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

Elenco delle figure

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 presentiamo i concetti fondamentali della teoria alla base della *visione artificiale* e lo studio fatto finora sul problema del *tampering detection*. Nel primo paragrafo diamo una visione generale del modello della camera, senza pretendere una completa e dettagliata descrizione, bensì una visione più generale per comprendere meglio gli argomenti trattati nei capitoli successivi. Nel secondo paragrafo 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*). 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

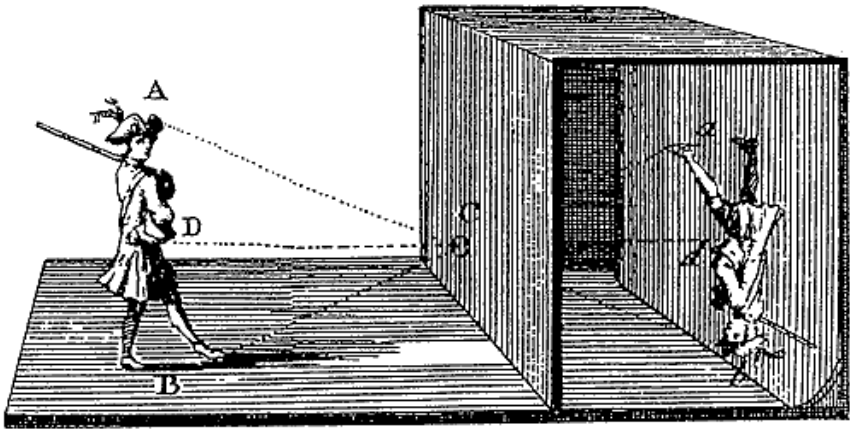


Figura 2.1: Athanasius Kircher, Principio della prospettiva nella camera obscura, 1671

2.1, mentre nei moderni sistemi di acquisizione viene realizzato tramite *lenti ottiche*.

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 $\mathbf{M} = [X, Y, Z]^T$ un qualsiasi punto nello spazio tridimensionale e con $\mathbf{m} = [u, v]^T$ 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* $\mathbf{c}$. 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.

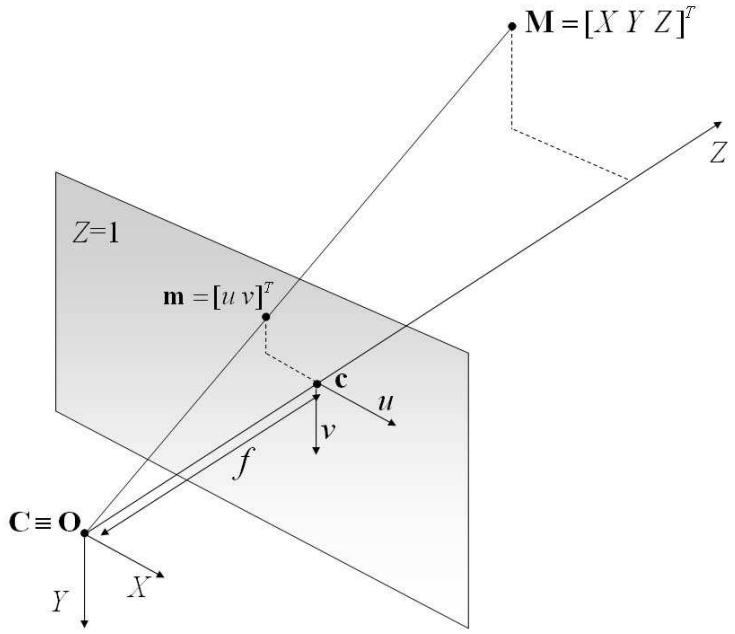


Figura 2.2: Schematizzazione del sistema ottico della camera

Consideriamo un punto **M** nello spazio tridimensionale, rappresentato tramite le coordinate omogenee $[X, Y, Z, 1]^T$, e il suo punto immagine **m** rappresentato dalle coordinate omogenee $[u, v, 1]^T$. La proiezione della camera può essere espressa come

$$\mathbf{m} = P\mathbf{M}, \tag{2.1}$$

dove P è una matrice di dimensioni 3×4 chiamata *matrice prospettica* [1]. 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 viene rappresentata attraverso due matrici: una che rappresenta i *parametri intrinseci* e un'altra che rappresenta i *parametri estrinseci* [2].

- I *parametri intrinseci* rappresentano le caratteristiche interne della camera come la distanza focale, il centro focale e le caratterisitiche di *distorsione* della lente.

