Real-Time Active Tampering Detection of Surveillance Camera and Implementation on Digital Signal Processor

Daw-Tung Lin
Dept. of Computer Science and Information Engineering
National Taipei University
New Taipei City, Taiwan
dalton@mail.ntpu.edu.tw

Chung-Han Wu
Institute of Electrical Engineering
National Taipei University
New Taipei City, Taiwan
i04577@hotmail.com

***Abstract*—Active and automatic camera tampering detection is more and more important, especially for wide area surveillance systems with large amount of video cameras installed. Generally, a camera functions for video recording and lacks the ability to detect whether tampering has occurred. Most tampering solutions are passive, as evidenced by many surveillance systems frequently left unattended. Tampering detection of real-time automated cameras is thus of priority concern important for timely operator warning. This work describes a novel algorithm for camera tamper detection scheme. Developed for video surveillance and security applications, the proposed approach identifies camera tampering by detecting large edge differences and grayscale histogram comparisons between current and previous frames. The optimized algorithm is implemented on PC and TI DAVINCI DM6437 DSP as well. Extensive experiments were performed. Experimental results indicate that the proposed approach can detect various tampering cases. The overall recognition performance was 94%.**

Keywords -- Camera tampering detection; video surveillance; camera occlusion; camera motion; camera defocusing

I. INTRODUCTION

Most public venues have installed many surveillance cameras, including airports, train stations, banks, schools and markets. Such surveillance systems are frequently left unattended. Tampering detection has been extensively studied. For instance, Aksay *et. al.* [1] introduced computationally efficient wavelet domain methods to ensure rapid detection of camera tampering and identified real-life security alarm problems. Algorithms were developed for detecting camera occlusion and reduced visibility based on a background model and wavelet transform. However background model based camera tampering detection algorithms are often unstable owing to various lighting conditions. Ribnick *et. al.*[2] used two stored buffer frames (short-term pool and long-term pool) as reference images to determine whether camera tampering has occurred. Nevertheless, any dissimilarity between the short-term pool and the long-term pool is flagged as tampering. Saglam and Temizel [3] provide a simple, low computational cost alternative. Pedro *et. al.* [4] developed an edge background model to detect camera tampering. However, for large objects or a crowd moving in front of the camera, the image characteristics are significantly altered, causing false alarms. Harasse *et. al.* [5] discussed a camera dysfunction detector designed mainly for surveillance systems inside vehicles. Among the other external parameters that can also influence the tampering results include illumination, weather conditions or voluntary actions. These parameters must be considered to prevent or at least detect these circumstances. This work focuses on active camera tampering detection, typically referred to as sabotage. This work presents a novel detection approach for camera tampering. An adaptive non-background model image is compared with incoming video camera frames and with an updated background image. Additionally, the moving areas of an image are monitored, along with a region based operation undertaken to reduce false alarms. This adaptive method is robust to objects moving in front of a camera. This work detects three types of camera tampering in real-time and possesses low false alarm. The proposed system is implemented on a TIDM6437 platform. Several optimization techniques, including instruction parallelism and multiple data packing in a single instruction, and floating-point arithmetic optimization, are considered. Importantly, the proposed detection algorithm is superior to conventional ones in its ability to automatically reply.

The rest of this paper is organized as follows. Section 2 introduces the proposed non-background camera tampering detection method utilizing the edge difference information and justifying for different types of abnormal situations. Section 3 then describes how defocusing and motion camera view are further identified by histogram analysis. Next, Section 4 summarizes the implementation and optimization of the proposed algorithm on the DM6437 DSP platform, followed by a summary of those results in Section 5. Conclusions are finally drawn in Section 6, along with recommendations for future research.

II. NON-BACKGROUND CAMERA TAMPERING DETECTION METHOD

Figure 1 shows the flowchart of the proposed camera tampering detection system and is described in detail as follows. Edge feature is one of the significant characteristics for justifying the preliminary scene changes as means of detecting camera tampering. All captured images are converted into grayscale images. Then, Canny edge detection algorithm is applied for each grayscale image. Let F_n denote the result of Canny edge detection of frame n. The proposed algorithm first establishes still edge information of the scene background for T_f frames and initializes continuous accumulating edges for T_f frames as shown in Equation (1). As can be observed in Equation (1), the result of AND operation of two

