

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



**SCENE SEGMENTATION FOR
TAMPERING DETECTION IN
EMBEDDED SYSTEMS**

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

Tesi di Laurea di:
Adriano GAIBOTTI, matricola 780200

Anno Accademico 2013-2014

A Gigino

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
1.1 Aumento nell'interesse dei contenuti multimediali, tra cui il <i>monitoraggio video</i> (IoT, domotica,...)	3
1.2 Esempi videosorveglianza e webcam meteo	3
1.3 Problema dell'affidabilità del contenuto video	3
1.4 Tampering Detection	4
1.5 Riassunto soa	4
1.6 Scopo della tesi	4
1.7 Soluzione proposta	4
1.8 Esperimenti fatti	4
1.9 Problemi rimasti aperti	4
1.10 Struttura della tesi	4
2 Stato dell'arte	5
2.1 Modello della camera	5
2.1.1 La matrice prospettica della camera	6
2.1.2 Parametri intrinseci	7
2.1.3 Parametri estrinseci	10
2.2 Rappresentazione digitale delle immagini	11
2.3 Tampering Detection	12
2.3.1 Concetti e terminologia	14
2.3.2 Tecniche basate su confronto di background	15
2.3.3 Tecniche basate su monitoraggio sequenziale	19
3 Impostazione del problema di ricerca	23
3.1 Modello delle osservazioni	23
3.1.1 Sfocatura	23
3.1.2 Spostamento della camera	25
3.1.3 Occlusione e guasti della camera	27
3.2 Tampering detection	27

4 Soluzione proposta	29
4.1 Indicatori utilizzati per identificare gli eventi di tampering	29
4.1.1 Misura della sfocatura nell'immagine	29
4.1.2 Misura del displacement	30
4.1.3 Comportamento degli indicatori nel tempo	31
4.2 Algoritmo per il monitoraggio degli indicatori	39
4.2.1 Monitoraggio one-shot	39
4.2.2 Monitoraggio sequenziale	39
4.3 Algoritmo di segmentazione	39
5 Realizzazioni sperimentali e valutazione	41
5.1 Acquisizione dei dataset	41
5.1.1 Sistemi di acquisizione utilizzati	42
5.1.2 Sequenze con presenza di tampering	43
5.1.3 Definizione dei ground thruth	43
5.1.4 Definizione delle metriche per la stima delle prestazioni	43
5.2 Risultati ottenuti	43
6 Direzioni future di ricerca e conclusioni	45
Bibliografia	47

Elenco delle figure

2.1	Athanasius Kircher, Principio della prospettiva nella camera obscura, 1671	6
2.2	Schematizzazione del sistema ottico della camera	7
2.3	Similitudini tra triangoli usate nel calcolo dei parametri intrinseci	8
2.4	Coordinate di un frame	9
2.5	Trasformazione dal sistema di coordinate dell'ambiente a quello della camera	10
2.6	Esempio di estrazione dei canali RGB	11
2.7	Esempio di estrazione dei canali YUV	12
2.8	Tecniche di tampering detection	13
2.9	Sistema di monitoraggio video	14
2.10	Comportamento della trasformata di Fourier nel caso di sfocatura	17
2.11	Sequenza di frame acquisiti a 30 fps	18
2.12	Sequenza di frame acquisiti ogni 30 secondi	19
3.1	Esempi di sfocature	24
3.2	Sequenza di otto frame consecutivi acquisiti ogni minuto	25
3.3	Esempio di spostamento della camera	26
3.4	Esempi di occlusione	27
3.5	Esempi di guasti	28
4.1	Esempio di cambi di luminosità tra il giorno e la notte	31
4.2	Esempio di presenza di sfocature dovute all'aumento del tempo di esposizione della camera	31
4.3	Energia media del gradiente lungo un'acquisizione di 24 ore (a) e suo detrending (b)	32
4.4	Energia media della luma lungo un'acquisizione di 24 ore (a) e suo detrending (b)	33
4.5	Energia media del gradiente (a) e della luma (b) nel caso di una sfocatura	34
4.6	Detrending dell'energia media del gradiente (a) e della luma (b) nel caso di una sfocatura	35

4.7	Energia media del gradiente (a) e della luma (b) nel caso di uno spostamento della camera	36
4.8	Detrending dell'energia media del gradiente (a) e della luma (b) nel caso di uno spostamento della camera	37
4.9	Esempio di sequenza dell'energia media della luma (a) e del suo detrending (b) con un displacement difficile da identificare	38
4.10	Esempio di spostamento della camera	39
5.1	Il Raspberry Pi utilizzato per l'acquisizione	42

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

Punti da sviluppare:

1.1 Aumento nell'interesse dei contenuti multimediali, tra cui il *monitoraggio video* (IoT, domotica,...)

1.2 Esempi videosorveglianza e webcam meteo

1.3 Problema dell'affidabilità del contenuto video

- Vitale nel caso della videosorveglianza
- Caso webcam limitare traffico in rete evitando di inviare frame compromessi
- Qualità del servizio

1.4 Tampering Detection

1.5 Riassunto soa

Enfatizzare il fatto che il lavoro presente in letteratura è legato solamente ad applicazioni di videosorveglianza

1.6 Scopo della tesi

1.7 Soluzione proposta

1.8 Esperimenti fatti

1.9 Problemi rimasti aperti

1.10 Struttura della tesi

La tesi è strutturata nel seguente modo.

Nel capitolo 2 si mostra lo stato dell'arte.

Nel capitolo 3 si illustra come è stato formalizzato il problema.

Nel capitolo 4 si illustra la soluzione proposta per risolvere il problema.

Nel capitolo 5 si mostrano le prove realizzate per validare la soluzione proposta, descrivendo anche la realizzazione dei dataset e i risultati ottenuti.

Nel capitolo 6 si mostrano le prospettive future di ricerca e si tirano le conclusioni.

Capitolo 2

Stato dell'arte

In questo capitolo presentiamo i concetti fondamentali della teoria alla base della *visione artificiale* e lo studio fatto finora sul problema del *tampering detection*. Nei paragrafi 2.1 e 2.2 diamo una visione generale del modello della camera e di come vengono rappresentate le immagini digitali, in modo da comprendere meglio gli argomenti trattati nei capitoli successivi. Nel paragrafo 2.3 descriviamo le soluzioni al problema del tampering detection presenti nella letteratura scientifica, definendo inoltre i concetti e la terminologia che verrà utilizzata nel resto della trattazione.

2.1 Modello della camera

Affinché un sistema di visione artificiale possa risolvere il problema per cui è stato progettato, è necessario che esso sia in grado di acquisire una porzione della realtà che lo circonda. Questo compito viene svolto da un particolare sensore chiamato *camera*, il cui scopo principale è quello di creare una *proiezione* dell'ambiente *tridimensionale* in un sistema *bidimensionale*. Il principio alla base della camera è un concetto inventato da *Filippo Brunelleschi* nel XV secolo, chiamato *prospettiva a punto unico di fuga* (*pinhole perspective*) [1]. Il modello matematico utilizzato considera uno spazio tridimensionale, detto *ambiente*, e un piano bidimensionale chiamato *immagine*. I punti su tale piano sono la *proiezione* dell'ambiente tridimensionale. La proiezione è dovuta a un punto, chiamato *centro ottico*, in cui viene convogliata la luce emanata dall'ambiente tridimensionale. In questa astrazione, ciascun punto dello spazio tridimensionale viene associato univocamente a un punto nel piano immagine attraverso il raggio di luce che passa dal centro ottico. All'epoca di Brunelleschi, il modello del centro ottico veniva realizzato da una *camera oscura*, come si può osservare nell'illustrazione in figura 2.1, mentre nei moderni sistemi di acquisizione viene realizzato tramite *lenti ottiche*.

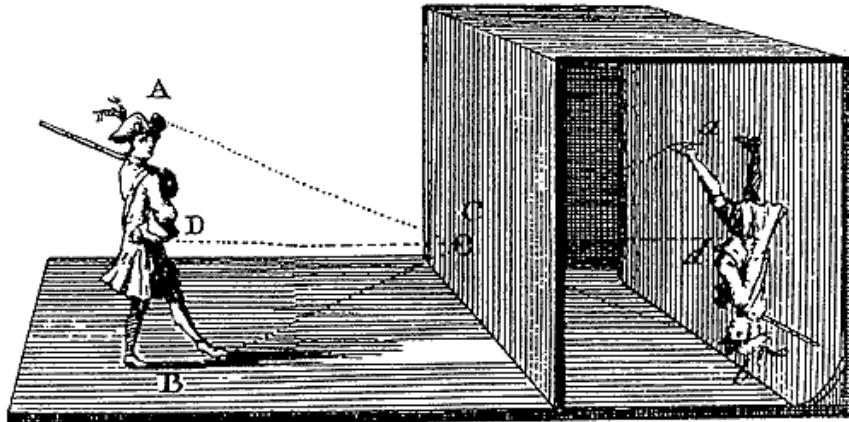


Figura 2.1: Athanasius Kircher, *Principio della prospettiva nella camera obscura*, 1671

2.1.1 La matrice prospettica della camera

Nella figura 2.2 sono illustrati i concetti che sono utilizzati per descrivere il sistema ottico della camera. Indichiamo con la lettera maiuscola $\mathbf{M} = [X, Y, Z]^T$ un qualsiasi punto nello spazio tridimensionale e con la lettera minuscola $\mathbf{m} = [u, v]^T$ la sua proiezione sul piano immagine dovuta al centro ottico \mathbf{C} . La linea che congiunge il centro ottico \mathbf{C} perpendicolarmente al piano immagine è detta *asse principale*, indicata con Z , e il suo punto di intersezione con il piano stesso viene definita *punto principale c*. La distanza tra il punto principale e il centro ottico viene definita *distanza focale*, e viene indicata con f .

La rappresentazione dei punti viene fatto attraverso le loro *coordinate omogenee*. In questo modo è possibile rappresentare le trasformazioni tra sistemi di coordinate di ordine differente tramite una singola trasformazione matriciale. Chiamiamo coordinate omogenee di un punto $\mathbf{m} = [u, v]$ del piano una qualsiasi terna ordinata $[U, V, W]$ di numeri reali tali che

$$W \neq 0, \frac{U}{W} = u, \frac{V}{W} = v.$$

Analogamente, le coordinate omogenee di un punto $\mathbf{M} = [x, y, z]$ nello spazio tridimensionale saranno costituite da una quaterna di numeri $[X, Y, Z, W]$ tali che

$$W \neq 0, \frac{X}{W} = x, \frac{Y}{W} = y, \frac{Z}{W} = z.$$

La coordinata W viene definita *valore di scala*; nel caso $W = 1$ le rimanenti coordinate omogenee rappresentano le *coordinate cartesiane* del punto.