- I *parametri estrinseci* rappresentano la posizione della camera rispetto al sistema di riferimento dello spazio tridimensionale.

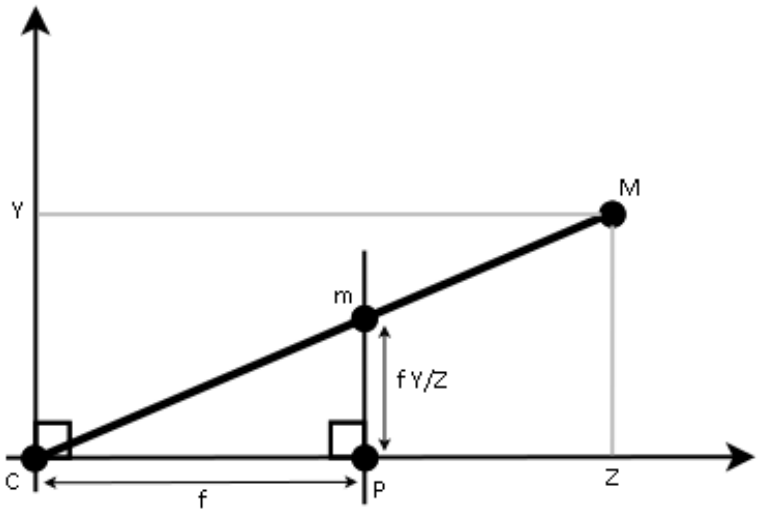


Figura 2.3: Similitudini tra triangoli usate nel calcolo dei parametri intrinseci

2.1.2 Parametri intrinseci

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

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

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

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

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

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

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = K \begin{bmatrix} I & | & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.3}$$

dove la matrice

$$K = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \tag{2.4}$$

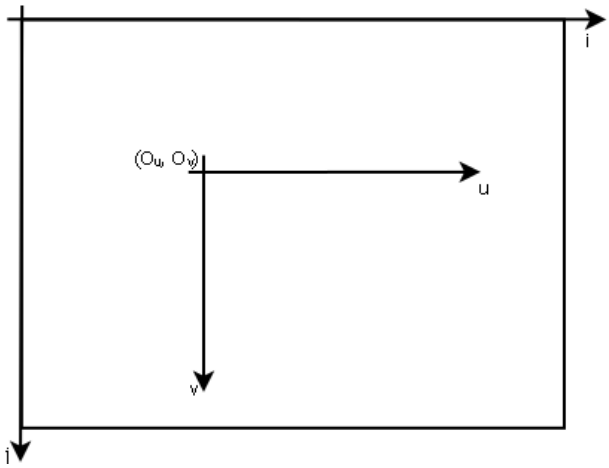


Figura 2.4: Coordinate di un frame

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

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

mentre la matrice di calibrazione(2.4) diventa

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

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

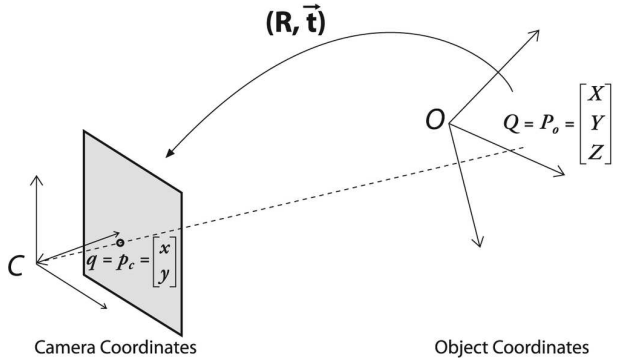


Figura 2.5: Trasformazione dal sistema di coordinate dell'ambiente a quello della camera

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

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

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 camere localizzate in una posizione arbitraria, 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

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

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

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

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 Tampering Detection

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 [3]. 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 nel tempo;
- tutti gli elementi della scena sono presenti nell'immagine in maniera *nitida*.

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.

