POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria


# ALGORITMO DI TAMPERING DETECTION OTTIMIZZATO TRAMITE SEGMENTAZIONE DELLA SCENA INQUADRATA

Relatore: Prof. Giacomo BORACCHI
Correlatore: Ing. Claudio MARCHISIO

Tesi di Laurea di:
Adriano GAIBOTTI, matricola 780200

Anno Accademico 2013-2014

*A Sara*

# Sommario

Nel campo dei sistemi di monitoraggio video, uno dei principali problemi è quello di identificare eventi che possano compromettere la corretta ripresa della scena. Può capitare, ad esempio, che dell'acqua piovana si depositi sulla lente della camera, rendendo l'immagine acquisita sfocata, oppure che la camera si sposti, a causa di un intenzionale intervento umano o per eventi naturali quali una raffica di vento, e non riprenda più la scena originale.

Il problema di individuare, in maniera automatica, questo tipo di eventi prende il nome di *tampering detection*. Nella letteratura scientifica questo problema è stato affrontato solamente per applicazioni di *videosorveglianza*, dove la camera opera con frequenze di acquisizione elevate. Tali prestazioni sono possibili solamente su sistemi di monitoraggio con una certa potenza computazionale, e che vengono alimentati a corrente. Lo scopo della tesi è lo sviluppo di un algoritmo di tampering detection per sistemi *embedded* e a basso consumo da utilizzarsi in scenari di monitoraggio. In particolare, l'algoritmo è caratterizzato da un basso carico computazionale ed è pensato per scenari, tipo il monitoraggio ambientale, dove il sistema, per ridurre il consumo energetico, acquisisce e analizza poche immagini al minuto o all'ora (*framerate bassi*). In questi casi scene ad *alta dinamicità*, come una strada in cui passano macchine e pedoni, non permettono di identificare eventi di tampering tramite un confronto tra frame consecutivi. Inoltre, operando a bassi framerate, si verificano variazioni di luminosità, tra immagini consecutive, più sostanziali rispetto al caso di acquisizione continua.
L'algoritmo proposto si basa su indicatori, estratti dalle singole immagini, calcolati a bassa complessità computazionale; tali indicatori vengono monitorati nel tempo attraverso tecniche *sequenziali* e di *outlier detection* per identificare l'istante in cui avviene l'evento di tampering. Data l'alta variabilità degli indicatori utilizzati, abbiamo introdotto una fase di *segmentazione* della scena, in modo da limitare l'analisi ad alcune regioni specifiche: questo permette di migliorare le prestazioni dell'algoritmo e diminuire il numero di falsi allarmi.

La tesi è stata svolta durante uno stage presso *STMicroelectronics*, interessata a sviluppare algoritmi intelligenti di elaborazione immagini da in-

tegrare nei propri dispositivi embedded, e a scenari di impiego per questi. Sono stati messi a punto diversi sistemi di acquisizione operanti a diversi framerate, che hanno permesso di generare i dataset per testare l'efficacia della soluzione proposta e, in particolare, dei vantaggi nell'utilizzo della segmentazione a supporto del tampering detection.

# Ringraziamenti

Ringrazio .....

# Indice

**Sommario** **iii**

**Ringraziamenti** **vii**

**1 Introduzione** **3**

**2 Stato dell'arte** **5**
2.1 Modello della camera . . . . . 5
2.2 Monitoraggio video: concetti e terminologia . . . . . 5
2.3 Tampering Detection . . . . . 6
2.3.1 Tecniche basate su confronto di background . . . . . 7
2.3.2 Tecniche basate su monitoraggio sequenziale . . . . . 10

**3 Impostazione del problema di ricerca** **13**
3.1 Modello delle osservazioni . . . . . 13
3.1.1 Scena e posizione della camera . . . . . 13
3.1.2 Sfocatura . . . . . 14
3.1.3 Spostamento della camera . . . . . 16
3.1.4 Occlusione e guasti della camera . . . . . 17
3.2 Tampering detection . . . . . 18

**4 Soluzione proposta** **19**
4.1 Estrazione dei descrittori del cambiamento . . . . . 19
4.2 Algoritmo di segmentazione . . . . . 19
4.3 Monitoraggio one-shot . . . . . 19
4.4 Monitoraggio sequenziale . . . . . 19

**5 Realizzazioni sperimentali e valutazione** **21**
5.1 Acquisizione dei dataset . . . . . 21
5.2 Risultati . . . . . 21