Consideriamo un punto \mathbf{M} nello spazio tridimensionale, rappresentato tramite le coordinate omogenee $[X, Y, Z, 1]^T$, e il suo punto immagine \mathbf{m} rappresentato dalle coordinate omogenee $[u, v, 1]^T$. La proiezione della camera

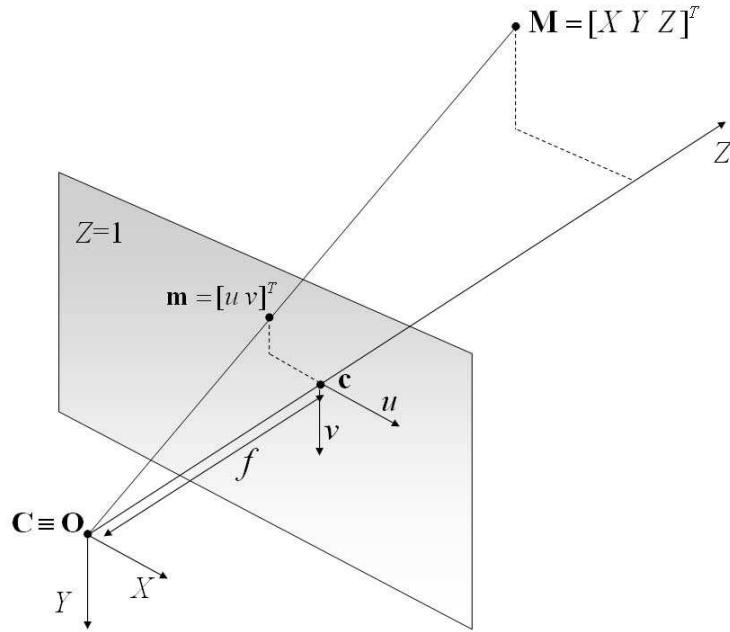


Figura 2.2: Schematizzazione del sistema ottico della camera

può essere espressa come

$$\mathbf{m} = P\mathbf{M}, \quad (2.1)$$

dove P è una matrice di dimensioni 3×4 chiamata *matrice prospettica* [2]. Tale matrice contiene al suo interno informazioni sui parametri della camera a cui è associata, e descrive sia la proiezione sul piano immagine che le trasformazioni di camera rispetto al sistema di riferimento tridimensionale. Formalmente la matrice prospettica viene definita attraverso la concatenazione di due matrici: una che rappresenta i *parametri intrinseci* e un'altra che rappresenta i *parametri estrinseci* [3].

- I *parametri intrinseci* rappresentano le caratteristiche interne della camera come la distanza focale, il centro focale e le caratteristiche di *distorsione* della lente.
- I *parametri estrinseci* rappresentano la posizione della camera rispetto al sistema di riferimento dello spazio tridimensionale.

2.1.2 Parametri intrinseci

La relazione fra le coordinate tridimensionali di un punto $\mathbf{M} = [X, Y, Z]$ e le coordinate della sua proiezione $\mathbf{m} = [u, v]$ è:

$$u = f \cdot \frac{X}{Z},$$

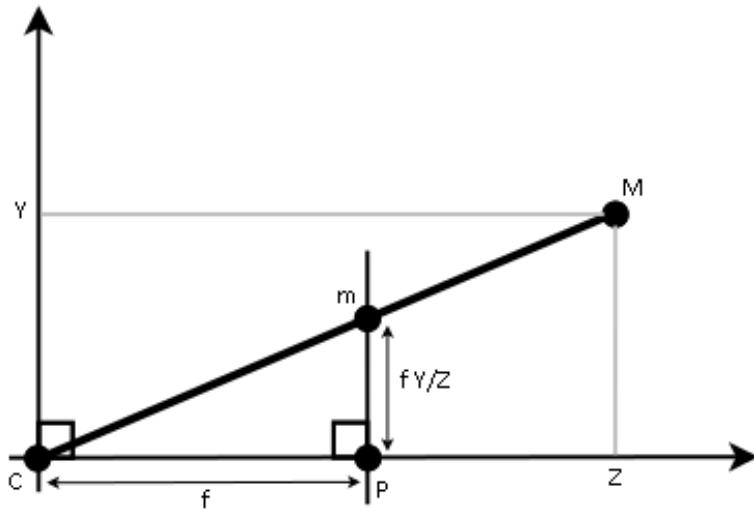


Figura 2.3: Similitudini tra triangoli usate nel calcolo dei parametri intrinseci

$$v = f \cdot \frac{Y}{Z}.$$

Tale relazione deriva dalle proprietà dei triangoli simili, come mostra la figura 2.3. La *mappatura* viene definita, dunque, dalla seguente relazione:

$$[X, Y, Z]^T \mapsto \left[f \cdot \frac{X}{Z}, f \cdot \frac{Y}{Z} \right]^T. \quad (2.2)$$

Se il punto nello spazio e la sua proiezione sono rappresentati attraverso le loro coordinate omogenee, la (2.2) può essere riscritta come

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} f \cdot X \\ f \cdot Y \\ Z \\ 1 \end{bmatrix} = K \begin{bmatrix} I & | & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.3)$$

dove la matrice

$$K = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \quad (2.4)$$

viene detta *matrice di calibrazione* della camera.

Nella formula (2.2) abbiamo assunto che l'origine delle coordinate del piano immagine si trovi nel punto principale, come è illustrato nella figura 2.2. Se consideriamo il punto principale di coordinate $[p_u, p_v]$, la (2.2) diventa

$$[X, Y, Z]^T \mapsto \left[f \cdot \frac{X}{Z} + p_u, f \cdot \frac{Y}{Z} + p_v \right]^T, \quad (2.5)$$

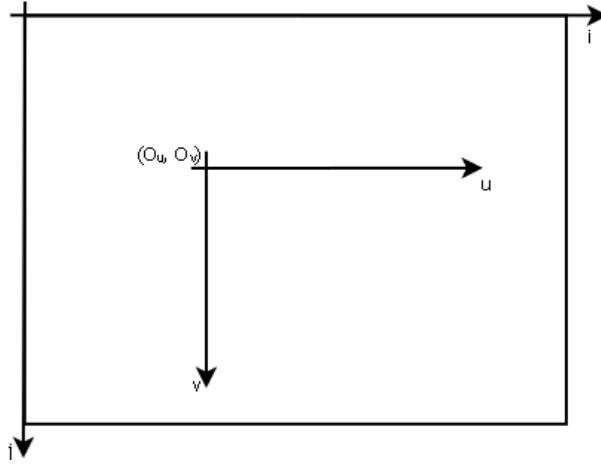


Figura 2.4: Coordinate di un frame

mentre la matrice di calibrazione(2.4) diventa

$$K = \begin{bmatrix} f & p_u \\ f & p_v \\ 1 \end{bmatrix}. \quad (2.6)$$

Nel caso in cui il sensore della camera sia *digitale*, l'immagine deve essere quantizzata come una matrice di W colonne e H righe, in cui ciascun elemento prende il nome di *pixel* (dalla contrazione di *picture element*). Ogni pixel possiede, quindi, delle coordinate che definiscono la sua posizione sulla matrice del frame. Se definiamo con (i, j) le coordinate del generico pixel sulla matrice del frame, dove l'origine è posta nell'angolo in alto a sinistra, e con (O_u, O_v) le coordinate del punto principale secondo il sistema di riferimento del frame (si veda a riguardo la figura 2.4), possiamo mettere in relazione le coordinate dell'immagine con le coordinate frame nel seguente modo:

$$u = (i - O_u) \cdot S_u$$

e

$$v = (j - O_v) \cdot S_v,$$

dove S_u e S_v sono le dimensioni orizzontali e verticali del singolo pixel. È necessario introdurre, nella matrice di calibrazione, l'informazione del numero di pixel per unità di distanza in coordinate immagine m_u e m_v , rispettivamente nelle direzioni u e v , modificando la matrice di calibrazione definita in (2.6) come

$$K = \begin{bmatrix} \alpha_u & u_0 \\ \alpha_v & v_0 \\ 1 \end{bmatrix}, \quad (2.7)$$

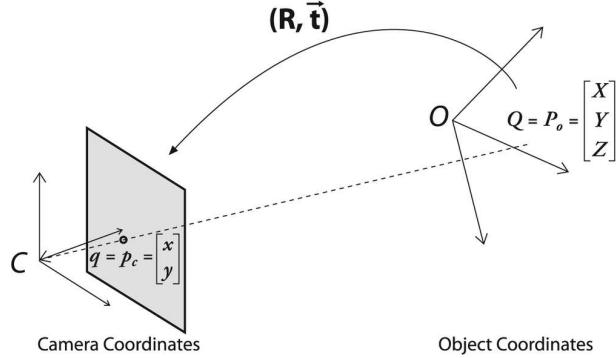


Figura 2.5: Trasformazione dal sistema di coordinate dell'ambiente a quello della camera

dove $\alpha_u = fm_u$ e $\alpha_v = fm_v$, mentre il punto principale $[p_u, p_v]$ viene riscritto come $[u_0, v_0] = [m_u p_u, m_v p_v]$.

Nel caso in cui la matrice di calibrazione sia nota, la camera viene detta *completamente calibrata*.

La matrice di calibrazione K rappresenta, quindi, il cambio di sistema di riferimento nel piano immagine, ridefinendo il centro della camera (u_0, v_0) in modo che coincida con il punto principale.

2.1.3 Parametri estrinseci

La matrice K descritta dall'equazione (2.7) è valida per camere poste all'origine del sistema di riferimento. Se consideriamo delle camere poste in un punto arbitrario dell'ambiente, quindi, è necessario che le coordinate dei punti, definite nel sistema di riferimento tridimensionale, siano ridefinite secondo il sistema di coordinate della camera. Questa trasformazione viene definita attraverso una *rototraslazione*, nello spazio, del sistema di riferimento dell'ambiente, in modo che coincida con quello della camera. Se $\tilde{\mathbf{M}}$ rappresenta le coordinate cartesiane di un punto nel sistema di riferimento dell'ambiente e $\tilde{\mathbf{M}}_{cam}$ rappresenta lo stesso punto nel sistema di riferimento della camera, si può scrivere

$$\tilde{\mathbf{M}}_{cam} = R(\tilde{\mathbf{M}} - C), \quad (2.8)$$

dove C rappresenta le coordinate del centro della camera nel sistema di riferimento dell'ambiente, R è una *matrice di rotazione* 3×3 che rappresenta l'orientamento della camera rispetto al sistema di riferimento dell'ambiente e K è la matrice di calibrazione della camera. L'equazione (2.8) può essere riscritta in forma matriciale: se definiamo \mathbf{M} e \mathbf{M}_{cam} come le rappresentazioni in coordinate omogenee di $\tilde{\mathbf{M}}$ e $\tilde{\mathbf{M}}_{cam}$, abbiamo

$$\mathbf{M}_{cam} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \mathbf{M},$$

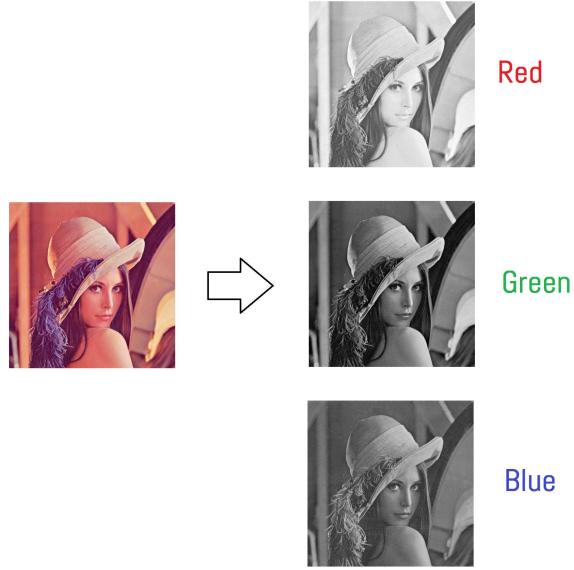


Figura 2.6: Esempio di estrazione dei canali RGB

da cui, riprendendo l'equazione 2.3,

$$\begin{aligned}\mathbf{m} &= K \begin{bmatrix} I & | & 0 \end{bmatrix} \mathbf{M}_{cam} \\ &= K \begin{bmatrix} I & | & 0 \end{bmatrix} \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \mathbf{M} \\ &= K \begin{bmatrix} R & | & -RC \end{bmatrix} \mathbf{M}.\end{aligned}$$

Possiamo, quindi, scrivere l'equazione della matrice di proiezione della camera come

$$P = K \begin{bmatrix} R & | & t \end{bmatrix},$$

dove $t = -RC$ prende il nome di *vettore di traslazione*.

