Tampering detection for low-power smart cameras

Adriano Gaibotti[1,2], Claudio Marchisio[2], Alexandro Sentinelli[2], and Giacomo Boracchi[1]

[1] Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Via Ponzio 34/5, 20133, Milano (MI), Italy
adriano.gaibotti@mail.polimi.it, giacomo.boracchi@polimi.it

[2] STMicroelectronics, Advanced System Technology, Via Camillo Olivetti 2, 20864, Agrate Brianza (MB), Italy
{adriano.gaibotti, claudio.marchisio, alexandro.sentinelli}@st.com

Abstract. One of the most critical issues in smart cameras, used as nodes in wireless multimedia sensor networks (WMSN), is the automatic detection of events that could compromise the correct acquisition of the scene, such as the presence of water on the camera lens, or the displacement of the device, or tampering events. In general, techniques used to identify these type of events are called tampering detection algorithms. We introduce a solution for the tampering detection issue which can be used for low-power and embedded smart cameras, in which the acquisition is done with low frame rate, such as one frame every minute. The solution consists in a segmentation of the acquired scene in adaptive regions, and monitoring some features in each region, in order to find a change in their behavior that could be associated to a tampering event. Experiments show that our solution increase the performances with respect to monitoring the whole scene, with low computational complexity.

Keywords: tampering, defocus, displacement, segmentation, gradient

1 Introduction

[**Giacomo**: Adriano: metti un paio di foto per spiegare il problema che consideriamo.]
Random Toughs:

- Smart cameras, low-power monitoring of scene. Description of the application scenario. Low frame rate.
- Cameras organized in a multimedia network, continuous acquisition and streaming is not feasible
- The problem of false alarms, radio module activation
- Other tampering attacks like obfuscation (??) which might be due to environmental phenomena such as rain, fog and mist over the camera lenses have to be detected by image analysis methods
- Displacement can be perceived by MEMS as well but these device alone are prone to false alarms. Visual inspection is necessary to reduce false alarms

(a) (b)

Fig. 1. Examples of tampering events due to atmospheric phenomena. In Figure 1(a) there is an occlusion due to some snow on the camera lens, while in Figure 1(b) there is a blur due to rain drops on the camera lens.

– constrained environment: algorithms have to operate with a low computational complexity and memory requirement

2 Related Works

[**Giacomo**: Adriano: metti tutte le reference, ciascuna con un commento di una frase per dire che fa ed una frase (o mezza) per dire i problemi che ha (con particolare riferimento all'ambito low-power) Poi le aggiustiamo in maniera organica]

[1]: uses background subtraction methods in order to identify defocus and occlusions, doing a comparison in the wavelet domain for defocus detection and histogram comparison for occlusion detection. Displacement is not treated.

[2]: uses background subtraction methods in order to identify defocus, occlusions, and displacements. Comparison in the Fourier domain for defocus detection, histogram comparison for occlusion detection, comparison between current background and delayed background for displacement detection.

[3]: CUSUM CDT on gradient energy content in order to detect blur in frames

[4]: uses background subtraction methods in order to identify defocus, occlusions, and displacements. Comparison of edges pixels count for defocus detection, entropy comparison for occlusion detection, block matching algorithm for displacement detection.

[5]: comparison between frames belonging to a buffer in order to find high values of dissimilarity, associated to tampering.

[6]: implementation in a FPGA of a solution based on background modeling, histograms comparisons, edges comparisons.

[7]: tampering detection inside a moving vehicle; uses background subtraction methods in order to identify defocus, occlusions, and displacements. Comparison of edges pixels count for defocus detection, entropy comparison for occlusion detection, block matching algorithm for displacement detection.

[8]: monitoring of the number of key points extracted by SURF in order to detect defocus events, partition in blocks and HOG descriptors matching for each block in order to detect occlusions. These types of solutions requires a lot of computations

3 Problem Formulation

[**Giacomo**: Adriano: metti le formule di quello della tesi circa displacemente e out of focus. Poi condensiamo il tutto]

$$z(x) = \mathcal{D}[y](x) = \mathcal{B}[y](x) + \eta(x), \qquad x \in \mathcal{X} \tag{1}$$

$$\mathcal{B}[y](x) = \int_{\mathcal{X}} y(s)h(x,s)ds, \tag{2}$$

$$\mathcal{B}[y](x) = \int_{\mathcal{X}} y(s)h(s-x)ds, \tag{3}$$

$$z_t(x) = \mathcal{D}_t[y_t](x) = \mathcal{B}_t[y_t](x) + \eta(x), \qquad x \in \mathcal{X}. \tag{4}$$

$$z_t(x) = \begin{cases} y_t(x) + \eta(x) \text{ per } t < T^* \\ w_t(x) + \eta(x) \text{ per } t \geqslant T^* \end{cases}, \tag{5}$$