**6 Direzioni future di ricerca e conclusioni** **23**

**Bibliografia** **25**

# Elenco delle figure


2.1 Sistema di monitoraggio video . . . . . 6
2.2 Comportamento della trasformata di Fourier nel caso di sfocatura . . . . . 9
2.3 Sequenza di frame acquisiti a 30 fps . . . . . 10
2.4 Sequenza di frame acquisiti ogni 30 secondi . . . . . 11

3.1 Esempi di sfocature . . . . . 14
3.2 Sequenza di otto frame consecutivi acquisiti ogni minuto . . . 16
3.3 Esempio di spostamento della camera . . . . . 16
3.4 Esempi di occlusione . . . . . 17

# Elenco delle tabelle

# Capitolo 1

# Introduzione

*"Terence: Mi fai un gelato anche a me? Lo vorrei di pistacchio.*
*Bud: Non ce l'ho il pistacchio. C'ho la vaniglia, cioccolato, fragola, limone e caffè.*
*Terence: Ah bene. Allora fammi un cono di vaniglia e di pistacchio.*
*Bud: No, non ce l'ho il pistacchio. C'ho la vaniglia, cioccolato, fragola, limone e caffè.*
*Terence: Ah, va bene. Allora vediamo un po', fammelo al cioccolato, tutto coperto di pistacchio.*
*Bud: Ehi, macché sei sordo? Ti ho detto che il pistacchio non ce l'ho!*
*Terence: Ok ok, non c'è bisogno che t'arrabbi, no? Insomma, di che ce l'hai?*
*Bud: Ce l'ho di vaniglia, cioccolato, fragola, limone e caffè!*
*Terence: Ah, ho capito. Allora fammene uno misto: mettici la fragola, il cioccolato, la vaniglia, il limone e il caffè. Charlie, mi raccomando il pistacchio, eh."*

Pari e dispari

Negli ultimi anni le applicazioni di tipo multimediale sono aumentate in maniera esponenziale, soprattutto per quanto riguarda i contenuti video. L'abbassamento dei prezzi e delle dimensioni dei *sensori* e delle componenti hardware

# Capitolo 2

# Stato dell'arte

In questo capitolo elenchiamo quelle che sono le principali tecniche, presenti nella letteratura scientifica, utilizzate per identificare tentativi di manomissione su camere di videosorveglianza.

## 2.1 Modello della camera

## 2.2 Monitoraggio video: concetti e terminologia

Prima di concentrarci sul problema del tampering detection, definiamo, i concetti e i termini che verranno utilizzati nel seguito della trattazione.
Lo scenario che consideriamo è quello di una camera che deve riprendere una particolare *scena*. La posizione e l'orientamento della camera determinano l'*inquadratura* della scena. L'acquisizione, da parte della camera, della scena nell'istante di tempo $i$ viene definita *immagine* o *frame* i-esimo. La figura 2.1 illustra questi concetti.
Nel seguito della trattazione useremo una specifica terminologia. Indicheremo con $\mathcal{X}$ l'insieme dei *pixel* costituenti l'immagine acquisita dalla camera,

$$\mathcal{X} \subset \mathbb{N}^2,$$

e con $x \in \mathcal{X}$ il singolo pixel. Quando vorremo considerare il frame acquisito all'istante di tempo $i$, useremo $z_i$, con $i = 1, \ldots, \infty$. In particolare, per indicare il valore della *luminosità* del pixel $x$ per il frame $i$-esimo, useremo il termine $z_i(x)$, con