2.2 Rappresentazione digitale delle immagini

Nel paragrafo 2.1.2 abbiamo introdotto il concetto di *immagine digitale* come rappresentazione numerica dell'immagine bidimensionale. Esistono vari modi di rappresentare un'immagine in formato digitale. Le rappresentazioni che a noi interessa considerare sono [3]:

- *scala di grigi*, in cui l'immagine è rappresentata come una matrice \mathbf{I} di dimensioni $H \times W$, dove l'elemento $\mathbf{I}(i, j)$ rappresenta il valore di *intensità luminosa (luma)* del pixel di coordinate (i, j) ;
- *RGB*, dove l'immagine è rappresentata come una matrice *tridimensionale* \mathbf{I}_{RGB} di dimensioni $H \times W \times 3$, in cui vengono memorizzati i

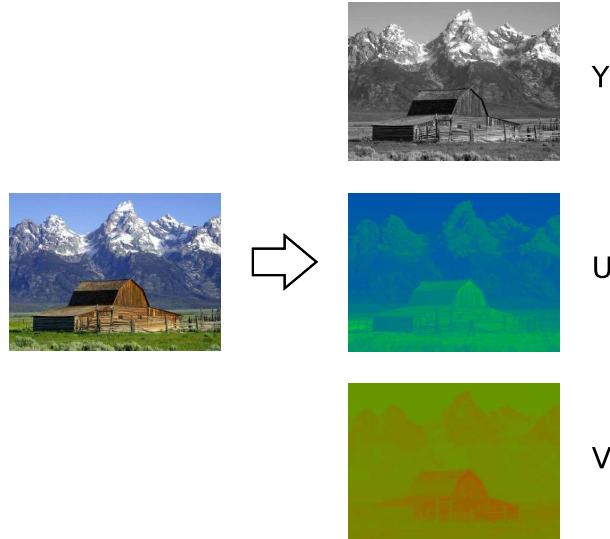


Figura 2.7: Esempio di estrazione dei canali YUV

livelli di intensità luminosa dei tre colori fondamentali *rosso* (R), *verde* (G) e *blu* (B) per ciascun pixel (figura 2.6);

- *YUV*, in cui lo spazio colore viene definito tramite un componente di luma (Y) e due componenti di *crominanza* (UV), che rappresentano rispettivamente i valori di differenza dal colore blu ($U = B - Y$) e dal colore rosso ($V = R - Y$) (figura 2.7).

Le immagini digitali possono essere memorizzate in diversi formati. Spesso vengono utilizzati algoritmi di *compressione*, che possono essere

- a perdita di informazione (*lossy*), come nel caso delle immagini *JPEG*;
- senza perdita di informazione (*lossless*), come nel caso dei file d'immagine *PNG* o *GIF*.

2.3 Tampering Detection

Nei moderni sistemi di videosorveglianza troviamo spesso algoritmi in grado di 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 [4]. Affinché questi algoritmi funzionino correttamente, è importante che l'*affidabilità* del sistema di acquisizione sia preservata. Questa proprietà viene soddisfatta quando:

- la camera mantiene la stessa inquadratura nell'area di interesse;

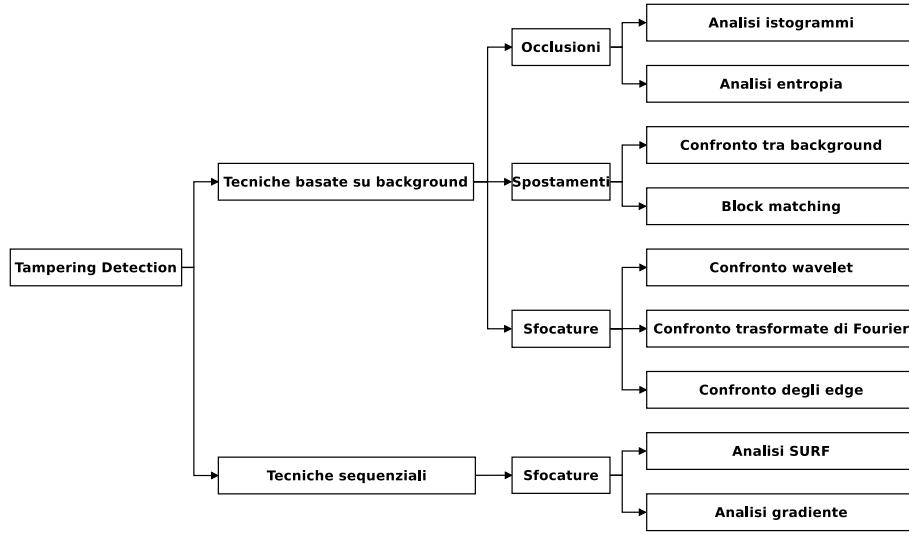


Figura 2.8: Tecniche di tampering detection

- tutti gli elementi della scena sono distinguibili all'interno dell'immagine.

Definiamo tampering un qualsiasi evento che determini un cambio di inquadratura o che non permetta la corretta acquisizione di una parte o della totalità della scena. Possiamo classificare gli eventi di tampering in quattro categorie:

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

La letteratura scientifica presenta molte tecniche in grado di affrontare il problema dell'identificazione di questo tipo di eventi. Lo schema in figura 2.8 mostra le principali tecniche di tampering detection presenti in letteratura. Queste possono essere divise in due categorie:

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

Nel seguito del paragrafo verranno descritti i principali metodi utilizzati in queste due categorie.

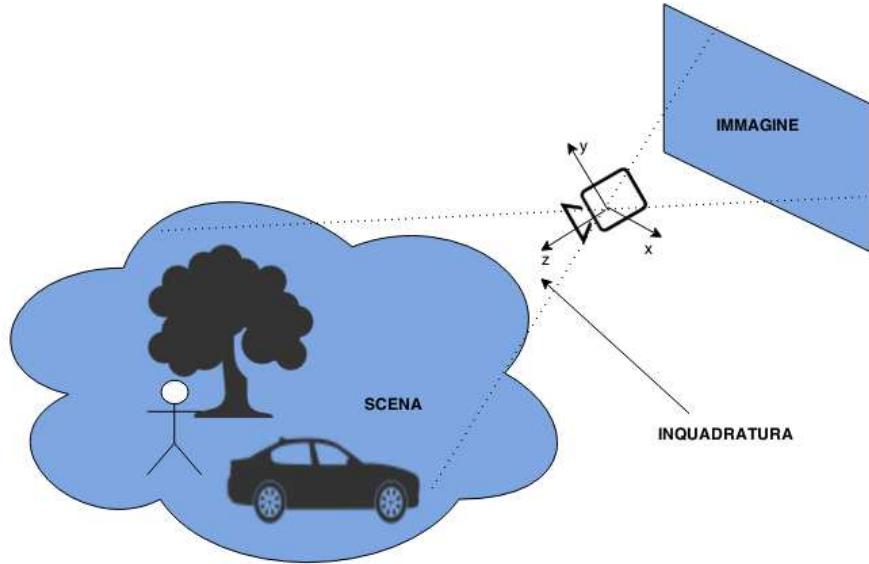


Figura 2.9: Sistema di monitoraggio video

2.3.1 Concetti e terminologia

Prima di concentrarci su come è stato affrontato il 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 t viene definita *immagine* o *frame* t -esimo. La figura 2.9 illustra questi concetti.

Per semplificare la trattazione, considereremo immagini estratte in *scala di grigi*. Ciascuna immagine, quindi, verrà rappresentata come una matrice di pixel, in cui ciascun elemento rappresenta l'intensità luminosa (*luma*) del pixel corrispondente. L'estensione al caso di immagini *RGB* è immediata: possiamo, infatti, applicare i vari algoritmi in maniera indipendente per ciascun canale di colore, per poi mediare i risultati ottenuti.

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 $i = 1, \dots, \infty$. In particolare, per indicare il valore della *luminosità* del pixel x per il frame t -esimo, useremo

il termine $z_t(x)$, con

$$z_t(x) \in [0, 255] \forall x \in \mathcal{X}.$$

2.3.2 Tecniche basate su confronto di background

Nella maggior parte dei lavori dedicati al problema del tampering detection, il metodo principalmente utilizzato consiste nel confrontare ciascun frame con un modello che viene calcolato a partire dalle osservazioni precedenti. Tale metodo è ampiamente utilizzato in vari ambiti della visione artificiale e prende il nome di *background subtraction*. Una tecnica generale di calcolo del background è presentata in [5]. Il valore del modello di background per il pixel x è calcolato, in maniera ricorsiva, secondo la seguente formula:

$$B_{t+1}(x) = \begin{cases} aB_t(x) + (1-a)z_t(x) & , \text{ se } |z_t(x) - z_{t-1}(x)| \leq T_t(x) \\ B_t(x) & , \text{ se } |z_t(x) - z_{t-1}(x)| > T_t(x) \end{cases}, \forall x \in \mathcal{X},$$

dove $0 < a < 1$ è chiamato *parametro di aggiornamento (update parameter)* e $T_t(x)$ 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:

$$T_{t+1}(x) = \begin{cases} aT_t(x) + (1-a)(c|z_t(x) - B_t(x)|) & , \text{ se } |z_t(x) - z_{t-1}(x)| \leq T_t(x) \\ T_t(x) & , \text{ se } |z_t(x) - z_{t-1}(x)| > T_t(x) \end{cases}, \forall x \in \mathcal{X},$$

dove $c > 1$ e $0 < a < 1$. Lo stesso modello viene utilizzato in altri lavori, come ad esempio in [6] e [7], mentre una variante molto usata consiste nel calcolare il modello di background a partire dall'*estrazione dei contorni* di ciascun frame ([8], [9]). Un approccio diverso, ma comunque riconducibile allo stesso filone, lo troviamo in [10], 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 tampering.

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 [5] e [6] questo evento è associato a un cambiamento nella struttura dell'*istogramma* del frame occluso rispetto a quello del background. In particolare, se z_t è un frame in cui è avvenuta un'occlusione, ci si aspetta che il suo istogramma contenga i valori più alti in un intervallo più ristretto rispetto a quello del background B_t , in quanto la maggior parte dell'immagine prenderà il colore dell'oggetto occludente o diventerà più scura.