Per $t < T^*$ abbiamo che i frame vengono acquisiti in condizioni di funzionamento normale:

$$z_t(x) = y_t(x) + \eta(x), \forall x \in \mathcal{X}, \text{ per } t = 1, \ldots, T^* - 1.$$

All'istante di tempo $t = T^*$ avviene un evento di tampering, il quale compromette i frame per $t \geq T^*$. In particolare, nel caso di una sfocatura avremo

$$z_t = \mathcal{B}_t[y_t](x) + \eta(x), \forall x \in \mathcal{X}, \text{ per } t \geq T^*,$$

mentre nel caso di uno spostamento della camera avremo

$$z_t = w_t(x) + \eta(x), \forall x \in \mathcal{X}, \text{ per } t \geq T^*.$$

4 Proposed Solution

4.1 Scene Segmentation

4.2 Indicators

[**Giacomo**: Adriano: mettere formule degli indicatori e anche del frame difference qua]

$$g(t) = \mathcal{G}[z_t] = \frac{\sum_{\mathcal{X}} \|\nabla z_t(x)\|_2^2}{|\mathcal{X}|}, \tag{6}$$

In particolare, per il calcolo delle derivate orizzontali abbiamo utilizzato il seguente filtro f_h:

$$f_h = f \circledast \begin{bmatrix} 1 & 0 & -1 \end{bmatrix},$$

mentre per il calcolo delle derivate verticali abbiamo utilizzato il seguente filtro f_v:

$$f_v = f \circledast \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix},$$

dove abbiamo indicato con $\circledast$ l'operatore di convoluzione. Il filtro f, invece, è ottenuto tramite un campionamento della *funzione gaussiana* h, con media 0 e deviazione standard σ

$$h(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right), \tag{7}$$

e ponendo il valore massimo di questa funzione nel centro del filtro. Con questi filtri è possibile calcolare la *norma del gradiente* nel seguente modo:

$$\|\nabla z_t(x)\|_2^2 = \left(z_t \circledast f_h\right)(x)^2 + \left(z_t \circledast f_v\right)(x)^2. \tag{8}$$

Una volta calcolata la norma del gradiente è possibile farne la media come specificato in (6). Il risultato finale è un indicatore *scalare* per ciascun frame acquisito, che può essere monitorato per individuare eventi di sfocature. In particolare ci aspettiamo che l'evento di sfocatura provochi un abbattimento del valore di g.

$$l(t) = \mathcal{L}[z_t] = \frac{\sum_{\mathcal{X}} z_t(x)}{|\mathcal{X}|}, \tag{9}$$

$$\frac{\partial g}{\partial t}(t) = g(t) - g(t-1), \tag{10}$$

$$\frac{\partial l}{\partial t}(t) = l(t) - l(t-1). \tag{11}$$

4.3 Outlier Detection

$$\begin{aligned} \Gamma_{min}^k &= \hat{\mu}_g^k - \gamma\hat{\sigma}_g^k \\ \Gamma_{max}^k &= \hat{\mu}_g^k + \gamma\hat{\sigma}_g^k \end{aligned}, \tag{12}$$

dove $\hat{\mu}_g^k$ indica il valore medio delle osservazioni del training set

$$\hat{\mu}_g^k = \frac{\sum_{\tau=1}^{T_o} \frac{\partial g^k}{\partial t}(\tau)}{T_o},$$

$\hat{\sigma}_g^k$ indica la deviazione standard delle osservazioni del training set

$$\hat{\sigma}_g^k = \sqrt{\frac{1}{T_o - 1} \sum_{\tau=1}^{T_o} \left(\frac{\partial g^k}{\partial t}(\tau) - \hat{\mu}_g^k(\tau)\right)^2}$$

e $\gamma > 1$ è un parametro moltiplicativo ottenuto sperimentalmente.

