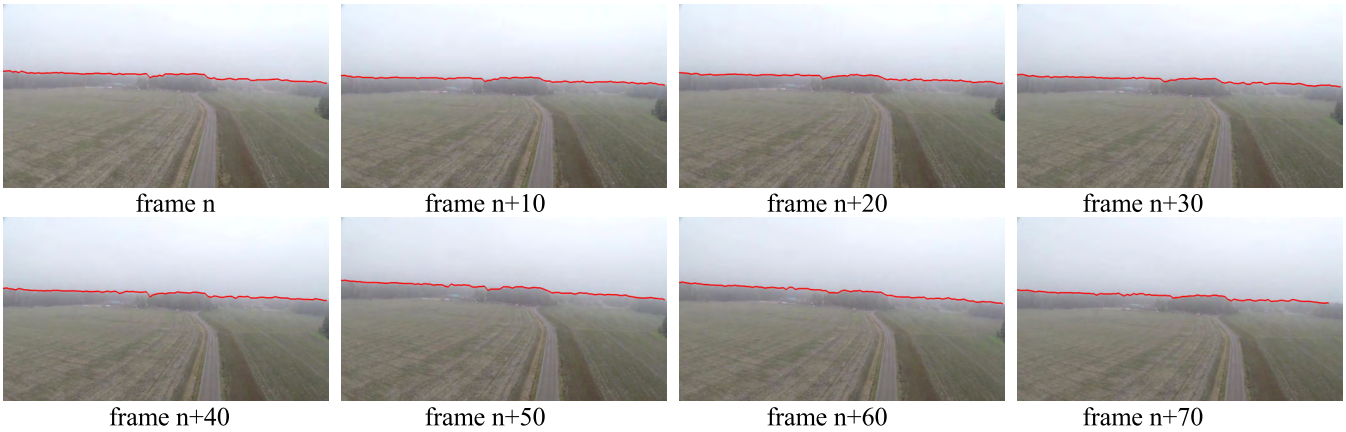


FIGURE 16. Skyline detection in foggy weather in continuous image set 6, image resolution: 1280 × 720.

for any other algorithms. Therefore, before determining the percentage of accuracy, a definition for ''accurate skyline detection result'' must be given. In this algorithm, the skyline candidate points are the basic units of the skyline profile. This paper thus defines accurate skyline candidate points in scenes with clear skylines as having a pixel difference of less than 1 between its vertical coordinate in the sampling column, and that in the sampling column of the skyline identified by multiple people with the naked eye. For scenes with an indistinct skyline, a difference within 5 pixels indicates accuracy, to make up for interference from clouds and fog. The number of correct skyline candidate points in the profile is used to calculate the similarity between the single-image skyline detection result and the actual skyline, as shown in (16):

$$Similarity = {}^{N_C}/_{N_T} \times 100\% \tag{16}$$

In which N_T is the total number of sampling columns in the image, and N_C is the total number of correct skyline candidate points. The similarity must exceed 80% between the single-image accurate skyline detection result and the actual skyline. To better account for the accuracy of our algorithm, in addition to the 200 skyline images of different weather and scenes, the test images also included six sets of continuous skyline shots taken by an unmanned vehicle during its flight, with a total image number of about 20,863. The single shots are either obtained by the researchers or downloaded from the Internet. Continuous image sets 1–3 were captured by the airborne camera, and 4-6 were downloaded from youtube [12]–[14]. In these images, the factors affecting the result of skyline detection can be divided into four categories:

1) The skyline shape, such as linear or nonlinear.

2) The brightness of the image, such as sunny, overcast, or cloudy days, for which the overall brightness of the image is different.

3) The complexity of the image, such as interference from roads, forests, or clouds on the ground or in the sky.

4) The noise interfering with the image signal.

FIGURE 17. Skyline detection results that were sampled from the 200 single images under different weather and scenes.

Figs. 11 to 16 are, respectively, some of the detection results of the six sets of continuous shots containing the skyline.

Fig. 11 is selected from the images set 1 and there are complex textures of the ground objects in this figure. Because the proposed algorithm connects the regional segments of the skyline based on the similarity degree between the endpoints of the regional segments of the skyline, the edge texture that is not part of the skyline in the image will not influence the detection result.

Fig. 12 shows the images with the noise interfering in the image set 2. The noise brings some unnecessary edge features in the image. Since the algorithm calculates the average brightness of the regions to filter the skyline candidate points, the noise may not generate the regional segments of the skyline. The proposed algorithm can obtain the correct skyline profile under the noise interfering.