Indichiamo con $H_b(\cdot)$ il valore del b -esimo bin dell'istogramma di un frame,

con $1 \leq b \leq 32$. Per identificare un evento di occlusione vengono considerate due disequazioni:

$$\begin{aligned} & (H_{\max(H(z_t))-1}(z_t) + H_{\max(H(z_t))}(z_t) + H_{\max(H(z_t))+1}(z_t)) \\ & > Th_1 (H_{\max(H(z_t))-1}(B_t) + H_{\max(H(z_t))}(B_t) + H_{\max(H(z_t))+1}(B_t)), \\ & \sum_{b=1}^{32} H_b(|z_t - B_t|) > Th_2 \sum_{b=1}^k H_b(|z_t - B_t|), \end{aligned}$$

dove $Th_1 > 1$, $Th_2 > 1$ e $0 \leq k < 32$ sono soglie che possono essere modificate in base alla sensibilità che si richiede da parte dell'algoritmo. Un approccio simile è presente in [8], [9] e [11], 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.

Per riuscire a identificare delle occlusioni *parziali* le tecniche descritte sopra non sono efficaci, in quanto tali eventi non sono in grado di modificare in maniera sostanziale la struttura dell'istogramma dei frame. Una soluzione ([9]) consiste nel dividere ciascuna immagine in un certo numero di *blocchi* della stessa dimensione, in modo da applicare le tecniche descritte sopra per ciascuna partizione.

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 [6] il metodo proposto per identificare uno spostamento della camera consiste nel confrontare l'immagine di background B_t con B_{t-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:

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

Lo spostamento della camera viene identificato quando $P > Th \cdot K$, 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*. 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. In [9] e [8]

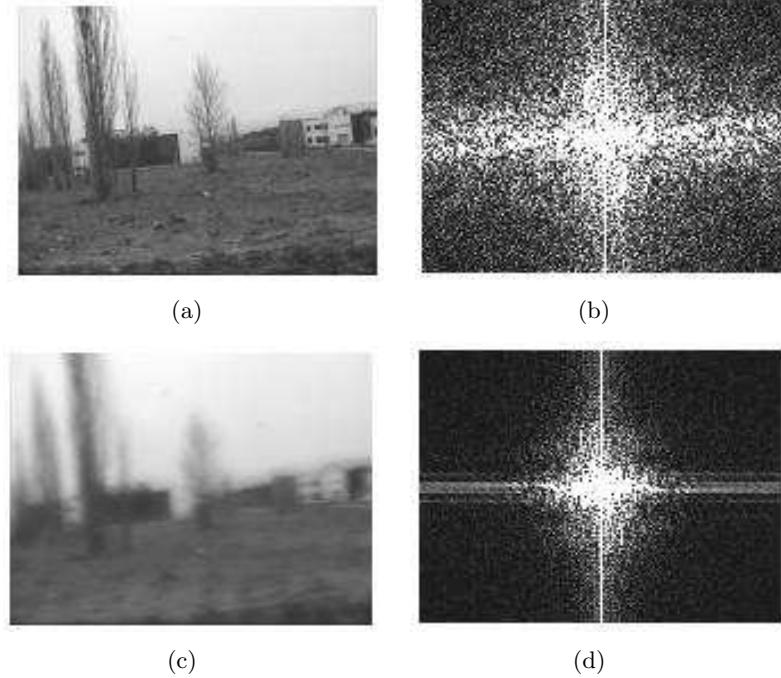


Figura 2.10: Comportamento della trasformata di Fourier nel caso di sfocatura

l’algoritmo calcola il vettore di spostamento m del frame corrente rispetto al background, e calcola la *zero-mean normalized cross correlation* (ZNCC) [12]:

$$ZNCC_t(m) = \frac{\sum_{x \in \mathcal{X}} (B_{t-1}(x) - \mu_{B_{t-1}})(z_t(x+m) - \mu_{z_t})}{\sigma_{B_{t-1}} \sigma_{z_t}},$$

dove μ_{z_t} e σ_{z_t} rappresentano la media e la deviazione standard della luminosità nell’immagine z_t . 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 [13], con la differenza che, invece di analizzare la correlazione dei pixel, viene analizzata quella degli istogrammi.

Identificazione di sfocature

La conseguenza di un evento di sfocatura è la perdita di dettagli nell’immagine. In [5] e [6] questo fenomeno è associato a una diminuzione dell’energia ad alta frequenza. Per analizzare questo cambiamento, [5] confronta ciascun frame con il background nel dominio delle *wavelet* [14], mentre [6] utilizza il dominio della *trasformata di Fourier* [15]. Nella figura 2.10 vediamo un esempio di come si comporta la trasformata di Fourier nel caso di sfocature: nel passaggio da un frame nitido (Fig. 2.10(a)) a uno sfocato (Fig. 2.10(c))

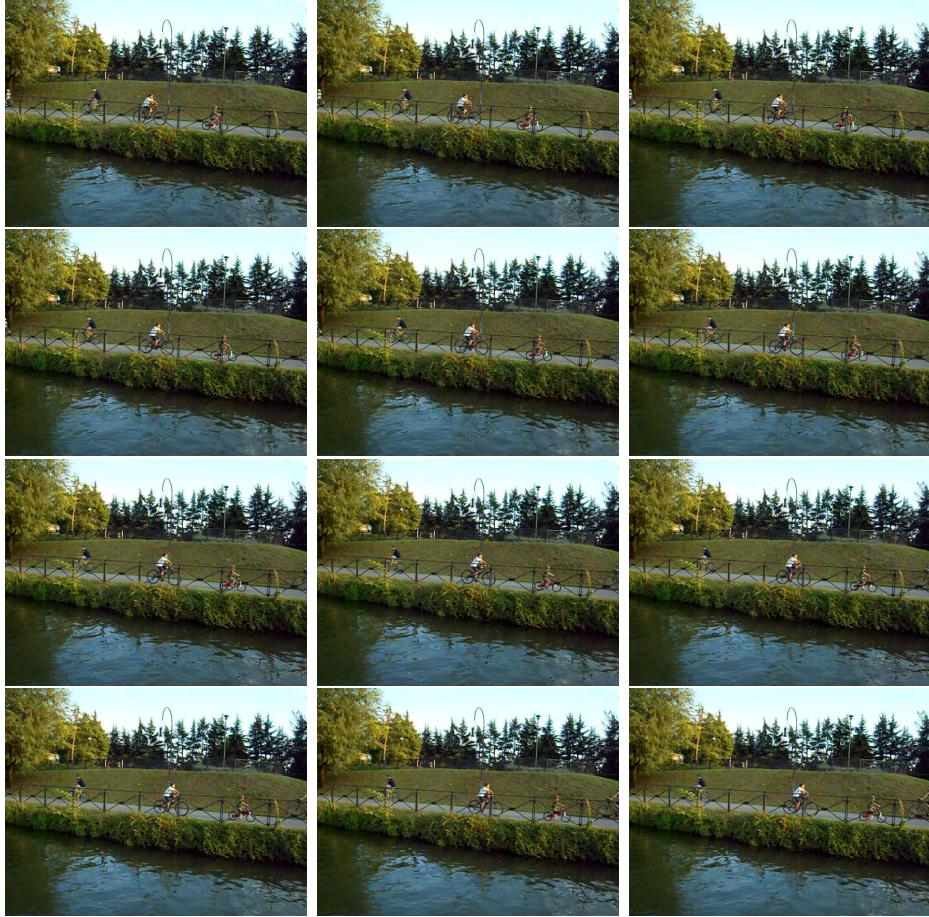


Figura 2.11: Sequenza di frame acquisiti a 30 fps

abbiamo un crollo delle componenti ad alta frequenza nelle trasformate di Fourier (rispettivamente Fig. 2.10(b) e Fig. 2.10(d)). L'evento di tampering viene individuato quando l'*energia media* della trasformata di Fourier (o di quella wavelet) del frame corrente $E_{HF}(z_t)$ è Th volte minore di quella del background $E_{HF}(B_t)$:

$$E_{HF}(z_t) \leq Th \cdot E_{HF}(B_t),$$

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 [11], [9], [8] e [13], consiste nell'estrare i contorni dalle immagini secondo il metodo di Sobel [16] o di Canny [17], e confrontare il numero di pixel dei contorni con quelli del background. Quando il numero di pixel dei contorni del frame corrente $\sum edges(z_t)$ è Th volte più piccolo

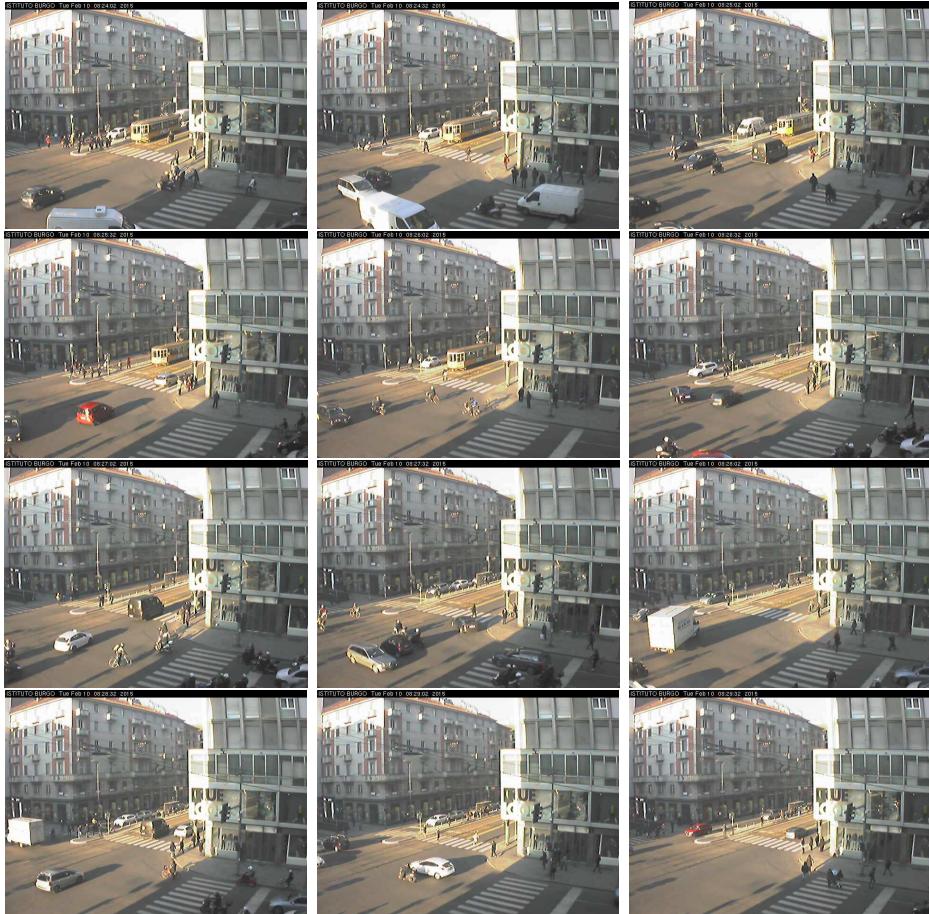


Figura 2.12: Sequenza di frame acquisiti ogni 30 secondi

di quello del background $\sum edges(B_t)$:

$$\sum edges(z_t) \leq Th \sum edges(B_t),$$

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

2.3.3 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 2.11 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 evento di tampering sulla camera. La figura 2.12 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 non permette l’impiego di un modello di background per il problema del tampering detection. Evidenziamo, inoltre, che le tecniche viste finora richiedono che il sistema di monitoraggio possieda elevate capacità computazionali, in quanto l’aggiornamento del background e l’analisi di tampering detection non può rallentare la frequenza di acquisizione.