$$\begin{aligned} \Gamma_{min}^k &= \hat{\mu}_l^k - \gamma \hat{\sigma}_l^k \\ \Gamma_{max}^k &= \hat{\mu}_l^k + \gamma \hat{\sigma}_l^k \end{aligned}, \tag{13}$$

dove $\hat{\mu}_l^k$ indica il valore medio delle osservazioni del training set

$$\hat{\mu}_l^k = \frac{\sum_{\tau=1}^{T_o} \frac{\partial l^k}{\partial t}(\tau)}{T_o},$$

$\hat{\sigma}_l^k$ indica la deviazione standard delle osservazioni del training set

$$\hat{\sigma}_l^k = \sqrt{\frac{1}{T_o - 1} \sum_{\tau=1}^{T_o} \left(\frac{\partial l^k}{\partial t}(\tau) - \hat{\mu}_l^k(\tau) \right)^2}$$

e $\gamma > 1$ è un parametro moltiplicativo ottenuto sperimentalmente.

4.4 Algorithm Summary

[**Giacomo**: Adriano: inserisci qui l'algoritmo e traducilo in inglese. Se riusciamo lo spostiamo prima di tutte le sottosezioni]

5 Experiments

5.1 Dataset Description

[**Giacomo**: Adriano: Prova a mettere qua le info] The proposed method have been tested on a dataset of frame sequences in which there were tampering events. A part of this dataset was taken using a Raspberry Pi with its camera module, and the tampering was introduced by moving the device or putting water on the camera. The other part of this dataset is made of frame sequences taken from web-cams available on the Internet. In these sequences there were a few events of displacements or defocus, but to have a larger set of tampering events we

5.2 Alternative Approaches

- Full
- Adaptive Region
- Voronoi Regions

5.3 Performance Assessment

[**Giacomo**: Adriano: Dire come vengono calcolate le ROC curves TPR e FPR, le cifre di merito insomma, spiegando bene che parametro varia]

[**Giacomo**: Adriano: metti entrambe le ROC curves, affiancate e per bene ed alcuni esempi di sequenze]

Algorithm 1: Blur detection algorithm

Configuration:

1. Extract regions $\{R_k\}, k = 1, \ldots, K$
2. **for** $t = 1, \ldots, T_o$ **do**
3. Acquire frame z_t
4. **for** $k = 1, \ldots, K$ **do**
5. Compute $g^k(t)$, $\frac{\partial g^k}{\partial t}(t)$ for the region R_k
   end
6. Compute $g(t)$, $\frac{\partial g}{\partial t}(t)$
   end
7. **for** $k = 1, \ldots, K$ **do**
8. Define thresholds Γ^k_{min} and Γ^k_{max}
   end
9. Define CDT parameters on $g(t)$ variance

Operational phase:

10. **for** $t = T_o, \ldots, \infty$ **do**
11. Acquire frame z_t
12. Compute $g(t)$, $\frac{\partial g}{\partial t}(t)$
13. $n = 0$
14. **for** $k = 1, \ldots, K$ **do**
15. Compute $g^k(t)$, $\frac{\partial g^k}{\partial t}(t)$ for the region R_k
16. **if** $\frac{\partial g^k}{\partial t}(t) < \Gamma^k_{min} \vee \frac{\partial g^k}{\partial t}(t) > \Gamma^k_{max}$ **then**
17. $n = n + 1$
    end
    end
18. **if** $n \geq K - 1$ **then**
19. z_t is a defocused frame
    end
20. **if** *CDT detect a change on* $g(t)$ *variance* **then**
21. z_t is a defocused frame
    end
    end

Algorithm 2: Displacement detection algorithm

Configuration:

Extract regions $\{R_k\}, k = 1, \ldots, K$
for $t = 1, \ldots, T_o$ **do**
    Acquire frame z_t
    for $k = 1, \ldots, K$ **do**
        Compute $l^k(t)$, $\frac{\partial l^k}{\partial t}(t)$ for the region R_k
    end
end
for $k = 1, \ldots, K$ **do**
    Define thresholds Γ^k_{min} and Γ^k_{max}
end

Operational phase:

for $t = T_o, \ldots, \infty$ **do**
    Acquire frame z_t
    $n = 0$
    for $k = 1, \ldots, K$ **do**
        Compute $l^k(t)$, $\frac{\partial l^k}{\partial t}(t)$ for the region R_k
        if $\frac{\partial l^k}{\partial t}(t) < \Gamma^k_{min} \vee \frac{\partial l^k}{\partial t}(t) > \Gamma^k_{max}$ **then**
            $n = n + 1$
        end
    end
    if $n \geq K - 1$ **then**
        z_t is a displaced frame
    end
end