$$z_i(x) \in [0, 255].$$

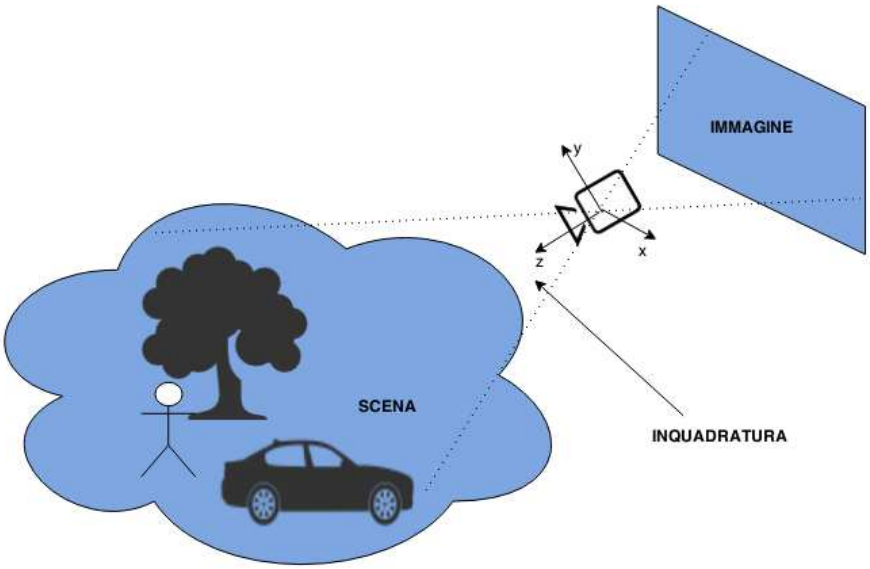



*Figura 2.1: Sistema di monitoraggio video*

## 2.3 Tampering Detection

In un'applicazione di monitoraggio video consideriamo che la camera, in condizioni di funzionamento ottimale, mantenga la stessa inquadratura nel tempo e che sia in grado di acquisire, in maniera nitida, tutti gli elementi di interesse presenti nella scena. Definiamo tampering un qualsiasi evento che determina un cambio di inquadratura o che non permette la corretta acquisizione di parte o totalità della scena. Possiamo classificare gli eventi di tampering in quattro categorie:

- sfocature,
- spostamenti della camera,
- occlusioni dell'obiettivo,
- guasti della camera.

Nei moderni sistemi di videosorveglianza troviamo spesso algoritmi utilizzati per identificare particolari eventi all'interno della scena ripresa dalla camera. Ad esempio è possibile avere un software in grado di identificare le targhe delle automobili che superano il limite di velocità, oppure la presenza di oggetti incustoditi in una stazione [1]. Affinché questi algoritmi funzionino correttamente, è importante che l'*affidabilità* del sistema di acquisizione sia preservata. Per fare questo la letteratura scientifica offre molte tecniche che permettano l'identificazione automatica di eventi in grado di compromettere

la corretta ripresa della scena da parte della videocamera. Possiamo dividere le principali tecniche di tampering detection in due categorie:

- tecniche basate su confronto di background,
- tecniche basate su monitoraggio sequenziale.

### 2.3.1 Tecniche basate su confronto di background

Nella maggior parte dei lavori dedicati a questo problema, il concetto principale consiste nel confrontare ciascun frame con un modello che viene calcolato utilizzando i frame precedenti. Tale modello è ampiamente utilizzato in vari ambiti di visione artificiale e prende il nome di *modello di background*. Un metodo generale di calcolo del background è presentato in [2]. Indichiamo con $z_i(x)$ il valore, nel pixel $x$, della luminosità nell'$i$-esimo frame. Il valore del modello di background è calcolato in maniera ricorsiva secondo la seguente formula:

$$B_{i+1}(x) = \begin{cases} aB_i(x) + (1-a)z_i(x) & \text{, se } |z_i(x) - z_{i-1}(x)| \leq T_i(x) \\ B_i(x) & \text{, se } |z_i(x) - z_{i-1}(x)| > T_i(x) \end{cases},$$

dove $0 < a < 1$ è chiamato *parametro di aggiornamento* (*update parameter*) e $T_i(x)$ è una soglia che permette di identificare un cambiamento sostanziale di luminosità nel pixel $(x)$. Questa soglia viene aggiornata in maniera ricorsiva secondo la seguente formula:

$$T_{i+1}(x) = \begin{cases} aT_i(x) + (1-a)(c|z_i(x) - B_i(x)|) & \text{, se } |z_i(x) - z_{i-1}(x)| \leq T_i(x) \\ T_i(x) & \text{, se } |z_i(x) - z_{i-1}(x)| > T_i(x) \end{cases},$$

dove $c > 1$ e $0 < a < 1$. Lo stesso modello viene applicato in altri lavori ([17], [19]), mentre una variante molto usata consiste nell'estrarre il modello di background a partire dall'*estrazione dei contorni* da ciascun frame ([10], [9]). Un approccio diverso, ma comunque riconducibile allo stesso filone, lo troviamo in [14], dove il background è sostituito da un *buffer* contenente gli ultimi $n$ frame acquisiti dalla camera, che vengono confrontati con l'ultima osservazione per identificare eventi di tamepering.

#### Identificazione di occlusioni

L'evento di occlusione avviene quando un oggetto opaco viene posto vicino alla camera, in modo da coprire la scena ripresa. In [2] e [17] questo evento è associato a un cambiamento nella struttura dell'*istogramma* del frame occluso rispetto a quello del background. Infatti, nel caso di occlusioni, i valori dell'istogramma tendono a concentrarsi in un intervallo molto ristretto, verso i livelli più bassi della scala di grigi.