Fig. 13 presents the continuous shots in the image set 3. It can be seen that the brightness of image darken gradually. The proposed algorithm works well with the varying illumination.

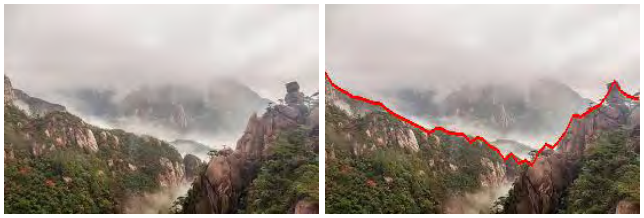
Fig. 14 shows the sunset images in the image set 4, it can see that the proposed algorithm accurately obtains the skyline profile using connecting skyline segments, although there is the interference from the setting sun.

Fig. 15 is selected form the image set 5. It is rainy day and the water drop attached to the camera lens. The skyline profile can be obtained under the interference from water drop.

In Fig. 16, the images in foggy day in the image set 6. We use the same program to generate above detection results.

TABLE 1. Percentage accuracy of algorithm for test images.

	The skyline shape	The brightness of the image	The complexity of the image	The noise interfering with the image signal	Total number of images	Number of images with accurate identification	Percentage accuracy
Single image					200	193	96.5%
Image Set 1	Nonlinear	High	High	Medium	2185	2137	97.8%
Image Set 2	Linear	Low	Medium	High	2170	2037	93.8%
Image Set 3	Linear	Low	Low	Medium	2641	2594	98.2%
Image Set 4	Nonlinear	High	Medium	Low	3910	3813	97.5%
Image Set 5	Linear	Medium	High	High	2706	2627	97.1%
Image Set 6	Nonlinear	Medium	Low	Low	7051	6927	98.2%
Total					20863	20328	97.4%



(a) (b)

FIGURE 18. Incorrect detection result due to the low similarity between the skyline segments. (a) The original image, image resolution:275 × 183. (b)Result of skyline detection of (a).

(a) (b)

FIGURE 19. Incorrect detection result with serious noise interfering. (a) The original image, image resolution: 320 × 240. (b)Result of skyline detection of (a).

It mean that the proposed algorithm can detect the skyline profile under different weather conditions.

Fig. 17 contains some of the detection results for the 200 single images. Table 1 lists the statistical results of the percentage accuracy for the images. The shots in image sets having no skyline will not be counted in the statistical results. According to the experimental results, the proposed algorithm correctly finds the skyline profile in 97% of the test images.

The following figures will explain the reasons for the error detection of skyline. Figs. 18 to 20 present some incorrect results of skyline detection. In these images, (a) is the original copy, and (b) is the skyline detection result of the algorithm.

In Fig. 18, the skyline should include the outlet of the middle mountains. However, because the algorithm must connect two regional segments of the skyline with the highest similarity, therefore, the skyline detection result presents the profile with the highest similarity.

Fig. 19 shows that the serious noise interfering generates the false texture as the line in the image. Because the texture fits the regional feature of ''bright above and dim below'', the propose algorithm detects the false texture as the real skyline.

When the skyline is incomplete or shorter in the image, the algorithm may connect several regional segments as the detection result with great length. In Fig. 20, because there is no complete skyline in the image, the algorithm connects the segments generated from the outlet of tree as detection result.

C. SPEED AND RESOLUTION ASSESSMENT

This paper shows that the proposed skyline algorithm satisfies the requirement for processing speed by adjusting the interval between sampling columns. The test platform currently used is as follows: CPU with an Intel Pentium 4 640 running at 3.2 GHz with 4 GB of memory, and a Windows XP operating system. In this experiment, the sampling interval is set to be 1% of the image width, with decimals discarded unconditionally. For example, the sampling interval is set to 3 for an image of 320 × 240. Changing the sampling interval will affect the processing speed of the skyline detection algorithm, such that a larger sampling interval leads to a higher algorithm processing speed.

Fig. 21 shows the skyline detection result for an image of 400 × 266 resolution, with a sampling interval of 4 and

FIGURE 20. Incorrect detection result with incomplete skyline. (a) The original image, image resolution: 1280 × 720. (b)Result of skyline detection of (a).

FIGURE 21. (a) The original image, image resolution: 400 × 266. (b)Result of skyline detection of (a), sampling column interval: 4, and processing time: 3.26 millisecond.