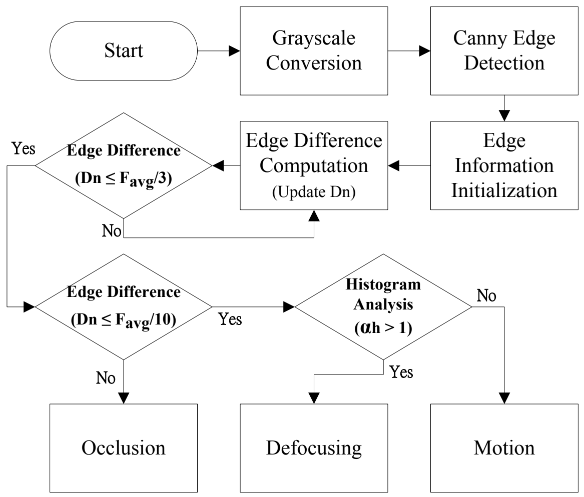

Start
Grayscale Conversion
Canny Edge Detection
Edge Information Initialization
Edge Difference Computation (Update Dn)
Edge Difference (Dn ≤ Favg/3)
Yes
No
Edge Difference (Dn ≤ Favg/10)
Yes
No
Histogram Analysis (αh > 1)
Yes
No
Occlusion
Defocusing
Motion

Figure 1. System Flowchart

consecutive frames of edge images remains the same if there is no motion. For the sake of stabilization, we established the scene edge information for a specified number of seconds; and T_f refers to the total number of frames. According to our experiments, 5 seconds is the most appropriate time. Too short of a time interval produces too little edge information, leading to a high false alarm rate. Thus, the average of edge information F_{avg} of accumulating T_f+1 frame is used as the judgment criteria as computed by Equation (2).

$$F_{sum} = \sum_{n=0}^{T_f} (F_n \,\&\, F_{n+1}) \tag{1}$$

$$F_{avg} = F_{sum}/(T_f + 1) \tag{2}$$

Once the initialization step has been finished and obtain F_{avg}, the edge difference D_n is then computed as the AND operation of the current frame edge image (F_n) and the previous frame edge image (F_{n-1}) as shown in Equation (3). Tampering situation occurs when the edge difference D_n changes dramatically comparing to the reference criteria as indicated in Equation (4). We define the occlusion tampering occurs whenever one-third of the camera scene is covered or blocked. If D_n is greater than or equal to $F_{avg}/3$, we treat it as normal situation. Otherwise, if D_n is further less than $F_{avg}/3$ and greater than $F_{avg}/10$, it indicates there is significantly large enough amount of edge changes between the most recent two image frames. In other words, the occlusion condition happens. Suppose D_n is less than $F_{avg}/10$, it means almost the whole image area is different than the initial one, and then it is possible that motion or defocusing tampers occur. We will need to further justify between these two conditions in the Section 3.

$$D_n = F_{n-1} \,\&\, F_n \tag{3}$$

$$\begin{cases} \text{Normal}, & D_n > F_{avg}/3 \\ \text{Occlusion}, & F_{avg}/10 < D_n \le F_{avg}/3 \\ \text{Motion or Defocusing}, & \text{otherwise} \end{cases} \tag{4}$$

Figure 2 illustrates various camera tampering situations based on the proposed edge difference comparison method. As shown in Figure 2(a) and (b), the edges do not significantly differ in terms of the event. Figure 2(c) shows motion

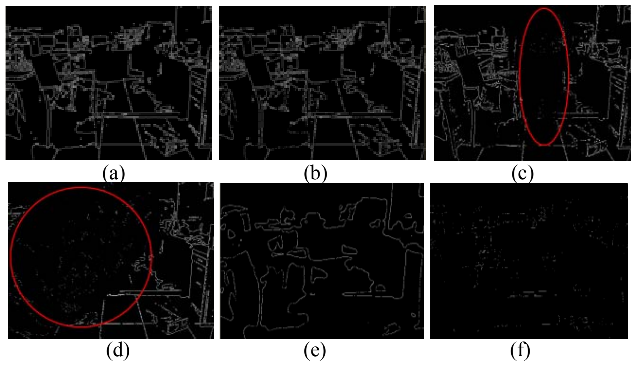


Figure 2. Camera tampering event types: (a) the current frame edge image; (b) edge difference (Eq. (3)) with non-sabotage (c) a fast moving object appears; (d) object occlusion on camera view is detected; (e) edge difference of defocusing situation; (f) edge difference of motion situation.

object in front of the camera, this movement does not cause false alarms because the system is running fast. Figure 2(d) indicates that a hand covering the front of the camera is considered to be occlusion owing to reduced edges information. Figures 2(e) and (f) show the images with reduced edges owing to defocusing. The edge difference of camera defocusing situation is larger than those of the camera motion events. By utilizing the proposed Equations (3) and (4), camera tampering can be initially detected by comparing the current frame and the previous frame in terms of edge detection. However, among the external factors that affect the results include light interference, crowds of people and camera shaking, thus necessitating the importance of camera tampering and surveillance systems with a low false alarm is very important.