Un approccio alternativo consiste nel monitorare nel tempo il comportamento di alcuni indicatori estratti dalle singole immagini acquisite. Si presuppone che, quando il sistema di monitoraggio opera in condizioni di funzionamento *ottimali*, gli indicatori analizzati 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) [18] o *change-detection test* [19].

Troviamo alcuni esempi nell’identificazione di sfocature: in [7] la soluzione consiste nel monitorare nel tempo il *numero di SURF* [20], 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 [21], dove le sfocature vengono identificate monitorando l’*energia media del gradiente* delle immagini acquisite:

$$m_t = \mathcal{M}[z_t] = \int_{\mathcal{X}} \|\nabla z_t(x)\|_1 dx,$$

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) [22].

Il test statistico utilizzato non richiede alcuna informazione *a priori* del processo che si sta monitorando, e sfrutta una sequenza iniziale $\{m_t\}, t = 1, \dots, T$ di indicatori estratti da frame non sfocati, in modo da configurare automaticamente i suoi parametri. Tale sequenza prende il nome di *training set* e, assumendo che i dati abbiano una distribuzione *gaussiana*, permette di stimare i parametri di m_t in assenza di sfocature che determinano l’*ipotesi nulla* (indicata con Θ^0), e di definire le *ipotesi alternative*, indicate con Θ^1 , che identificano qualsiasi cambiamento non stazionario.

Per garantire una stima accurata dei parametri, [21] consiglia di utilizzare un training set ampio, ad esempio $T > 400$. Il test opera su *sotto-sequenze* di misure m_t (ad esempio da 20 misure) e stima la transizione da Θ^0 a Θ^1 misurando la *log-verosimiglianza* tra la pdf in assenza di sfocature e le pdf delle varie ipotesi alternative nella sotto-sequenza τ , ovvero

$$r(\tau) = \ln \frac{N_{\Theta^1}(\phi_\tau)}{N_{\Theta^0}(\phi_\tau)},$$

dove ϕ_τ è il valore medio della misura m_t nella sotto-sequenza τ , e N_Θ è la *distribuzione gaussiana multivariata* parametrizzata in Θ .

Il metodo CUSUM considera la *somma cumulata*

$$S(\tau) = \sum_{t=1}^{\tau} r(t)$$

e identifica un cambiamento in m_t quando

$$g(\tau) = S(\tau) - \nu(\tau),$$

ovvero la differenza tra il valore della somma cumulata all'istante τ e il suo valore minimo nel tempo

$$\nu(\tau) = \min_{t=1,\dots,\tau} S(t),$$

superà una certa soglia h .

L'utilizzo di un descrittore leggero come l'energia media del gradiente permette di operare con una bassa complessità computazionale, tanto da poter essere utilizzato come soluzione a livello di nodo per una *wireless multimedia sensor network* (WMSN) [23]. D'altro canto, confrontato con le tecniche descritte nel paragrafo 2.3.2, l'utilizzo di tecniche sequenziali genera un numero più alto di *falsi allarmi*, il che si traduce, all'interno di una rete di sensori, in un aumento dei messaggi di allarme inviati, che devono essere considerati, quindi, come un costo.

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

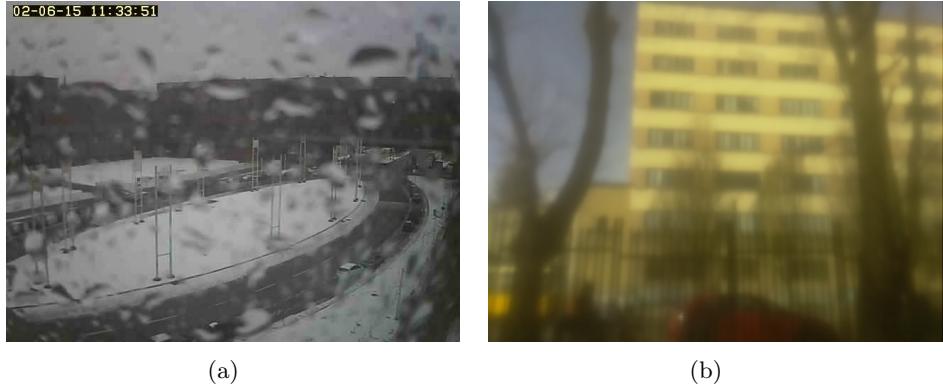
Il nostro campo di osservazione si concentra su quegli eventi che si interpongono tra la scena ripresa da una camera e il sensore che acquisisce le immagini. Non vogliamo, cioè, identificare degli eventi particolari che avvengono nella scena, come un oggetto lasciato incustodito [4], bensì vogliamo identificare quegli eventi che non permettano al sensore 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 Sfocatura

Il fenomeno della sfocatura avviene quando un elemento trasparente o semi-trasparente 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;



(a)

(b)

Figura 3.1: Esempi di sfocature

- 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 riguardi 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 la trattazione presente in [21], questo fenomeno può essere modellato come un operatore di *degradazione* \mathcal{D} applicato a un'immagine y , considerata priva di errori, i.e.,

$$z = \mathcal{D}[y]. \quad (3.1)$$

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 η corrispondente al rumore, i.e.,

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

dove, come abbiamo specificato nel paragrafo 2.3.1, indichiamo con x le coordinate dei *pixel* dell'immagine e con \mathcal{X} l'insieme dei pixel che formano l'immagine. Per praticità possiamo assumere che l'operatore di blur sia *lineare*. Quindi, considerando l'immagine come un dato *continuo*,

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



Figura 3.2: Sequenza di otto frame consecutivi acquisiti ogni minuto

dove $h(x, \cdot)$ rappresenta la *risposta all'impulso*, *point spread function* (PSF), della sfocatura sul pixel x . L'effetto di $h(x, \cdot)$ è quello di rendere le differenze di intensità, tra pixel adiacenti, più morbide (*smooth*). Nel caso in cui la sfocatura sia applicata sulla totalità dell'immagine e fosse *spazio invariante* (come nel caso della figura 3.1(b)), allora è possibile modellare l'operatore di blur come una *convoluzione*¹:

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

dove $h(\cdot)$ è un filtro gaussiano o uniforme.

Nel caso più generale possiamo considerare che la camera acquisisca una sequenza di N osservazioni (dove N può tendere all'infinito) $\{z_t\}, t = 1, \dots, N$, quindi la formula (3.2) si può riscrivere come

$$z_t(x) = \mathcal{D}_t[y_t](x) = \mathcal{B}_t[y_t](x) + \eta(x), \quad x \in \mathcal{X}. \quad (3.5)$$

La sequenza delle immagini $\{y_t\}, t = 1, \dots, N$, 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, a causa del continuo passaggio di automobili e pedoni. Questo problema fa sì che l'identificazione delle sfocature non possa avvenire facendo un semplice confronto 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 ($z_t \neq z_{t+1}$), sarebbe difficile capire se è cambiato il contenuto delle immagini ($y_t \neq y_{t+1}$) o l'operatore di sfocatura ($\mathcal{B}_t \neq \mathcal{B}_{t+1}$).

3.1.2 Spostamento della camera

Lo spostamento della camera avviene quando cambia la sua inquadratura. Le cause possono essere, ancora una volta, di tipo naturale, ad esempio una

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



(a)

(b)

Figura 3.3: Esempio di spostamento della camera

raffica di vento che sposta la camera, oppure dovute a un intervento umano, uno spostamento intenzionale per evitare che la scena venga ripresa. 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 $\{y_t\}$ di immagini generate da una camera in una certa posizione, e la sequenza $\{w_t\}$ di immagini generate dalla stessa camera in una posizione differente.

Possiamo modellare la sequenza di immagini $\{z_t\}$ in cui avviene uno spostamento della camera all'istante T^* nel seguente modo:

$$z_t(x) = \begin{cases} y_t(x) + \eta(x) & \text{per } t < T^* \\ w_t(x) + \eta(x) & \text{per } t \geq T^* \end{cases}, \quad (3.6)$$

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 molto bassi (come ad esempio un'immagine al minuto), possiamo considerare lo spostamento come istantaneo e, quindi, tenere come riferimento il modello (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 allarmi, come verrà mostrato dalla fase sperimentale.



Figura 3.4: Esempi di occlusione

3.1.3 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. Esempi di occlusione sono illustrati in figura 3.4. Può succedere (Fig. 3.4(a)) che un oggetto venga posto intenzionalmente davanti alla lente della camera, in modo da coprire la scena, oppure (Fig. 3.4(b)), a causa di una nevicata, può avvenire che della neve si depositi sulla lente e, quindi, venga compromessa la corretta acquisizione.

Quando parliamo di guasti, invece, possiamo considerare due casi:

- guasto della *trasmissione*;
- guasto del *sensore*.

Il primo caso può avvenire quando considerano delle WMSN, in particolare quando i frame vengono trasmessi attraverso la rete dai sensori verso un nodo centrale. Quando avviene un guasto nella rete parte dell’immagine non viene trasmessa, come illustrato nella figura 3.5(a). Il caso del sensore guasto, come ad esempio quello nella figura 3.5(b), comporta generalmente un aumento del rumore presente nel frame.

Questi problemi non sono stati affrontati durante il corso della 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

Nel paragrafo 2.3 abbiamo introdotto in maniera molto generale il concetto di tampering detection. In questo paragrafo diamo una definizione più formale al problema. L’algoritmo di tampering detection consiste nell’analizzare la sequenza dei frame generati dalla camera in modo da determinare



Figura 3.5: Esempi di guasti

un possibile cambiamento nell'inquadratura della scena o una sfocatura. Consideriamo il caso generale di una sequenza di frame $\{z_t\}, t = 1, \dots, \infty$. Per $t < T^*$ abbiamo che i frame vengono acquisiti in condizioni di funzionamento normale:

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

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

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

mentre nel caso di uno spostamento della camera avremo

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

L'obiettivo dell'algoritmo è quello di stimare l'istante T^* .

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, in quanto stiamo operando con framerate bassi. Non utilizzeremo, quindi, le tecniche basate su background che sono state descritte nel paragrafo 2.3.2. Nel prossimo capitolo vedremo la soluzione che abbiamo deciso di utilizzare per lo sviluppo dell'algoritmo di tampering detection.

Capitolo 4

Soluzione proposta

In questo capitolo descriviamo la soluzione che abbiamo proposto per risolvere il problema del tampering detection, formulato nel capitolo 3.

4.1 Indicatori utilizzati per identificare gli eventi di tampering

4.1.1 Misura della sfocatura nell'immagine

Nel paragrafo 3.1.1 abbiamo modellizzato il fenomeno della sfocatura secondo la formula (3.5):

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

Ricavare un indicatore in grado di misurare direttamente il grado di sfocatura di un'immagine è difficile. Quello che è possibile fare, come proposto in [21], è utilizzare una misura *indiretta* di questo operatore.

Come abbiamo visto nel paragrafo 3.1.1, l'operatore di sfocatura \mathcal{B} ha come effetto principale quello di rendere le differenze di intensità tra pixel adiacenti più morbide (*smooth*). In base a questo è possibile identificare un evento di sfocatura andando a monitorare l'*energia media del gradiente* di ciascuna immagine:

$$g_t = \mathcal{G}[z_t] = \frac{\sum_{\mathcal{X}} \|\nabla z_t(x)\|_2^2}{|\mathcal{X}|}, \quad (4.1)$$

dove abbiamo indicato con $|\mathcal{X}|$ la *cardinalità* dell'insieme dei pixel \mathcal{X} , e con $\|\cdot\|_2$ la norma di tipo \mathcal{L}_2 ¹.