Un approccio simile è presente in [10], [9] e [8], in cui l'evento di occlusione è associato a un abbassamento dell'*entropia*:

$$E = -\sum_k p_k \ln(p_k),$$

dove $p_k$ rappresenta la probabilità che il livello di grigio $k$ sia presente all'interno dell'immagine.

**Identificazione di spostamenti della camera**

Quando viene spostata la camera, in modo cambi l'inquadratura della scena, l'immagine di background $B_i$ viene lentamente aggiornato in modo da riflettere i cambiamenti avvenuti nei frame. In [17] il metodo proposto per identificare uno spostamento della camera consiste nel confrontare l'immagine di background $B_i$ con $B_{i-k}$, ovvero con un secondo modello *ritardato* di $k$ frame, dove $k \in \mathbb{Z}^+$. L'approccio consiste nel calcolare un *valore di proporzione* $P$, ottenuto dal confronto tra i due modelli:

$$P = \begin{cases} P+1 & \text{, se } B_i(x) \neq B_{n-k}(x) \\ P & \text{, se } B_i(x) = B_{n-k}(x) \end{cases}.$$

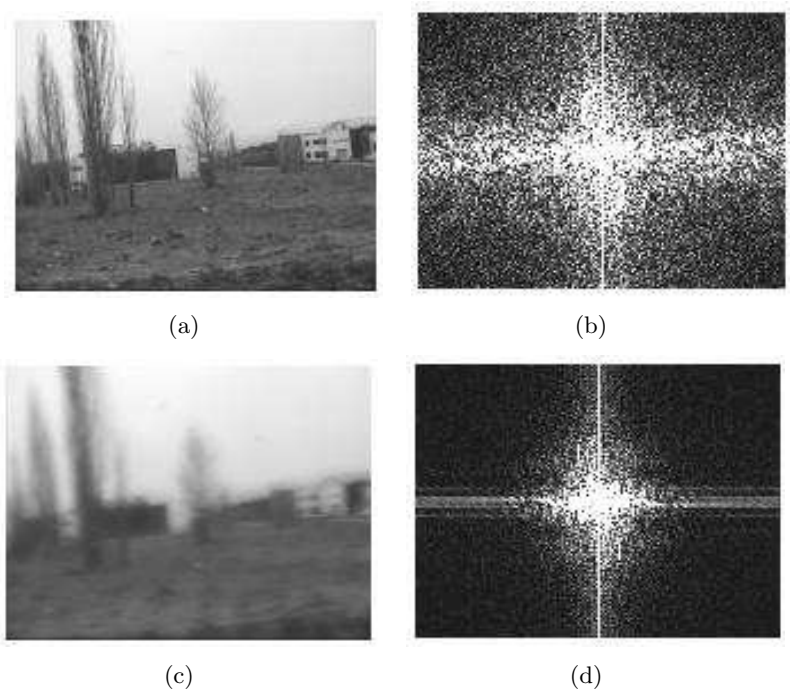
Lo spostamento della camera viene identificato quando $P > ThK$, dove $0 < Th < 1$ è un valore di soglia scelto in base alla sensibilità che si vuole dare all'algoritmo e $K$ è il numero totale di pixel dell'immagine.
Un altro approccio è quello di utilizzare tecniche di *block matching*. In [9] e [10] il confronto viene fatto utilizzando la formula *zero-mean normalized cross correlation* (ZNCC) [15]:

$$ZNCC_i(m) = \frac{\sum_{x \in \mathcal{X}} (B_{i-1}(x) - \mu_{B_{i-1}})(z_i(x+m) - \mu_{z_i})}{\sigma_{B_{i-1}} \sigma_{z_i}},$$

dove $\mu_{z_i}$ e $\sigma_{z_i}$ rappresentano la media e la deviazione standard della luminosità nell'immagine $z_i$. Un'altra cifra di merito, utilizzata in [8], è la *minimum squared difference* (MSD):

$$MSD = \frac{1}{K} \sum_{x \in \mathcal{X}} (z_i - B_i)^2$$

L'algoritmo di block matching fornisce due parametri: il primo parametro, $m$, indica il *vettore della traslazione* tra il background e il frame corrente, mentre il secondo parametro è lo ZNCC relativo a quel vettore. Lo spostamento viene individuato quando il vettore $m$ ha una lunghezza minima e lo ZNCC corrispondente supera una certa soglia. Un metodo simile viene utilizzato anche in [11], con la differenza che, invece di analizzare la correlazione dei pixel, viene analizzata quella degli istogrammi.