III. Histogram Analysis And Justification For Motion And Defocusing Sabotage

As described in the previous section, we are able to identify the camera occlusion sabotage by comparing D_n value. However, D_n is less than $F_{avg}/10$ in the case that the whole image area is different than the initial one. Therefore, we need to further justify whether it is in motion or defocusing conditions by grayscale image histogram analysis. Aksay *et. al.* [1] developed an algorithm to detect the camera occlusion tampering. While calculating the histograms of the current frame F_n and the previous frame F_{n-1}, that algorithm was implemented in two steps. If both steps are satisfied, the camera view is assumed to be occluded. We also discovered that the defocused edge image F_n has higher histogram values in a specific range than those of the previous focused edge image frame F_{n-1} does because defocusing condition makes most of the scene becoming blurred. As is anticipated, a significant portion of F_n has a shadowy outline color or darkens. Therefore, we compare a specific range of the histograms of the current frame F_n and the previous frame F_{n-1}. Typically, the lower gray value portion and higher gray value portion of the histogram are chosen. Two thresholds Th_1 and Th_2 are selected by extensive experiments with $0 < Th_1 \le 96$ and $160 \le Th_1 < 256$. A closer threshold to the lower bound implies a higher sensitivity. Notably, $Th_1=72$ and $Th_2=168$ normally generates satisfactory results. Next, the statistical histogram values $H_{F_n}(x)$ and $H_{F_{n-1}}(x)$ of frame F_n and F_{n-1} are calculated, respectively, and compared by Equation (5), where α_h is a constant value based on various applications. Additionally, the sum of histogram values between *0* to Th_1 and Th_2 to *255* are reduced when defocusing occurs. This algorithm helps to distinguish between motion and defocusing for $\alpha_h \leqq 1$ and $\alpha_h>1$, respectively. Figure 3 illustrates camera tampering scenarios based on the justification (Equation (5)).

$$\sum_{x=0}^{Th_1} H_{F_{n-1}}(x) + \sum_{x=Th_2}^{255} H_{F_{n-1}}(x) > \alpha_h\left(\sum_{x=0}^{Th_1} H_{F_n}(x) + \sum_{x=Th_2}^{255} H_{F_n}(x)\right) \tag{5}$$

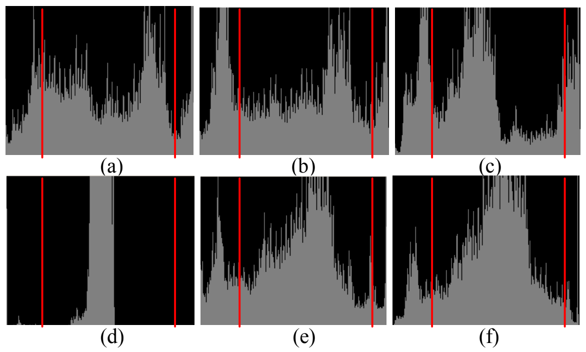


Figure 3. Histogram analysis for various camera tampering scenarios (x-axis: grey level, y-axis: pixel number): (a) No objects are in front of the camera; (b) Moving objects appear in front of the camera; (c) & (d) Camera occlusion tampering; (e) Motion tampering; (f) Defocusing tampering.

Figure 3 shows a case study of system operation. Figure 3(a) shows no moving objects in front of the camera which presents evenly distributed histogram values. Figure 3(b) shows the histogram when objects appear in front of the camera with movement that does not cause false alarms, because moving objects do not lead to substantial changes in the histogram value between Th_1 and Th_2. According to Figures 3(c) and 3(d), the camera is occluded and the histograms show the difference. However, histogram information is not used to detect occlusion which easily brings about false alarms. Therefore, only the edge information mentioned in Section 2 that detects occlusion is relatively stable. Figure 3(e) shows camera motion histogram in Th_1 and Th_2, which is similar to Figure 3(a). Because the camera motion is still obviously observed in this scene, monotonous scenes increase false alarms using the histogram detection method. In Figure 3(f), camera defocusing in the histogram in Th_1 and Th_2 is reduced (α_h>1). Therefore, using histogram analysis (Equation (5)) to detect the motion and defocusing tampering can enhance camera tampering detection accuracy. Under defocusing and motion on both sides of the histogram threshold values, significant changes are generated; this feature is characterized by its ability to distinguish between different tamper detection capabilities.

