Rapid detection of camera tampering and abnormal disturbance for video surveillance system

Deng-Yuan Huang a, Chao-Ho Chen b, Tsong-Yi Chen b, Wu-Chih Hu c, Bo-Cin Chen b

a Department of Electrical Engineering, Dayeh University, 168 University Rd., Dacun, Changhua 515, Taiwan, ROC
b Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, 415 Chien Kung Rd., Kaohsiung 807, Taiwan, ROC
c Department of Computer Science and Information Engineering, National Penghu University of Science and Technology, 300 Liu-Ho Rd., Makung, Penghu 880, Taiwan, ROC

Abstract

Camera tampering may indicate that a criminal act is occurring. Common examples of camera tampering are turning the camera lens to point to a different direction (i.e., camera motion) and covering the lens by opaque objects or with paint (i.e., camera occlusion). Moreover, various abnormalities such as screen shaking, fogging, defocus, color cast, and screen flickering can strongly deteriorate the performance of a video surveillance system. This study proposes an automated method for rapidly detecting camera tampering and various abnormalities for a video surveillance system. The proposed method is based on the analyses of brightness, edge details, histogram distribution, and high-frequency information, making it computationally efficient. The proposed system runs at a frame rate of 20–30 frames/s, meeting the requirement of real-time operation. Experimental results show the superiority of the proposed method with an average of 4.4% of missed events compared to existing works.

1. Introduction

Video surveillance systems are widely used in the fields of environmental safety, traffic control, and crime prevention. Important public places such as government agencies, malls, schools, railway stations, airports, military bases, and historical sites are often equipped with digital camera recording systems for video surveillance. However, when cameras are tampered with, the video surveillance system will fail to work properly. Moreover, long-term monitoring of a screen by operators is difficult and many video cameras are frequently left unattended. An intelligent video surveillance system that can automatically analyze live video content, detect suspicious activities, and trigger an alarm to notify operators, is desirable.

Camera tampering is any sustained event which thoroughly alters the image seen by a video camera. Common examples of camera tampering in video surveillance systems are turning the camera lens to point to a different direction (i.e., camera motion) and covering the lens by opaque objects or with paint (i.e., camera occlusion). In general, camera problems are caused by: (1) deliberate actions, such as camera motion and occlusion; (2) weather conditions, such as image blurring due to fogging; (3) abnormal disturbances, such as screen shaking, defocus, color cast, and screen flickering. For automated camera tampering and abnormality detection systems, high reliability and a relatively low false alarm rate are strongly desirable. Most research on the detection of camera tampering and abnormalities has focused on discovering events that move, cover, or defocus the camera [1–6] in a video surveillance system. However, other abnormalities such as screen shaking, fogging, color cast, and screen flickering have received less attention.

Aksay et al. [1] proposed computationally efficient wavelet domain methods for the rapid detection of camera tampering and identified real-life security-related problems. Two algorithms were presented for detecting an obscured camera view and reduced visibility based on a learned background model together with the wavelet transform. However, camera tampering detection based on background modeling often suffers from instability due to varying light source intensity. Ribnick et al. [2] presented an approach to identify camera tampering by detecting large differences between older and more recent frames in video sequences, which are separately stored in two buffers, labeled as the short-term pool and the long-term pool, respectively. Three measures of image dissimilarity are then used to compare the frames to determine whether camera tampering has occurred. However,
several preset thresholds are required and need to be tuned manually for optimal performance.

Sağlam and Temizel [3] proposed adaptive algorithms to detect and identify abnormalities in video surveillance when the camera lens is defocused, moved, or covered. In their method, background subtraction is utilized to build the absolute background, which is used to determine camera tampering types. In the detection of camera defocus, the discrete Fourier transform is used and then a Gaussian windowing function is applied to eliminate low-frequency content. By comparing the high-frequency components of the current frame image and its background, a defocused camera view can be detected. In the detection of a moved camera, a delayed background image is built and compared with the current background using a preset criterion. The detection of a covered camera is done using the peak histograms of the current frame and its background. However, many thresholds must be set.