(a) (b)

(c) (d)

*Figura 2.2: Comportamento della trasformata di Fourier nel caso di sfocatura*

### Identificazione di sfocature

La conseguenza di un evento di sfocatura è la perdita di dettagli nell'immagine. In [2] e [17] questo fenomeno è associato a una diminuzione dell'energia ad alta frequenza. Per analizzare questo cambiamento, [2] confronta ciascun frame con il background nel dominio delle *wavelet* [12], mentre [17] utilizza il dominio della *trasformata di Fourier* [6]. Nella figura 2.2 vediamo un esempio di come si comporta la trasformata di Fourier nel caso di sfocature: nel passaggio da un frame nitido (Fig. 2.2(a)) a uno sfocato (Fig. 2.2(c)) abbiamo un crollo delle componenti ad alta frequenza nelle trasformate di Fourier (rispettivamente Fig. 2.2(b) e Fig. 2.2(d)). L'evento di tampering viene individuato quando l'*energia media* della trasformata di Fourier (o di quella wavelet) del frame corrente $E_{HF}(z_i)$ è $Th$ volte minore di quella del background $E_{HF}(B_i)$:

$$E_{HF}(z_i) \leq ThE_{HF}(B_i),$$

dove $0 < Th < 1$ è un valore di soglia scelto in base alla sensibilità che si vuole dare all'algoritmo.
Un altro approccio consiste nell'analizzare la perdita di dettagli confrontando i contorni (*edges*) del frame corrente con quelli del background. Questo metodo, utilizzato in [8], [9], [10] e [11], consiste nell'estrarre i contorni dalle

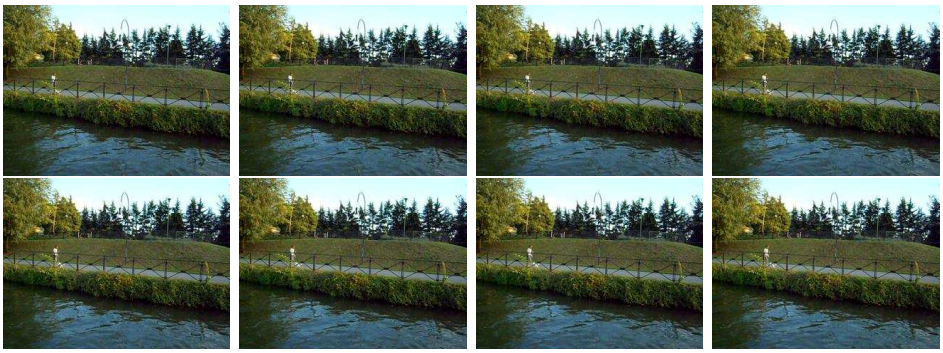


*Figura 2.3: Sequenza di frame acquisiti a 30 fps*

immagini secondo il metodo di Sobel [18] o di Canny [7], e confrontare il numero di pixel dei contorni con quelli del background. Quando il numero di pixel dei contorni del frame corrente $\sum edges(z_i)$ è $Th$ volte più piccolo di quello del background $\sum edges(B_i)$:

$$\sum edges(z_i) \leq Th \sum edges(B_i),$$

dove $0 < Th < 1$ è un valore di soglia scelto in base alla sensibilità che si vuole dare all'algoritmo.

### 2.3.2 Tecniche basate su monitoraggio sequenziale

Le tecniche viste finora, come è stato detto nel paragrafo precedente, permettono di aggiornare ciascun frame con un modello della scena che viene calcolato in base alle osservazioni precedenti. Questo approccio risulta fattibile nel caso in cui la camera operi con un framerate *continuo*, solitamente tra i 30 frame per secondo (*fps*) e i 2 fps. In questi casi, infatti, possiamo considerare che, tra un frame e il successivo, non avvengano grandi cambiamenti all'interno della scena e non cambi eccessivamente la luminosità media. La figura 2.3 mostra come, in un'acquisizione fatta a 30 fps, le differenze tra frame consecutivi siano quasi inesistenti. Un evento di tampering può, quindi, essere identificato in maniera molto semplice, in quanto una differenza molto elevata tra il frame analizzato e il background può essere dovuto solamente a un'azione sulla camera. La figura 2.4 mostra, invece, un esempio di frame acquisiti ogni 30 secondi. Notiamo come le differenze tra immagini consecutive, in questo caso, siano più marcate rispetto al caso dell'acquisizione continua. Questo fa sì che l'utilizzo di un modello di background risulti inefficiente. Evidenziamo, inoltre, come le tecniche viste finora richiedano capacità computazionale da parte del sistema di monitoraggio.
Un approccio alternativo consiste nel monitorare nel tempo il comportamento di alcuni indicatori estratti dalle singole immagini acquisite. Si presup-