Lavorando nel dominio discreto delle immagini digitali, possiamo calcolare le derivate dell'intensità luminosa (*luma*) per mezzo di convoluzioni con filtri

¹ $\|x\|_2 = \sqrt{\sum_t x_t^2}$

derivativi. In particolare, per il calcolo delle derivate orizzontali abbiamo utilizzato il seguente filtro f_i :

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

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

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

dove abbiamo indicato con \circledast l'operatore di convoluzione e con f un *filtro gaussiano* di dimensione 5×5 :

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

dove $k = 2$ e la deviazione standard $\sigma = 1$. Con questi filtri è possibile calcolare la *norma del gradiente* nel seguente modo:

$$\|\nabla z_t(x)\|_2^2 = (z_t \circledast f_i)(x)^2 + (z_t \circledast f_j)(x)^2.$$

Una volta calcolata la norma del gradiente è possibile farne la media come specificato nella formula (4.1). Il risultato finale è un indicatore *scalare* per ciascun frame acquisito, che può essere monitorato per individuare eventi di sfocature. In particolare l'evento è associato a un *crollo* del valore di g .

4.1.2 Misura del displacement

Nel paragrafo 3.1.2 abbiamo modellizzato il fenomeno dello spostamento della camera secondo la formula (3.6)

$$z_t(x) = \begin{cases} y_t(x) + \eta(x) & \text{per } t < T^* \\ w_t(x) + \eta(x) & \text{per } t \geq T^* \end{cases}, x \in \mathcal{X}$$

dove T^* indica l'istante in cui avviene il cambiamento.

Uno spostamento della camera, quindi, è associato a un cambiamento *globale* dei valori di intensità luminosa (*luma*) nei pixel dell'immagine. In base a questo è possibile identificare un evento di spostamento della camera andando a monitorare l'*energia media della luma* di ciascuna immagine:

$$l_t = \mathcal{L}[z_t] = \frac{\sum_{x \in \mathcal{X}} z_t(x)}{|\mathcal{X}|}, \quad (4.2)$$

dove abbiamo indicato con $|\mathcal{X}|$ la *cardinalità* dell'insieme dei pixel \mathcal{X} .

Il risultato finale è un indicatore *scalare* per ciascun frame acquisito, che può essere monitorato per individuare eventi di spostamenti della camera.



(a)

(b)

Figura 4.1: Esempio di cambi di luminosità tra il giorno e la notte



(a)

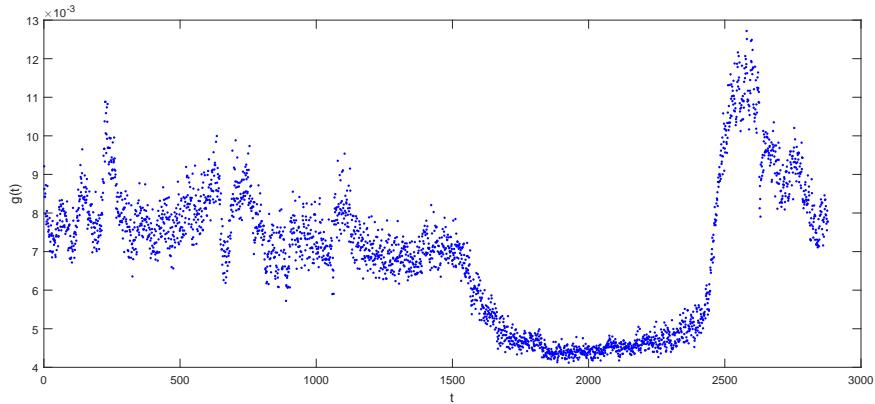
(b)

Figura 4.2: Esempio di presenza di sfocature dovute all'aumento del tempo di esposizione della camera

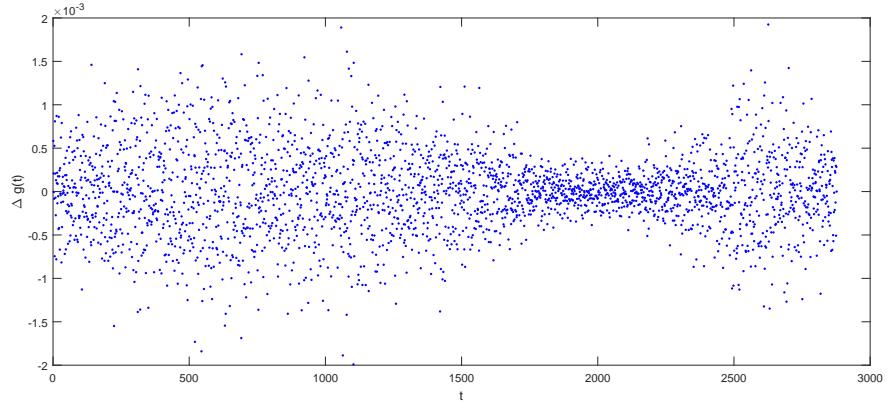
4.1.3 Comportamento degli indicatori nel tempo

Analizzando alcune sequenze video abbiamo notato che, anche in assenza di eventi di tampering, vi sono alcuni fattori che sono in grado di far variare il valore degli indicatori estratti. Tra i più importanti abbiamo:

- *Cambi di luminosità* che avvengono nel corso della giornata. Se consideriamo l'esempio in figura 4.1, possiamo notare come, nel passaggio dal giorno (figura 4.1(a)) alla notte (figura 4.1(b)), le differenze di luminosità siano elevate.
- *Dinamicità della scena*. La ripresa di una scena movimentata, come ad esempio una strada, ha come risultato che ciascun frame sia diverso dagli altri. Ciò si traduce in una variabilità elevata degli indicatori che abbiamo utilizzato. Inoltre, col passare del tempo, può succedere che cambi anche il *grado di dinamicità* della scena. Considerando ancora l'esempio della strada, infatti, avremo dei momenti in cui il traffico è



(a)

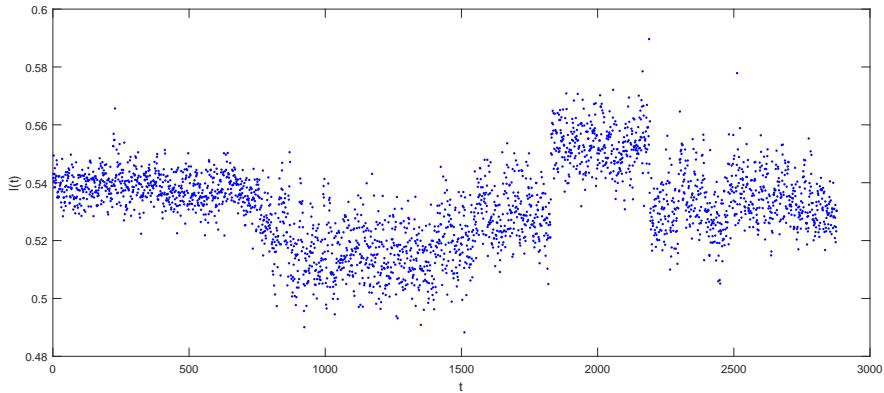


(b)

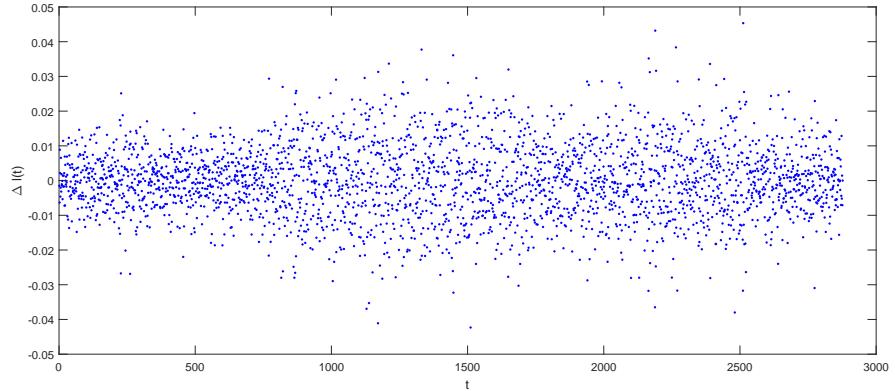
Figura 4.3: Energia media del gradiente lungo un'acquisizione di 24 ore (a) e suo detrending (b)

più intenso (nelle cosiddette *ore di punta*) e altri in cui le macchine passano meno spesso (tipicamente durante la notte).

- *Configurazione automatica della camera.* Solitamente le camere sono in grado di configurare in maniera automatica alcuni parametri, in base alle condizioni di luminosità esterne. Ad esempio, durante la ripresa di scene notturne la camera solitamente aumenta il *tempo di esposizione* del sensore, in modo da ricevere più luce possibile. Ciò si traduce in un *aumento del rumore* su tutta la scena acquisita (come vediamo nel frame in figura 4.1(b)) e in una *presenza di sfocature* quando vengono immortalati degli oggetti in movimento (come ad esempio le macchine che si muovono nella figura 4.2(b)).



(a)



(b)

Figura 4.4: Energia media della luma lungo un'acquisizione di 24 ore (a) e suo detrending (b)

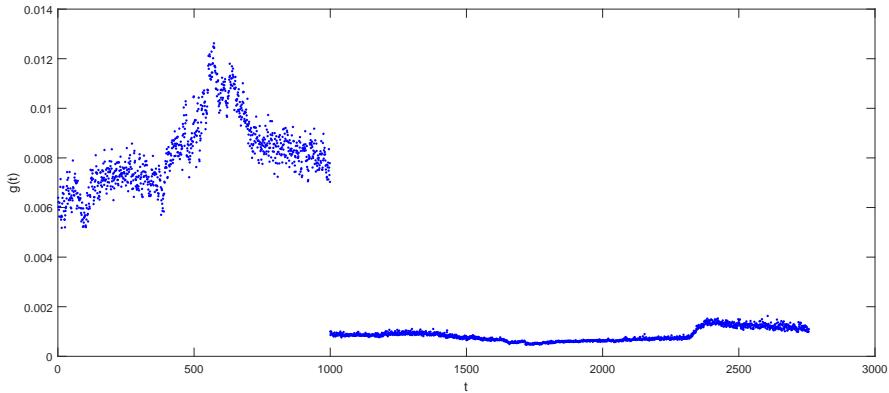
Questi fenomeni fanno sì che i nostri indicatori abbiano una dinamica *diffilmente prevedibile* e che *non siano stazionari*, come possiamo vedere negli esempi delle figure 4.3(a) e 4.4(a). Di conseguenza non è possibile applicare direttamente le tecniche di CDT come fatto in [21].