Lin and Wu [4] identified camera tampering by detecting edge differences and analyzing the grayscale histograms between current and previous frames. An adaptive non-background model image is compared with both incoming video frames and an updated background image for edge difference detection and abnormality justification. Three types of camera tampering and abnormality, namely occlusion, defocus, and motion, were detected in a timely fashion with an overall recognition rate of 94% in their test scenarios. However, differentiating between camera defocus and motion may be unstable if only edge difference information is used. A detection method for camera tampering was also proposed in [5]. In more recent years, a method that uses an adaptive background codebook model was utilized for classifying camera tampering into displacement and obstruction types [6].

In general, camera tampering may indicate that a criminal act might be happening. Detecting abnormalities and triggering alarms to notify operators may thus decrease crime. This study thus develops a system for the rapid detection of camera tampering and various abnormalities in a video surveillance system. To achieve this goal, a computationally efficient method for the rapid detection of various abnormalities, including screen shaking, fogging, color cast, and screen flickering, is proposed. Camera motion, occlusion, and defocus are also detected, which will be compared with existing works.

The rest of this paper is organized as follows. The proposed method for detecting camera tampering and various abnormalities is introduced in Section 2. Experimental results are provided to demonstrate the performance of the proposed method in Section 3. Finally, concluding remarks are given in Section 4.

2. Proposed method

Fig. 1 shows a flowchart of the proposed method for the detection of camera tampering and other abnormalities, including fogging, defocus, color cast, and screen flickering. Screen shaking is first detected to determine whether the input images can be used to build absolute backgrounds. In this work, two backgrounds with a delay of \( n \) frames, i.e., \( B_t \) and \( B_{t-n} \), are built when video frames are stable; otherwise, the alarm for screen shaking is triggered. Then, the difference between backgrounds \( B_t \) and \( B_{t-n} \) is evaluated to determine the types of camera tampering and various abnormalities. If the difference between them is larger than a threshold \( \theta_B \), camera motion or occlusion is determined; otherwise, fogging, defocus, color cast, or screen flickering is determined. Finally, a background update is carried out to timely respond to changes in the input video frames.

In Section 2.1, the method of detecting screen shaking is described. Background modeling and updating are introduced in Section 2.2 and 2.3, respectively. Then, the method of evaluating the background (i.e., \( B_t \) and \( B_{t-n} \)) difference is explained in Section 2.4. Finally, the methods of detecting camera motion, occlusion, and other abnormalities, such as fogging, defocus, color cast, and screen flickering, are given in Section 2.5 and 2.6, respectively.

2.1. Detection of screen shaking

Screen shaking is often caused by wind or vibrations from nearby vehicles. Screen shaking makes the absolute background unstable. As shown in Fig. 2(b) and (d), for shaking frames, the number of pixels with larger gray intensities in a frame-difference image is higher than that of pixels whose gray intensities are smaller. However, for stable frames, the number of pixels with smaller gray intensities in a frame-difference image is higher than that of pixels whose gray intensities are larger. Based on this observation,
the decision rule for determining whether the frame is shaking or stable is derived as [7]:

\[
\begin{align*}
\text{if } N_s > h_2 & \quad \text{frames are shaking} \\
\text{else} & \quad \text{frames are stable}
\end{align*}
\]

(1)

where

\[
N_s = \sum_{x,y} s(x,y), \quad \text{where } s(x,y) = \begin{cases} 1, & |I_t(x,y) - I_{t-1}(x,y)| > \theta_1 \\ 0, & \text{otherwise} \end{cases}
\]