*Figura 2.4: Sequenza di frame acquisiti ogni 30 secondi*

pone che, quando il sistema di monitoraggio opera in condizioni di funzionamento *ottimali*, gli indicatori monitorati presentino una certa *stazionarietà*, ovvero siano considerati dei campioni *indipendenti* tra loro e distribuiti secondo una stessa funzione di ripartizione. L'evento di tampering viene considerato come un *cambiamento nella stazionarietà* di questi indicatori.
Il monitoraggio può avvenire utilizzando tecniche statistiche, come ad esempio *change-point method* (CPM) [16] o *change-detection test* [13]. Troviamo alcuni esempi nell'identificazione di sfocature: in [19] la soluzione consiste nel monitorare nel tempo il *numero di SURF* [5], in quanto tali descrittori decrementano in maniera considerevole il loro numero in presenza di sfocature. Notiamo, però, che l'utilizzo di una tecnica del genere richiede un elevato numero di calcoli per ricavare le SURF. Il metodo, quindi, si presta poco a essere utilizzato su sistemi di monitoraggio a basso consumo. Un altro esempio è dato da [3], dove le sfocature vengono identificate monitorando l'*energia media del gradiente* delle immagini acquisite:

$$m_i = \mathcal{M}[z_i] = \int_{\mathcal{X}} \| \bigtriangledown z_i(x) \|_1 dx, \tag{2.1}$$

dove $\|\cdot\|_1$ si riferisce alla *norma* $\mathcal{L}^1$. Per identificare il cambio di stazionarietà di questo indicatore vengono utilizzate tecniche di CDT basate su *somme cumulate* (CUSUM) [4], utilizzando un sottoinsieme delle prime osservazioni come sequenza di *training*, utilizzata per configurare il test.

# Capitolo 3

# Impostazione del problema di ricerca

In questo capitolo descriviamo il problema, affrontato dall'algoritmo di tampering detection, in maniera formale e rigorosa. Il primo paragrafo illustra il modello delle osservazioni e gli eventi che siamo interessati a identificare, mentre il secondo paragrafo formalizza il concetto di tampering detection.

## 3.1 Modello delle osservazioni

Il nostro campo di osservazione si concentra su quegli eventi che si interpongono tra la scena ripresa da una camera e il sensore che acquisisce le immagini. Non vogliamo, cioè, identificare degli eventi particolari che avvengono nella scena, come un oggetto lasciato incustodito [1], bensì vogliamo identificare quegli eventi tali per cui il sensore non è più nelle condizioni di riprendere, in maniera ottimale, la scena, quali ad esempio sfocature o spostamenti della camera. Nel seguito cerchiamo di dare una definizione formale di questi eventi.

### 3.1.1 Scena e posizione della camera

Prima di tutto definiamo, in maniera rigorosa, i termini che verranno utilizzati nel seguito della trattazione.
Lo scenario che consideriamo è quello di una camera che deve riprendere una particolare *scena*. La posizione e la rotazione della camera determinano l'*inquadratura* della scena. In un'applicazione di monitoraggio video consideriamo che la camera, in condizioni di funzionamento ottimale, mantenga la stessa inquadratura nel tempo e che si in grado di acquisire, in maniera nitida, tutti gli elementi di interesse presenti nella scena. Definiamo tampering un qualsiasi evento che determina un cambio di inquadratura o che non


(a) (b)

*Figura 3.1: Esempi di sfocature*

permette la corretta acquisizione di parte o totalità della scena. Possiamo classificare gli eventi di tampering in quattro categorie:

- Sfocature
- Spostamenti della camera
- Occlusioni
- Guasti della camera

### 3.1.2 Sfocatura

Il fenomeno della sfocatura avviene quando un elemento trasparente o semitrasparente si interpone tra la lente della camera e la scena ripresa, oppure quando viene cambiata la messa a fuoco, causando una perdita nei dettagli della scena ripresa.
Nella figura 3.1 sono mostrati degli esempi in cui sono presenti delle sfocature. Queste possono essere di origine diversa:

- dovute a *cause naturali*, come ad esempio dell'acqua piovana che si deposita sulla lente (figura 3.1(a)), o la condensa dovuta all'umidità e alle basse temperature, oppure un raggio di sole incidente sull'obiettivo della camera;
- per *intervento dell'uomo*, che a sua volta può avvenire in maniera intenzionale (e in questo caso si può parlare di *manomissione*) oppure non intenzionale. Ad esempio, si può direttamente intervenire sulla messa a fuoco, nel caso sia possibile cambiarla manualmente; oppure (come nel caso della figura 3.1(b)) è possibile applicare una sostanza semitrasparente sulla lente della camera, come il gas di un deodorante spray.