Per eliminare le componenti in bassa frequenza possiamo fare un *detrending* di ciascun segnale calcolando la *differenza* tra l'istante corrente e quello precedente. Nel caso dell'energia media del gradiente avremo

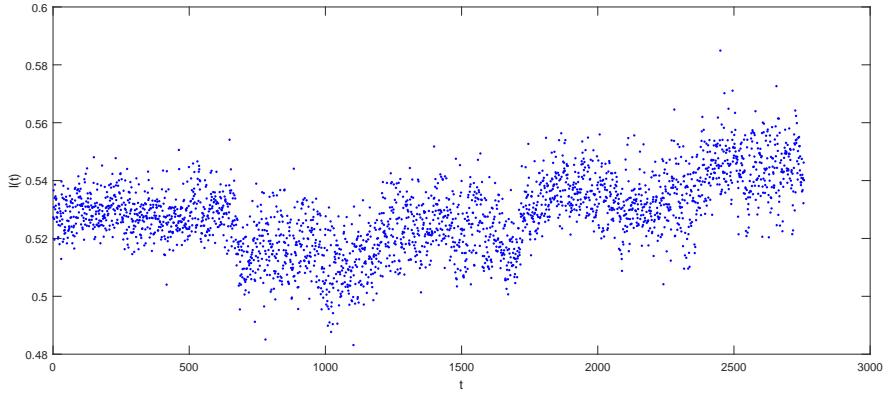
$$\Delta g_t = g_t - g_{t-1},$$

mentre per l'energia media della luma avremo

$$\Delta l_t = l_t - l_{t-1}.$$



(a)



(b)

Figura 4.5: Energia media del gradiente (a) e della luma (b) nel caso di una sfocatura

Vediamo un esempio di come si comporta il detrending sui nostri indicatori nelle figure 4.3(b) e 4.4(b). Di conseguenza non è possibile applicare direttamente le tecniche di CDT come fatto in [21].

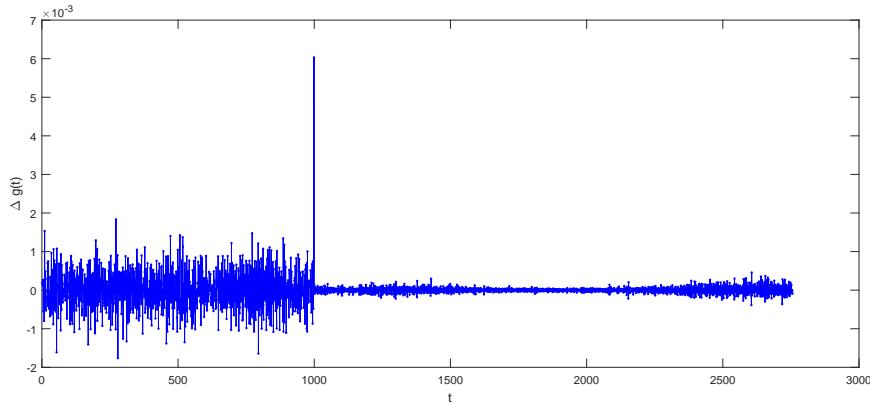
Per eliminare le componenti in bassa frequenza possiamo fare un *detrending* di ciascun segnale calcolando la *differenza* tra l'indicatore all'istante corrente e quello precedente. Nel caso dell'energia media del gradiente avremo

$$\Delta g_t = g_t - g_{t-1},$$

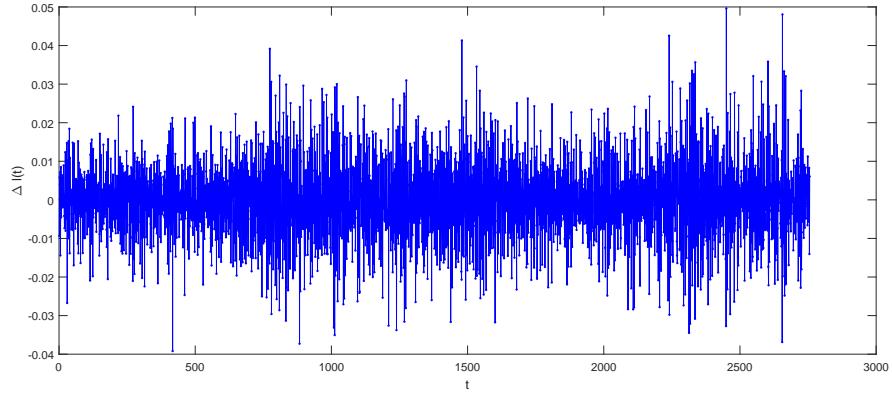
mentre per l'energia media della luma avremo

$$\Delta l_t = l_t - l_{t-1}.$$

Vediamo un esempio di come si comporta il detrending sui nostri indicatori nelle figure 4.3(b) e 4.4(b). Possiamo notare come le fluttuazioni in bassa



(a)



(b)

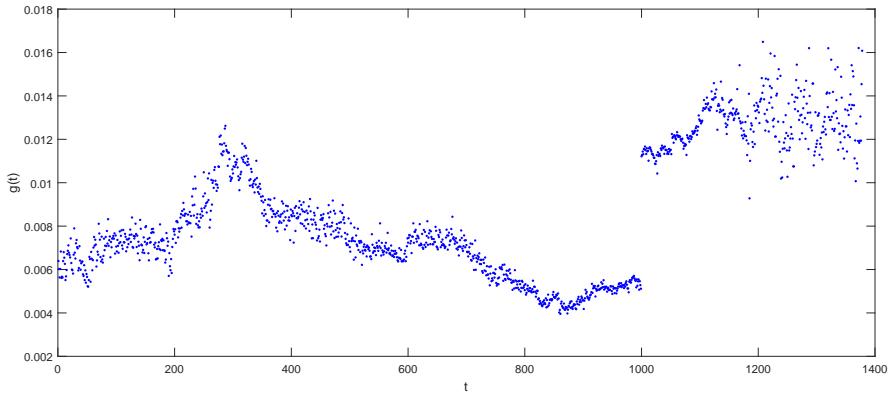
Figura 4.6: Detrending dell'energia media del gradiente (a) e della luma (b) nel caso di una sfocatura

frequenza vengano lavate via dal detrending, dato che in questo modo consideriamo solamente le differenze tra misure consecutive.

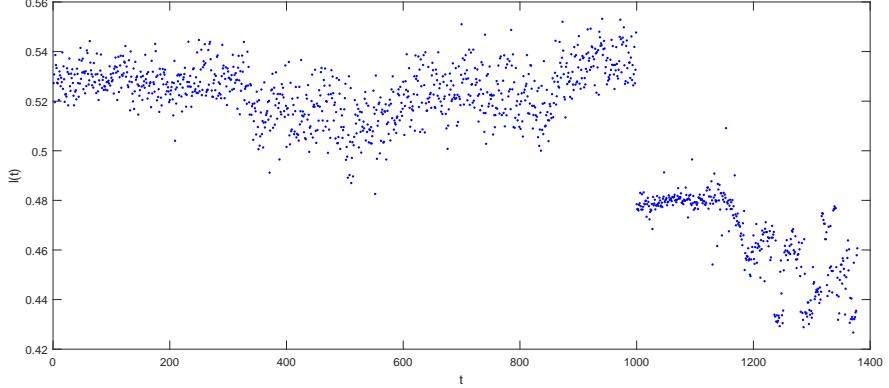
Come abbiamo detto prima, lo scopo del monitoraggio di questi indicatori è quello di individuare degli eventi di tampering. Nelle figure 4.5 e 4.7 possiamo vedere il comportamento dell'energia media di luma e gradiente rispettivamente per un caso di sfocatura e per un caso di spostamento della camera. In particolare, in entrambi i casi l'evento di tampering avviene al frame 1000². Il detrending di questi segnali è visualizzato nelle figure 4.6 e 4.8.

Possiamo vedere come, in generale, l'evento di tampering sia associato a un brusco salto o a un crollo *istantaneo* del valore di uno o entrambi gli

²Il modo in cui sono stati ottenuti gli eventi di tampering è descritto nel capitolo 5



(a)

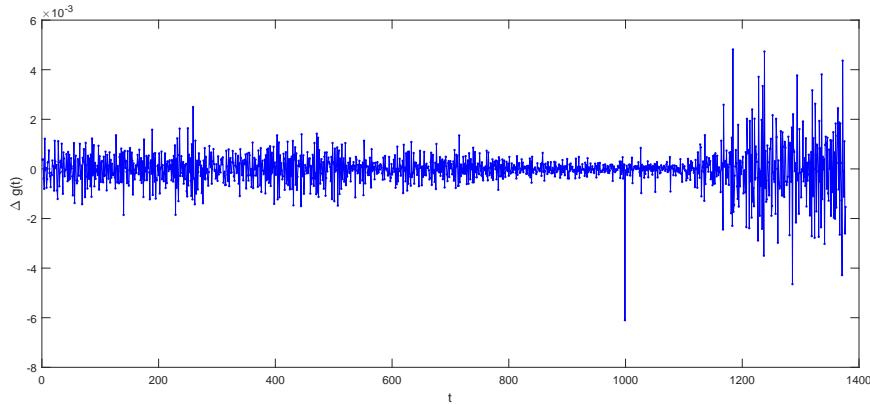


(b)

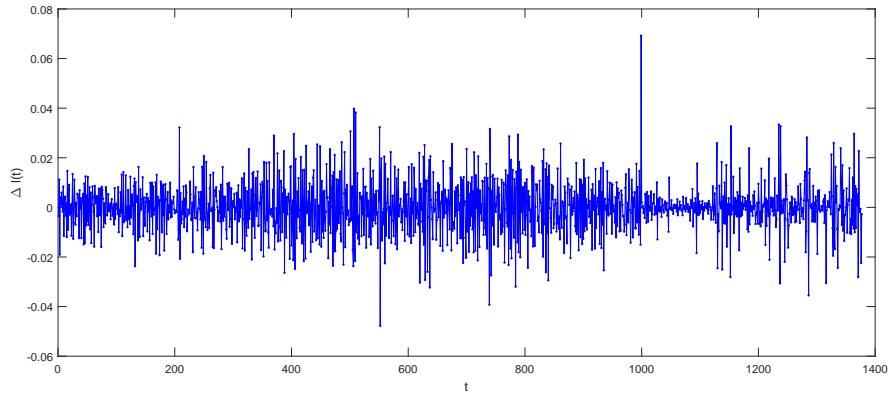
Figura 4.7: Energia media del gradiente (a) e della luma (b) nel caso di uno spostamento della camera

indicatori. Vanno notate alcune cose:

- L'evento di sfocatura non si traduce in un cambiamento nell'energia media della luma. Questo avviene perché questo tipo di tampering ha come effetto principale quello di rendere più morbide le differenze di intensità tra i pixel dell'immagine, mantenendo comunque il valore medio della luma invariato.
- Facendo il detrending della sequenza abbiamo dei valori che oscillano attorno allo zero, un picco in corrispondenza dell'istante in cui avviene il tampering (figure 4.6(a), 4.8(a), 4.8(b)) e infine altri valori che oscillano attorno allo zero. Questo è dato dal fatto che il detrending considera le differenze tra dati consecutivi e, quindi, la differenza



(a)

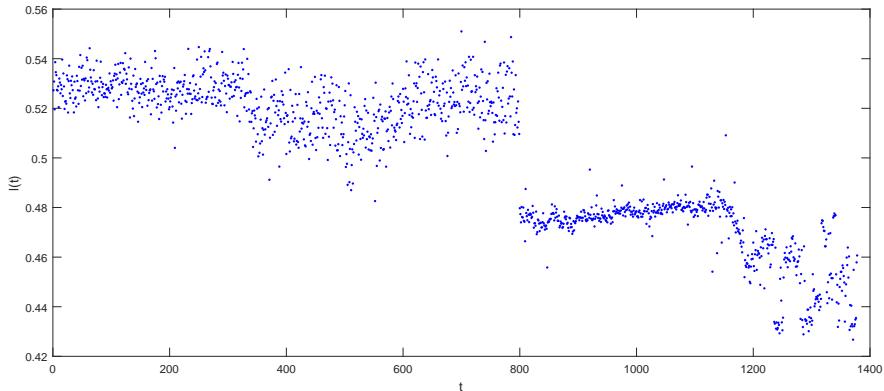


(b)