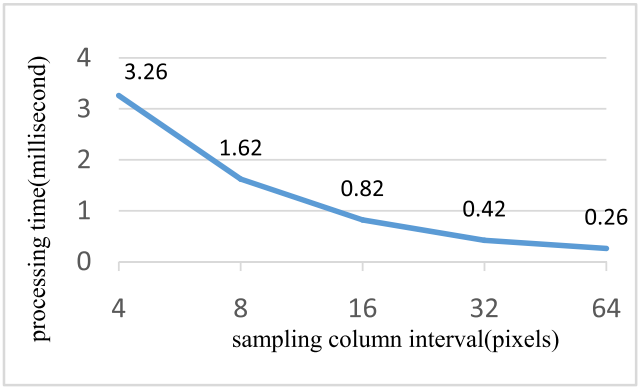


FIGURE 22. Processing time of image in Fig. 17(a) at different sampling column intervals.

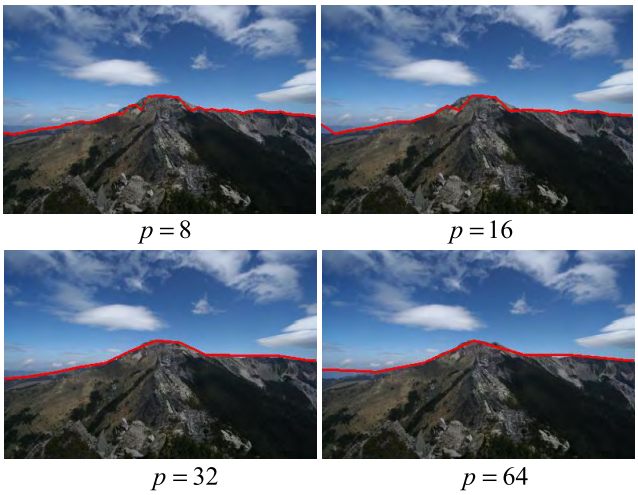


FIGURE 23. Skyline detection results of image in Fig. 17(a) at different sampling column intervals of 16, 32, 48, and 64 pixels.

TABLE 2. Processing times of different skyline detection algorithms.

Algorithm	Processing Time(ms)
Proposed algorithm	3.2 (p=3)
Terminal points detection[11]	1.9
Propeller removal[8]	20
Pixel-statistical[2]	18
Edge-based Hough[9]	38
Projection[7]	152
Processor :P IV,3.2GHz Image Resolution:320x240	

processing time of 3.26 millisecond. Fig. 22 shows the processing time required for the image of Fig. 21(a) at different sampling intervals. It can be seen that the processing is accelerated as the sampling interval is increased. However, increasing the sampling interval inevitably impacts the results of the skyline detection.

The four images of Fig. 23 are the detection results at sampling intervals of 8, 16, 32, and 64 pixels. One can see that as the sampling interval increases, the horizontal resolution of the skyline profile is reduced. However, since the vertical sampling interval increases, the horizontal resolution of the skyline profile is reduced. However, since the vertical resolution is not changed during the analysis of the skyline candidate points, the vertical positions of the candidate points in the sampling column stay the same, and the overall skyline profile after connecting these points is not affected.

Table 2 presents a performance comparison between the different skyline-detection algorithms. It can be seen that the processing speed of the proposed algorithm is much faster than the other four methods below. Although the method of [11] is faster than the proposed algorithm, however, the full skyline profile can not be obtained by only two terminal points of the skyline. If the center of the skyline is protrudent as the mountains or the buildings, the method of [11] can not provide the full skyline. Nevertheless, the result of the proposed algorithm can offer the full skyline profile, and the result can be also applied to the obstacle avoidance.

V. CONCLUSION

In this paper, a vision-based skyline detection algorithm is proposed that can locate nonlinear skyline profiles under different weather and environmental conditions. From the experimental results, it can be seen that in the case of a clear skyline, this algorithm can detect all skyline segments

with varying image characteristics, and shows a curvy skyline profile of mountains or woods. In scenarios with indistinct skylines or cloudy surroundings, this algorithm can identify the region of the sky joining the earth, and estimate a likely location of the skyline. For a total of 20,863 test images, the algorithm achieves an overall accuracy of 97%. The algorithm presented in this paper can provide skyline profile information under differ rent weather and environmental conditions, and adjust its processing time according to system demand in order to meet the requirement of real-time computation. It is suitable to be applied in the vision-based flight control systems of unmanned vehicles.

APPENDIX

TABLE 3. Symbol comparison table.