Inoltre, possiamo notare come nella figura 3.1(b) la sfocatura sia applicata su tutta l'immagine, mentre nella figura 3.1(a) la sfocatura si concentri solo in alcune aree (quelle dove sono presenti le gocce). Nel primo caso si parlerà, quindi, di sfocatura *totale*, mentre nel secondo caso di sfocatura *parziale*.
Riprendendo [3], questo fenomeno può essere modellato come un operatore di *degradazione* $\mathcal{D}$ applicato a un'immagine $y$, considerata priva di errori, i.e.,

$$z = \mathcal{D}[y]. \tag{3.1}$$

In particolare, all'interno dell'operatore $\mathcal{D}$ si può considerare il contributo dovuto a un operatore di *sfocatura* $\mathcal{B}$ (dall'inglese *blur*) e un termine aleatorio $\eta$ corrispondente al rumore, i.e.,

$$z(x) = \mathcal{D}[y](x) = \mathcal{B}[y](x) + \eta(x), \qquad x \in \mathcal{X} \tag{3.2}$$

dove abbiamo indicato con $x$ le coordinate dei *pixel* dell'immagine e $\mathcal{X}$ è l'insieme dei pixel che formano l'immagine. Considerando il caso continuo, possiamo assumere la sfocatura $\mathcal{B}$ come un operatore *lineare*,

$$\mathcal{B}[y](x) = \int_{\mathcal{X}} y(s)h(x,s)ds, \tag{3.3}$$

dove $h(x,s)$ rappresenta la *risposta all'impulso* (*point spread function* (PSF)) della sfocatura sul pixel $x$, il cui risultato consiste nel rendere le differenze di intensità, tra pixel adiacenti, più morbide (*smooth*). Nel caso in cui la sfocatura sia applicata sulla totalità dell'immagine (come nel caso della figura 3.1(b)), allora è possibile modellare l'operatore di blur come una *convoluzione*[1]:

$$\mathcal{B}[y](x) = \int_{\mathcal{X}} y(s)h(s-x)ds, \tag{3.4}$$

dove $h(.)$ è un filtro gaussiano o uniforme.
Nel caso più generale possiamo considerare che la camera acquisisca un sequenza di $N$ osservazioni $\{z_i\}, i = 1, \ldots, N$, quindi la formula 3.2 si può riscrivere come

$$z_i(x) = \mathcal{D}_i[y_i](x) = \mathcal{B}_i[y_i](x) + \eta(x), \qquad x \in \mathcal{X}. \tag{3.5}$$

La sequenza delle immagini $\{y_i\}, i = 1, \ldots, N$, può variare in maniera significativa nel suo contenuto, anche nel caso in cui la vista sia la stessa. Un esempio è illustrato nella figura 3.2, in cui le immagini riprese dalla camera sono acquisite ogni minuto. Nonostante l'inquadratura non cambi tra le acquisizioni, il contenuto delle singole immagini varia parecchio, dato il continuo passaggio di automobili e pedoni. Questo problema fa sì che l'identificazione delle sfocature non possa avvenire facendo un semplice confronto

[1] Il blur convoluzionale è quello che abbiamo utilizzato per generare, in maniera sintetica, sequenze con immagini sfocate.

*Figura 3.2: Sequenza di otto frame consecutivi acquisiti ogni minuto*

(a) (b)

*Figura 3.3: Esempio di spostamento della camera*

tra frame consecutivi, in quanto avremmo un numero troppo elevato di falsi positivi. Infatti, nel caso in cui avessimo un riscontro negativo dal confronto tra due frame ($z_i \neq z_{i+1}$), sarebbe difficile capire se è cambiato il contenuto delle immagini ($y_i \neq y_{i+1}$) o l'operatore di sfocatura ($\mathcal{B}_i \neq \mathcal{B}_{i+1}$).

### 3.1.3 Spostamento della camera

Lo spostamento della camera avviene quando cambia la sua vista. Le cause possono essere, ancora una volta, di tipo naturale, ad esempio una raffica di vento che sposta la camera, oppure dovute a un intervento malevolo da parte di un uomo. Un esempio è mostrato nella figura 3.3, dove vediamo che tra il frame riportato in figura 3.3(a) e quello in figura 3.3(b) c'è stato un evento che ha spostato la camera. Possiamo formalizzare il concetto di spostamento della camera nel modo seguente: consideriamo la sequenza $\{y_i\}$ di immagini generate da una camera in una certa posizione, e la sequenza $\{w_i\}$ di immagini generate dalla stessa camera in una posizione differente. Possiamo, dunque, considerare la sequenza di immagini $\{z_i\}$ in cui avviene