IV. Implementation And Optimization On DM6437

This work implemented and optimized the proposed active camera tampering algorithm on the digital signal processor TI TMS320C6437, due to the low cost DSP of TI DaVinci™ and C6000 families of digital signal processors. TI DM6437 is a high-performance fixed-point DSP with 32K bytes of embedded RAM for instructions and 80K bytes of embedded RAM for data memory. It runs at a clock speed of 594MHz and provides up to 4,800 million MAC (multiply-accumulate) operations per second. Additionally, the 128 K bytes static memory can be accessed by CACHE or SRAM [11]. Figure 4 shows our setup of a TI DM6437 EVM development environment with a target board, a D1 resolution (720×480) camera of YUV NTSC video signal and an output device (monitor).

In this work, the main optimization strategy is in twofold: (1) code porting and instruction level optimization and (2) floating point optimization. In code porting and instruction level optimization, the key functions are rewritten using efficient SIMD pipeline and DSP instructions to increase data throughput. Fortunately, the DSP system has a valuable hardware feature called intra-instruction parallelism (SIMD, VLIW) for software optimization. Single instruction multiple data (SIMD) refers to multiple processing units under the control of a single instruction working on different data. Very long instruction word (VLIW) refers to a processor architecture designed to exploit multiple non-identical function units running in instruction level parallelism. VLIW also considers fixed-point architecture design. The details of optimization procedures are illustrated as follows.

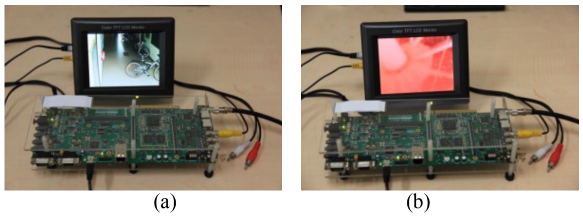


Figure 4. Experimental peripherals: DM6437 EVM board and monitor. (a) non-sabotage case. (b) Occlusion tampering is detected.

The data packing instruction in the DSP algorithm is applied to process 8 pixels (64 bits) at a time for one byte per pixel in the input image. Moreover, the number of branches falls, thus speeding up the execution time. Additionally, software pipelining is utilized to allow for the execution of multiple instructions in a single clock cycle. The proposed convolution algorithm is optimized. Based on intrinsic functions provided by TI, (e.g., LDDW, COMPEQ and MAXU4) that can comply with ANSI C instructions.

The regularization process requires floating point computation. However, calculating floating-point number operations is a heavy burden for a fixed point processor. Hence, the proposed method replaces the division operation with shifting where possible. However, if not a number of powers of 2, the original value cannot be divided using shifting. In this case, the IQmath library (i.e. a highly optimized and high-precision mathematical function library for C/C++ programmers) can be utilized to seamlessly port the bit-true fixed point algorithm into fixed point code on DSP devices [10]. Therefore, in this work, floating-point operations are performed using the IQmath library.

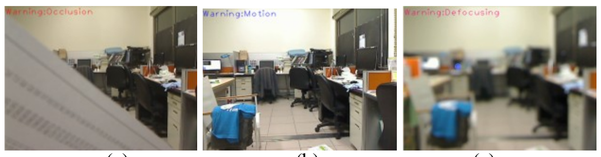
V. Experimental Results

The proposed system was implemented on TI DAVINCI DM6437 DSP and personal computer (PC) with Intel Core 2 Duo 2.4G CPU and 2GB RAM using Visual C++. Experimental results demonstrate that the final proposed system works in real-time at 22 frames per section (FPS) on DM6437 DSP and 26 FPS on PC. The system was tested in indoor and outdoor conditions with images that include lights (turn on and off), illumination changes and background motion (especially individuals and cars). During the sequence recording, the camera was intentionally sabotaged to test the system response. These actions included the three sabotage types: occlusion, motion and defocusing analyzed in this work, as described previously. One hundred tests were performed for each tampering type. Table 1 presents the performance of the proposed camera tampering detection system. The overall correct identification rates were 94% and 93% for PC based solution and DM6437 DSP implementation, respectively.

Figure 5 shows some camera tampering detection capabilities. In Figure 5(a) the lens was partially occluded with a sheet of paper, concealing a part of the scene to the camera. According to Figure 5(b), the camera was shifted to the right in order to partially change the field of view. Finally, Figure 5(c) indicates that the lens setting of the camera changed to defocus the image.