Symbol	Quantity
$f(x,y)$	input image
w	image width
h	image height
p	sampling column interval
m	index of the sampling column
$I(m\cdot p,y)$	integral image
$G_m(y)$	the pixel gradient of the m-th sampling column
$g_m(y)$	brightness analysis function of the m-th sample column
$B_m(n)$	the n-th skyline candidate region in the m-th sampling column
$r_m(n)$	The upper bound of $B_m(n)$
$s_m(n)$	The lower bound of $B_m(n)$
$\lambda_m(n)$	the y-coordinate of the pixel in $B_m(n)$ with the largest intensity gradient
$b_m(n)$	the y-coordinate of skyline candidate points in $B_m(n)$
$U_m(n)$	average image intensity for the region above the $b_m(n)$
$V_m(n)$	average image intensity for the region below the $b_m(n)$
$A_m(n)$	average image intensity for the region centered at the $b_m(n)$

REFERENCES

[1] 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.

[2] 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.*, vol. 3. Sep./Oct. 2002, pp. 2134–2140.

[3] Y.-F. Dong, Y.-F. Zhang, C. Zhu, and A.-B. Wang, "Extracting sea-sky-line based on improved local complexity," in *Proc. Int. Conf. Comput., Mechatronics, Control Electron. Eng. (CMCE)*, Aug. 2010, pp. 82–85.

[4] H.-Z. Yuan, X.-Q. Zhang, and Z.-L. Feng, "Horizon detection in foggy aerial image," in *Proc. Int. Conf. Image Anal. Signal Process. (IASP)*, Apr. 2010, pp. 191–194.

[5] 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. (ICMLA)*, Dec. 2006, pp. 17–21.

[6] T. G. McGee, R. Sengupta, and K. Hedrick, "Obstacle detection for small autonomous aircraft using sky segmentation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Apr. 2005, pp. 4679–4684.

[7] 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.

[8] 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.

[9] D. Dusha, W. Boles, and R. Walker, "Fixed-wing attitude estimation using computer vision based horizon detection," in *Proc. Fixed-Wing Attitude Estimation Comput. Vis. Based Horizon Detection, 12th Austral. Int. Aerosp. Congr.*, 2007, pp. 1–19.

[10] 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, 2005.

[11] C. C. Chiu and C. T. Lo, "Vision-only automatic flight control for small UAVs," *IEEE Trans. Veh. Technol.*, vol. 60, no. 6, pp. 2425–2437, Jul. 2011.

[12] T. Kuosmanen. (May 17, 2016). *Sunny Evening FPV*. [Online]. Available: https://www.youtube.com/watch?v=wKdpwAy3N0U/

[13] flying408. (Oct. 15, 2016). *Crazepony FX797T FPV Tiny Whoop Camera Overcast Rain Conditions Test*. [Online]. Available: https://www.youtube.com/watch?v=oWMod_2U9fA/

[14] makingrceasy. (Oct. 5, 2014). *Attempting to Fly FPV in Very Foggy and Wet Weather With My MiniQuad*. [Online]. Available: https://www.youtube.com/watch?v=Sytgacrh0p4/

YUN-JIUN LIU (S'11) received the B.S. degree in electrical engineering from National Taiwan Ocean University, Keelung, Taiwan, in 2006, and the M.S. degree in electrical and electronic engineering from the Chung Cheng Institute of Technology, National Defense University, Taoyuan, Taiwan, in 2009, where he is currently pursuing the Ph.D. degree in electrical and electronic engineering.

CHUNG-CHENG CHIU received the B.S. and M.S. degrees in electrical engineering from the Chung Cheng Institute of Technology, Taiwan, in 1990 and 1993 respectively, and the Ph.D. degree in electrical engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2004. In 2005, he joined the faculty of the Department of Electrical and Electronic Engineering, Chung Cheng Institute of Technology, where he is currently a Professor. His research interests include image recognition, image compression, document segmentation, computer vision, and the applications on intelligent transportation system. In 2003, he received the Dragon Golden Paper Award sponsored by the Acer Foundation and the Silver Award of Technology Innovation Competition sponsored by the AdvanTech.

JIA-HORNG YANG received the M.S. degree in electrical and electronic engineering from the Chung Cheng Institute of Technology (CCIT), National Defense University (NDU), Taoyuan, Taiwan, in 2000, and the Ph.D. degree in electrical and computer engineering from the Naval Postgraduate School (NPS), CA, USA, in 2009. After graduating from NPS, he was a Research and Development Officer with the Weapon System Center, CCIT, NDU, where he has been an Assistant Professor with the Electrical and Electronic Engineering Department since 2010. From 2011 to 2012, he was a Leader of the Electro-Optical Graduate School, CCIT. His research interest includes the automatic control system, adaptive control system, digital signal processing, and estimation and tracking.