(a) Occlusione volontaria (b) Occlusione dovuta a deposito di neve

*Figura 3.4: Esempi di occlusione*

uno spostamento della camera all'istante $T^*$ nel seguente modo:

$$z_i(x) = \mathcal{S}_i[y_i](x) = \begin{cases} y_i(x) + \eta(x) & \text{per} \quad i < T^* \\ w_i(x) + \eta(x) & \text{per} \quad i \geqslant T^* \end{cases}, \tag{3.6}$$

dove $\eta(x)$ è un rumore stazionario.
In questa formulazione abbiamo considerato lo spostamento come un fenomeno *istantaneo*; in generale, possiamo considerare una fase *transitoria* in cui l'inquadratura della camera cambia a ogni frame acquisito, fino a raggiungere la posizione finale. Dato che, nella nostra applicazione, la camera opera con framerate bassi (come ad esempio un'immagine al minuto), possiamo considerare lo spostamento come istantaneo e, quindi, tenere come riferimento il modello della formula 3.6.
Anche per lo spostamento della camera vale la considerazione fatta nel caso della sfocatura: il contenuto delle immagini varia con il passare del tempo, quindi identificare lo spostamento confrontando frame consecutivi nel tempo genera un alto numero di falsi positivi.

### 3.1.4 Occlusione e guasti della camera

Il fenomeno dell'occlusione avviene quando un oggetto opaco si pone a ridosso della camera, impedendo la visione di una parte, se non la totalità, della scena. L'effetto di questo evento è quello di impedire l'acquisizione della scena nella sua totalità. Un esempio di occlusione ce l'abbiamo quando qualcuno copre la camera con un cappello, o mette un foglio di giornale davanti alla lente del sensore, oppure può succedere che della neve si accumuli sull'ottica del sensore. Il guasto del sensore invece comporta un aumento del rumore presente nell'immagine.
Queste problemi non sono stati affrontati durante la tesi, comunque le tecniche messe a punto per rilevare spostamenti e sfocature possono essere utilizzate anche per rilevare questo tipo di eventi.

## 3.2 Tampering detection

L'algoritmo di tampering detection consiste nell'analizzare la sequenza di osservazioni $\{z_i\}, i = 1, \ldots, N$ in modo da determinare un possibile cambiamento nell'operatore di degradazione $\mathcal{D}$ o in quello di spostamento $\mathcal{S}$. Come abbiamo illustrato precedentemente, discriminare tra cambiamenti avvenuti a livello di contenuto della scena e cambiamenti dovuti a tampering può diventare complicato confrontando direttamente i frame tra loro.

# Capitolo 4

# Soluzione proposta


**4.1 Estrazione dei descrittori del cambiamento**

**4.2 Algoritmo di segmentazione**

**4.3 Monitoraggio one-shot**

**4.4 Monitoraggio sequenziale**

# Capitolo 5

# Realizzazioni sperimentali e valutazione

## 5.1 Acquisizione dei dataset

## 5.2 Risultati

Capitolo 6

# Direzioni future di ricerca e conclusioni

# Bibliografia


[1] `http://www.mitan.it/security-solution/videosorveglianza/sistemi-di-videosorveglianza-e-registrazione/`. Visitato il giorno 09/03/2015.

[2] 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.

[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] Cesare Alippi and Manuel Roveri. Just-in-time adaptive classifiers-part I: Detecting nonstationary changes. *Neural Networks, IEEE Transactions on*, 19(7):1145–1153, 2008.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[6] Ronald N Bracewell. The fourier transform and its applications. 1978.

[7] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[8] Damian Ellwart, Piotr Szczuko, and Andrzej Czyżewski. Camera sabotage detection for surveillance systems. In *Security and Intelligent Information Systems*, pages 45–53. Springer, 2012.

[9] 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.

[10] 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.

[11] Tomasz Kryjak, Mateusz Komorkiewicz, and Marek Gorgon. Fpga implementation of real-time head-shoulder detection using local binary patterns, svm and foreground object detection. In *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*, pages 1–8. IEEE, 2012.

[12] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):674–693, 1989.

[13] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[14] 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.

[15] Nuno Roma, José Santos-Victor, and José Tomé. A comparative analysis of cross-correlation matching algorithms using a pyramidal resolution approach. 2002.

[16] Gordon J. Ross, Dimitris K. Tasoulis, and Niall M. Adams. Nonparametric monitoring of data streams for changes in location and scale. *Technometrics*, 53(4):379–389, 2011.

[17] 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.

[18] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. 1968.

[19] 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.