(2)

where \(I_t(x,y)\) and \(I_{t-1}(x,y)\) denote the gray intensities of pixel \((x,y)\) at frame \(t\) and frame \(t-1\), respectively, and \(\theta_1\) and \(\theta_2\) are two thresholds, and \(N_s\) represents the number of pixels whose values are larger than \(\theta_1\) in frame-difference images. The frames at \(t\) and \(t-1\) are considered to be shaking if \(N_s\) is larger than \(\theta_2\), as shown in Eq. (1). However, for some special shaking cases, \(N_s\) may be below \(\theta_2\) due to the varying shaking speed of frames. To detect frame shaking more accurately, an iterative process for checking whether \(N_s\) is larger than \(\theta_2\) is required. The screen is considered to be shaking if \(N_s > \theta_2\) in a given period, \(T\) is the total number of checks that is set to 12 times per second. Thresholds \(\theta_1\) and \(\theta_2\) can be determined from statistical data obtained from experiments, though they may vary with the content of a frame-difference image. The modified decision rule for determining whether the frame is shaking is:

\[
\begin{align*}
\text{if } N_{(N_s > \theta_2)} > T & \quad \text{frames are shaking} \\
\text{else} & \quad \text{frames are stable}
\end{align*}
\]

(3)

It is noted that “frames are shaking” means that the background of frame \(t\) has moved relative to that of frame \(t-1\).

2.2. Modeling of absolute background

Motion detection in video sequences focuses on detecting regions corresponding to moving objects. Methods for this task can be roughly categorized as (1) background subtraction [8,9], (2) temporal differencing [10–12], and (3) optical flow approaches [13,14]. Background subtraction is suitable for detecting moving objects because the backgrounds in video streams are often stationary. In this paper, background modeling based on the temporal distribution of grayscales is built for each point, where the grayscale value with the maximum occurrence probability is assigned as the grayscale value of the absolute background. For details of background modeling, please refer to our previous study [15]. Here, the method of background modeling is briefly described below.

To facilitate the process of background extraction, a distribution of gray levels for a fixed point \(p(x,y)\) in consecutive frames of a video sequence is first built. Then, the probability of occurrence, also called the appearance probability (AP), for each gray level at that point is calculated. Next, gray levels at that point are grouped and assigned to classes, where each class comprises a certain range of gray levels centered at a specific gray level. Finally, the absolute background is built using the gray level with the maximum AP for each point. For the proposed method of background modeling, the mean and variance of the \(k\)th class for each point \(p(x,y)\) at next frame \(t+1\), i.e., \(\mu_{t+1}^k(p)\) and \(\Sigma_{t+1}^k(p)\), are temporally updated using Eq. (4) and Eq. (5), respectively.

\[
\mu_{t+1}^k(p) = \frac{N_k^t(p) \times \mu_k^t(p) + I_t(p)}{N_k^t(p) + 1}
\]

(4)
where \( N_k(p) \) is the number of points for the \( k \)th class at current frame \( t \), and \( I_t(p) \) is the grayscale value of point \( p(x, y) \). In this paper, the class is the bin grouping of grayscale values for each point.

In the proposed background modeling, the grayscale value corresponding to the class with the maximum occurrence probability is then assigned as the grayscale value of the background. The appearance probability \( AP_k(p) \) of the \( k \)th class for each point is calculated using Eq. (6). The grayscale value of the background is then assigned as the grayscale value of the background. The grayscale value of the background, determined using Eq. (7).

**2.4. Evaluation of difference of backgrounds**

Prior to detecting the events of camera tampering and various abnormalities, the difference of the input and background images is calculated using the background subtraction scheme. If the difference between them is large, it implies the possibility of camera motion or occlusion. However, it is undesirable for pedestrians walking in front of the lens to trigger an alarm. Using only the input image and its background image to detect camera tampering is not reliable. To improve reliability, two absolute backgrounds with a delay of \( n \) frames, i.e., \( B_t \) and \( B_{t-n} \), are built. In this work, \( n = 30 \) (obtained from experiments) is used. To determine how large the difference of backgrounds is, the following method is proposed.

\[
\sum_{x,y} s(x,y) \geq \theta_b \cdot N_t
\]

where

- \( s(x,y) = \begin{cases} 1, & B_t(x,y) \neq 0 \\ 0, & \text{otherwise} \end{cases} \)

where \( B_t(x,y) = B_{t-n}(x,y) \) is the background difference at pixel \((x, y)\), \( B_t(x,y) \) and \( B_{t-n}(x,y) \) are the gray intensities of pixel \((x, y)\) at background \( t \) and background \( t - n \), respectively, \( N_t \) is the image size, and \( \theta_b \) is a threshold (set to 0.7 from experiments). If Eq. (12) is satisfied, camera motion or occlusion is detected and the corresponding alarm is triggered; otherwise, fogging, defocus, color cast, and screen flickering is checked (see Fig. 1).