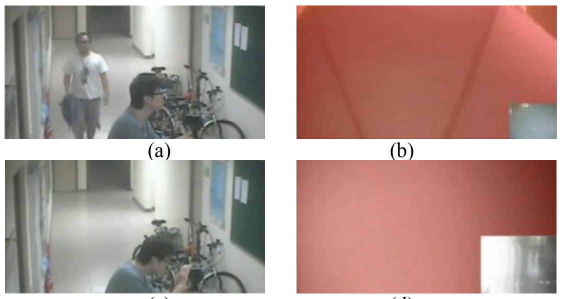
Tampering	Occlusion	Motion	Defocusing	Overall	False alarm
Correct Detection	97%	93%	91%	94%	6

Table 1. Performance of the proposed camera tampering detection system.



(a) (b) (c)

Figure 5. Results of video camera sabotages: (a) Occlusion case; (b) Motion case; and (c) Defocusing case.



(a) (b)

(c) (d)

Figure 6. Results of video camera sabotage on DSP: (a) Normal condition; (b) Occlusion case; (c) Normal condition; (d) Defocusing case.

Figure 6 shows camera tampering detection capabilities on the DSP system. Tampering detection in the DSP system also performs well for various types of sabotages. In the DSP application, a tamper event occurs when the red mask appears on the monitor. Additionally, on the DSP procedure, implementing in real-time requires adjusting the images to 320 x 240 for testing.

VI. Conclusions

This work presents an active and automatic camera tampering detection algorithm that can be implemented in DSP system. The proposed methods are based on edges comparisons of recent and older camera frames and the histogram analysis of a specific range of grayscale values. The overall system is highly robust and efficient. The proposed camera tampering detection system was implemented and optimized on a low-cost DM6437 DSP and can be easily embedded into IP camera devices. Extensive experiments were performed. Experimental results indicate that the proposed approach can detect various tampering cases. The overall recognition performance was 94%. Efforts are underway in our laboratory to test the proposed system under more complex conditions, including adverse outdoor environments (e.g., night, rain, fog or wind), and crowded scenarios. Although the proposed system is simplified as much as possible to reduce the computational time, more complex algorithms could be designed whenever additional features are added. Still, the real-time requirements must be maintained.

References

[1] A. Aksay, A. Temizel and A.E. Cetin, 2007. “Camera tamper detection using wavelet analysis for video surveillance.” Advanced Video and Signal Based Surveillance. London, United Kingdom, pp. 558-562.

[2] E. Ribnick, S. Atev, O. Masoud, N. Papanikolopoulos and R. Voyles, 2006. “Real-Time Detection of Camera Tampering.” Advanced Video and Signal Based Surveillance. Sydney, Australia, pp. 10-10.

[3] A. Saglam and A. Temizel, 2009. “Real-Time Adaptive Camera Tamper Detection for Video Surveillance.” Advanced Video and Signal Based Surveillance. Genova, Italy, pp. 430-435.

[4] P. Gil-Jimenez, R. Lopez-Sastre, P. Siegmann, J. Acevedo-Rodriguez and S. Maldonado-Bascon, 2007. “Automatic Control of Video Surveillance Camera Sabotage.” International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC), Vol. 4528, pp. 222-231.

[5] S. Harasse, L. Bonnaud, A. Caplier and M. Desvignes, 2004. “Automated camera dysfunctions detection.” In IEEE Southwest Symposium on Image Analysis and Interpretation. pp. 36-40.

[6] A.M. Bagci, Y. Yardimci and A.E. Cetin, 2002. “Moving object detection using adaptive subband decomposition and fractional lower-order statistics in video sequences.” European Association for Signal Processing, Vol. 82, No. 12, pp. 1941-1947.

[7] J. Fridrich, 1998. “Image watermarking for tamper detection.” International Conference on Image Processing. Chicago, USA, pp. 404-408 vol.2.

[8] J.H. Wei and K.N. Ngan, 1999. “Integrated shot boundary detection using object-based technique.” International Conference on Image Processing. Kobe , Japan, pp. 289-293 vol.3.

[9] D.K. Roberts, 2002. “Security camera video authentication.” Digital Signal Processing Workshop and the 2nd Signal Processing Education Workshop. Sedona, USA, pp. 125-130.

[10] Z. Nikolić et al, 2007. “Design and implementation of numerical linear algebra algorithms on fixed point DSPs.” EURASIP Journal on Advances in Signal Processing, Vol. 2007, No. 1, pp. 087046.