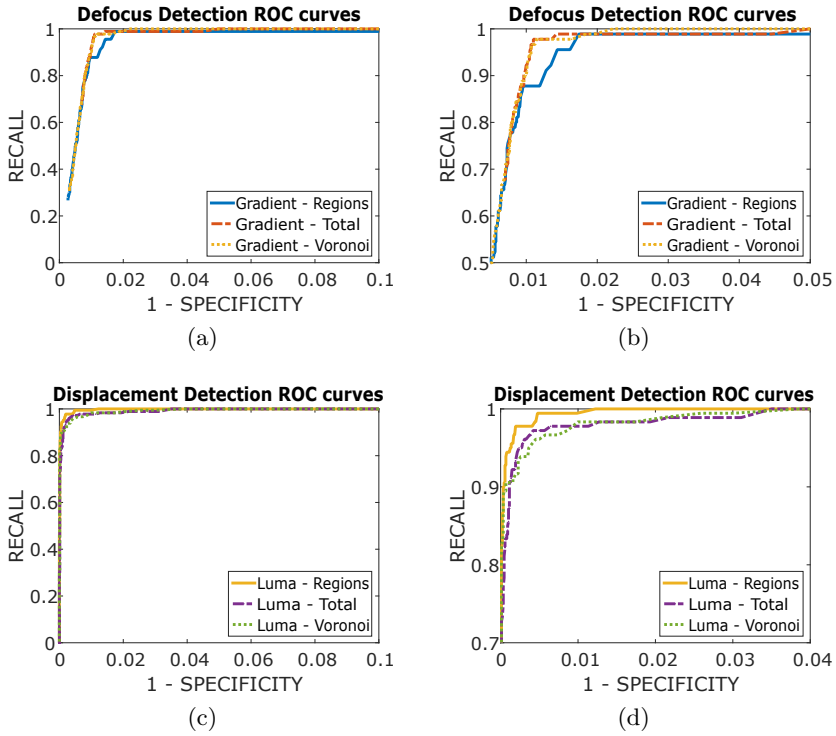

Fig. 2. Examples of tampering events due to atmospheric phenomena. In Figure 1(a) there is an occlusion due to some snow on the camera lens, while in Figure 1(b) there is a blur due to rain drops on the camera lens.

5.4 Discussion

[**Giacomo**: Adiano: Aggiungi qua la complessitá computazionale]

6 Conclusion

[**Giacomo**: Adriano: butta in inglese gli ongoing works (come ultima cosa)] .

Acknowledgments

Authors would like to thank ST for supporting Adriano Gaibotti.

References

1. Anil Aksay, Alptekin Temizel, and A. Enis Cetin. Camera tamper detection using wavelet analysis for video surveillance. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 558–562. IEEE, 2007.
2. Ali Saglam and Alptekin Temizel. Real-time adaptive camera tamper detection for video surveillance. In *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pages 430–435. IEEE, 2009.
3. Cesare Alippi, Giacomo Boracchi, Romolo Camplani, and Manuel Roveri. Detecting external disturbances on the camera lens in wireless multimedia sensor networks. *Instrumentation and Measurement, IEEE Transactions on*, 59(11):2982–2990, 2010.
4. Pedro Gil-Jiménez, R. López-Sastre, Philip Siegmann, Javier Acevedo-Rodríguez, and Saturnino Maldonado-Bascón. Automatic control of video surveillance camera sabotage. *Nature Inspired Problem-Solving Methods in Knowledge Engineering*, pages 222–231, 2007.
5. Evan Ribnick, Stefan Atev, Osama Masoud, Nikolaos Papanikolopoulos, and Richard Voyles. Real-time detection of camera tampering. In *Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on*, pages 10–10. IEEE, 2006.
6. T Kryjak, M Komorkiewicz, and M Gorgon. Fpga implementation of camera tamper detection in real-time. In *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*, pages 1–8. IEEE, 2012.
7. Sebastien Harasse, Laurent Bonnaud, Alice Caplier, and Michel Desvignes. Automated camera dysfunctions detection. In *Image Analysis and Interpretation, 2004. 6th IEEE Southwest Symposium on*, pages 36–40. IEEE, 2004.
8. Theodore Tsesmelis, Lars Christensen, Preben Fihl, and Thomas B Moeslund. Tamper detection for active surveillance systems. In *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pages 57–62. IEEE, 2013.