**2.5. Detection of camera motion and occlusion**

Fig. 3 shows the proposed method for the detection of camera motion and occlusion. In this work, camera motion includes events that move the camera or make it point to a different direction and camera occlusion includes events that cover the lens by opaque objects or with paint. To achieve more accurate detection of camera occlusion, two decision rules must be satisfied using histogram analysis and edge detection of the current frame and its background. If these two decision rules are satisfied, lens covering...
and painting are considered; otherwise, the camera is detected as having been moved.

Fig. 4 shows histogram distributions for normal, lens covered, lens painted, and background images. The distributions in Fig. 4(b) and (c) are more concentrated than those in Fig. 4(a) and (d). Based on this observation, the number of pixels around the maximum histogram in the obscured image (see Fig. 4(b)) and its background (see Fig. 4(d)) can be used to determine whether the lens has been covered or painted. Therefore, the first decision rule using histogram analysis is:

\[
\frac{\sum_{k=-n}^{n} H_{f_0+k}}{CF} \geq \frac{\sum_{k=-n}^{n} H_{f_0+k}}{BI} \times \theta_{\text{occlusion}}
\]  

(13)

where subscripts CF and BI denote the current frame (i.e., obscured image) and its background image, respectively, \( f_0 \) is the grayscale value corresponding to the maximum histogram in the obscured image, and \( H_{f_0} \) is the histogram at the grayscale value of \( f_0 \). The thresholds of \( n = 2 \) and \( \theta_{\text{occlusion}} = 1.5 \) are set from experiments.

Further analysis of the edge information of an obscured image and its background image, as shown in Fig. 5, reveals that edge information is lost when a frame is obscured. Therefore, the second decision rule using edge information is proposed as:

\[
S_{CF} \leq S_{BI} \times \theta_{\text{edge}}
\]  

(14)

where \( S_{CF} \) and \( S_{BI} \) denote the numbers of edge pixels in the obscured image and the background image, respectively, and \( \theta_{\text{edge}} \) is a proportional ratio, set to 0.7 from experiments. Hence, when the criteria in Eq. (13) and (14) are both satisfied, the event is considered as lens covering or painting; otherwise, camera movement is detected.

2.6. Detection of other abnormalities

Other abnormalities in this work include fogging, defocus, color cast, and screen flickering. The detection methods are proposed and described in subsections 2.6.1 to 2.6.4. Generally, both blurring and fogging can lead to edge reduction but the causes of them are different. Fogging is due to weather condition while defocus could be resulted from camera damage, mist, or water droplets. For a fogging image, the color saturation is greatly attenuated due to the scattering characteristics of airlight but it is not the case for a blurring image. Moreover, the distribution of grayscale histogram for a fogging image is more concentrated in the central portion than that of a normal image but it cannot be deduced from a blurring image. Therefore, it is necessary to distinguish the cases of blurring and fogging for an image prior to the detection of them. Fig. 6 shows...
comparisons with existing works for the detection of camera motion, occlusion, and defocus were carried out. The results show the superiority of the proposed system with an average of 4.4% of missed events, indicating its feasibility.

Since the proposed method is based on the analyses of brightness, edge details, histogram distribution, and the content of high-frequency components, it is computationally efficient. Moreover, the proposed system runs at a frame rate of 20–30 frames/s, meeting the requirement of real-time operation.

Acknowledgements

This work was partially supported by the National Science Council of Taiwan under Grants NSC102-2221-E-151-042 and NSC-102-2221-E-212-015.

References