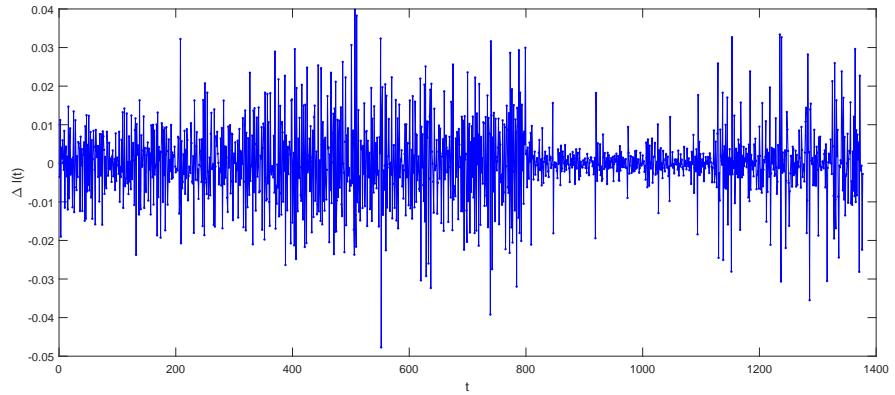
Figura 4.8: Detrending dell'energia media del gradiente (a) e della luma (b) nel caso di uno spostamento della camera

maggiore si ha proprio nell'istante in cui inizia l'evento di tampering, mentre solitamente le differenze tra misure consecutive sono minime.

- Questo fa sì che monitorare il detrending delle sequenze degli indicatori permetta di identificare più facilmente gli eventi di tampering rispetto ad analizzare la sequenza originale. Il risvolto della medaglia è che i cambiamenti *persistenti*, come quelli che stiamo considerando noi, diventano dei cambiamenti *istantanei*.
- Considerando l'evento di sfocatura, possiamo notare (figura 4.5(a)) come, in seguito all'evento, oltre ad aver un crollo istantaneo del valore dell'energia del gradiente, abbiamo anche un *abbassamento della sua varianza*. Questo permette di utilizzare tecniche di monitoraggio se-



(a)



(b)

Figura 4.9: Esempio di sequenza dell'energia media della luma (a) e del suo detrending (b) con un displacement difficile da identificare

quenziale sull'energia del gradiente, con dei CDT sulla varianza, per individuare eventi di sfocatura.

In definitiva, è possibile identificare un evento di tampering andando a monitorare il detrending degli indicatori descritti dalla formule (4.1) e (4.2). Il monitoraggio può essere fatto con tecniche a *basso carico computazionale*: infatti è possibile utilizzare semplicemente una soglia in modo da individuare il picco nel detrending dovuto all'evento di tampering. In particolare possiamo monitorare l'energia media del gradiente per individuare eventi di sfocatura, e l'energia media della luma per individuare eventi di spostamento della camera. Inoltre, per rendere più robusta l'identificazione di sfocature, è possibile usare un test sequenziale sull'energia media del gradiente in grado di individuare dei cambiamenti nella varianza.



(a)

(b)

Figura 4.10: Esempio di spostamento della camera

Dobbiamo fare un’ultima considerazione sullo spostamento della camera. Ci possono essere dei casi in cui l’evento è difficilmente individuabile monitorando la scena nella sua totalità. Un esempio è illustrato nella figura 4.9, dove vediamo un caso di spostamento della camera che avviene al frame 800. Questo evento, però, non si traduce, nel detrending del segnale, in un picco che si eleva rispetto agli altri, come nel caso della figura 4.8(b). Questo problema capita perché stiamo monitorando l’energia media della luma calcolata sulla *totalità* della scena. Possiamo avere situazioni, come nel caso della figura 4.9, in cui lo spostamento della camera non determina un cambiamento sostanziale nella luminosità media della scena.

Questo problema viene meno se consideriamo il contributo dell’energia della luma mediato non sulla totalità della scena, bensì su *regioni* specifiche analizzate separatamente. Infatti, se consideriamo un’area particolare della scena, come ad esempio quella occupata dal palazzo nella figura 4.10(a), durante uno spostamento della camera la variazione della luma mediata solo sui pixel appartenenti a quella regione sarà più marcata rispetto a quella della luminosità mediata su tutta l’immagine. Abbiamo deciso, quindi, di inserire una fase di *segmentazione* in cui vengono estratte le regioni della scena che la camera deve inquadrare. Nel paragrafo 4.3 vedremo in dettaglio come vengono estratte le regioni dalla scena.

4.2 Algoritmo per il monitoraggio degli indicatori

4.2.1 Monitoraggio one-shot

4.2.2 Monitoraggio sequenziale

4.3 Algoritmo di segmentazione

Configurazione:

1. Estraggo le regioni $\{R_k\}, k = 1, \dots, K$
2. **for** $t = 1, \dots, T_{training}$ **do**
3. Estraggo il frame z_t
4. **for** $k = 1, \dots, K$ **do**
5. | Calcolo $g_t^k, \Delta g_t^k$ per la regione R_k
6. | **end**
7. | **end**
8. **for** $k = 1, \dots, K$ **do**
9. | Definisco le soglie Th_{min}^k e Th_{max}^k
10. | **end**

Fase operativa:

11. **for** $t = T, \dots, \infty$ **do**
12. | Estraggo il frame z_t
13. | $n = 0$
14. | **for** $k = 1, \dots, K$ **do**
15. | Calcolo $g_t^k, \Delta g_t^k$ per la regione R_k
16. | **if** $\Delta g_t^k < Th_{min}^k \vee \Delta g_t^k > Th_{max}^k$ **then**
17. | | $n = n + 1$
18. | | **end**
19. | **end**
20. | **if** $n \geq K - 1$ **then**
21. | | z_t è un frame in cui è avvenuto uno spostamento della camera
22. | | **end**
23. | **end**

Algorithm 1: Algoritmo di identificazione di spostamenti della camera

Capitolo 5

Realizzazioni sperimentali e valutazione

In questo capitolo illustriamo come sono stati condotti gli esperimenti per la valutazione del nostro algoritmo di tampering detection. La prima parte è dedicata alla descrizione dei dataset utilizzati come riferimento, mostrando quali sistemi di acquisizione sono stati utilizzati e quali sono le metriche che abbiamo tenuto in considerazione. Nella seconda parte, invece, entriamo nel dettaglio su quali sono stati i risultati a valle di tutta l'analisi sperimentale.

5.1 Acquisizione dei dataset

Durante lo svolgimento della tesi abbiamo realizzato alcuni sistemi di acquisizione, in modo da avere un insieme di dataset (*benchmark*) molto ampio che comprendesse sequenze video con diverse qualità dei frame e con diversi framerate. Alcune di queste sequenze sono state utilizzate per testare l'algoritmo di segmentazione, mentre altre hanno verificato l'algoritmo di tampering detection. In generale, per ogni scena ripresa (dove per scena intendiamo una specifica inquadratura che *non cambia* per le varie sequenze) abbiamo:

- almeno una sequenza in cui non avviene nessun evento di tampering, che viene utilizzata per la creazione della mappa;
- almeno una sequenza in cui la camera subisce una sfocatura;
- almeno una sequenza in cui la camera subisce uno spostamento.



Figura 5.1: Il Raspberry Pi utilizzato per l'acquisizione

5.1.1 Sistemi di acquisizione utilizzati

Camere ST

Raspberry Pi Camera

Il sistema di acquisizione realizzato con le camere di ST aveva il problema di dover essere alimentato dalla rete elettrica. Non è stato possibile, quindi, utilizzarlo per acquisizioni all'esterno.

Per ovviare a questo problema abbiamo deciso di realizzare un altro sistema basato su un *Raspberry Pi modello B+* [24] con relativo *modulo camera* [25]. Il Raspberry Pi (Fig. 5.1) è un *single-board computer* (ovvero un computer implementato su una singola scheda elettronica) basato su un *system-on-chip* (SoC) *Broadcom BCM2835* [26], che incorpora un processore *ARM1176JZF-S* [27], una GPU *VideoCore IV* [28] e 512 MB di memoria. Utilizza un sistema operativo *Debian Linux* realizzato per processori ARM chiamato *Raspbian* [29]. Le ridotte dimensioni e il basso consumo di potenza hanno permesso l'utilizzo del Raspberry Pi per fare le acquisizioni in ambienti esterni, utilizzando una batteria per alimentare il tutto. Per interfacciarsi con il dispositivo abbiamo creato una *connessione SSH*, tramite USB, con uno smartphone: in questo modo abbiamo potuto lanciare i comandi necessari per far partire l'acquisizione. Per l'acquisizione abbiamo creato uno script in *Python*, utilizzando la libreria *picamera* [30] per interfacciarsi con il modulo camera.

Estrazione frame dal web

Un ultimo dataset è stato generato utilizzando sequenze di frame presenti sul portale di previsioni del tempo *ilMeteo.it* [31],

5.1.2 Sequenze con presenza di tampering

5.1.3 Definizione dei ground thruth

5.1.4 Definizione delle metriche per la stima delle prestazioni

5.2 Risultati ottenuti

Capitolo 6

Direzioni future di ricerca e conclusioni

Bibliografia

- [1] Antonio Manetti, Domenico De Robertis, and Giuliano Tanturli. *Vita di Filippo Brunelleschi: preceduta da La novella del Grasso*, volume 2. Milano: Il Polifilo, 1976.
- [2] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [3] David A Forsyth and Jean Ponce. Computer vision: A modern approach. 2002.
- [4] <http://www.mitan.it/security-solution/videosorveglianza/sistemi-di-videosorveglianza-e-registrazione/>. Visitato il giorno 09/03/2015.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [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] 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.
 - [11] Damian Ellwart, Piotr Szczuko, and Andrzej Czyżewski. Camera sabotage detection for surveillance systems. In *Security and Intelligent Information Systems*, pages 45–53. Springer, 2012.
 - [12] Nuno Roma, José Santos-Victor, and José Tomé. A comparative analysis of cross-correlation matching algorithms using a pyramidal resolution approach. 2002.
 - [13] 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.
 - [14] 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.
 - [15] Ronald N Bracewell. The fourier transform and its applications. 1978.
 - [16] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. 1968.
 - [17] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.
 - [18] 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.
 - [19] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.
 - [20] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

- [21] 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.
- [22] 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.
- [23] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51(4):921–960, 2007.
- [24] <http://www.raspberrypi.org/products/model-b-plus/>.
- [25] <http://www.raspberrypi.org/products/camera-module/>.
- [26] <http://www.broadcom.com/products/BCM2835/>.
- [27] <http://www.arm.com/products/processors/classic/arm11/arm1176.php/>.
- [28] http://www.broadcom.com/products/technology/mobmm_videocore.php/.
- [29] <http://www.raspbian.org/>.
- [30] <http://picamera.readthedocs.org/en/release-1.9/>.
- [31] <http://www.ilmeteo.it/webcam/>.