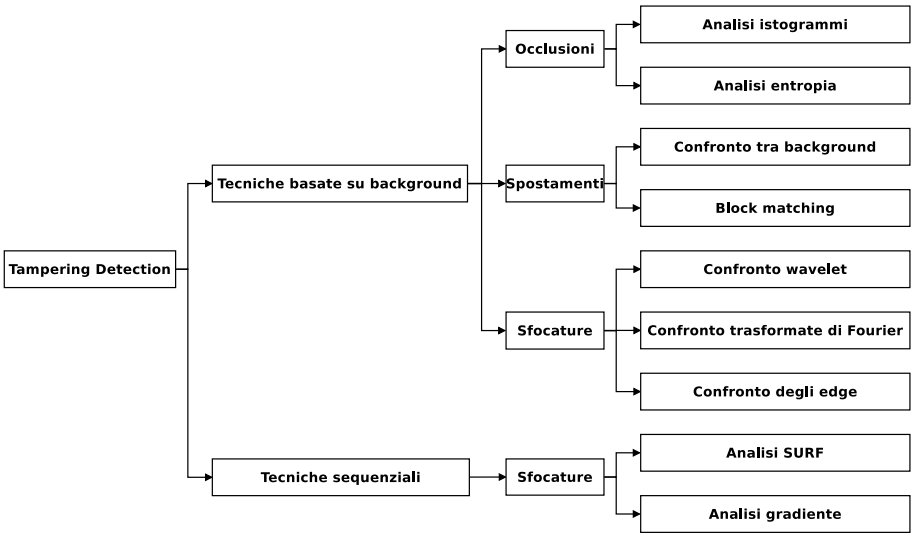


Figura 2.6: Tecniche di tampering detection

Per affrontare il problema dell'identificazione di questo tipo di eventi, 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. Lo schema in figura 2.6 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.

2.2.1 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 i viene definita *immagine* o *frame* i-esimo. La figura 2.7 illustra questi concetti.
Per semplificare l'analisi, 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.

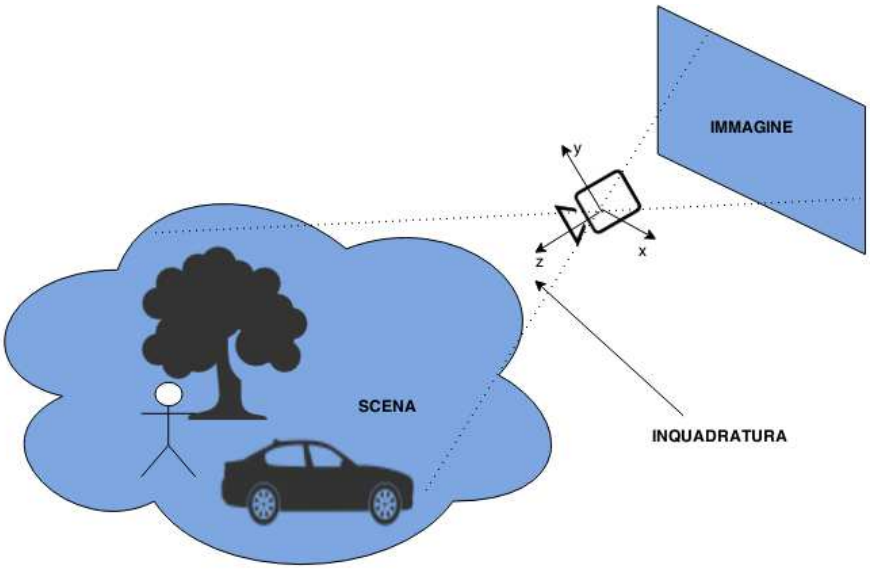


Figura 2.7: Sistema di monitoraggio video

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

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

2.2.2 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 [4]. Il valore del modello di background per il pixel x 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:

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

dove $c > 1$ e $0 < a < 1$. Lo stesso modello viene utilizzato in altri lavori, come ad esempio in [5] e [6], mentre una variante molto usata consiste nel calcolare il modello di background a partire dall'*estrazione dei contorni* di ciascun frame ([7], [8]). Un approccio diverso, ma comunque riconducibile allo stesso filone, lo troviamo in [9], 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 [4] e [5] questo evento è associato a un cambiamento nella struttura dell'*istogramma* del frame occluso rispetto a quello del background. In particolare, se z_n 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_n, 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:

$$\begin{aligned} &\left(H_{max(H(z_i))-1}(z_i) + H_{max(H(z_i))}(z_i) + H_{max(H(z_i))+1}(z_i)\right) \\ &> Th_1 \left(H_{max(H(z_i))-1}(B_i) + H_{max(H(z_i))}(B_i) + H_{max(H(z_i))+1}(B_i)\right), \end{aligned}$$

$$\sum_{b=1}^{32} H_b\left(|z_i - B_i|\right) > Th_2 \sum_{b=1}^{k} H_b\left(|z_i - B_i|\right),$$

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. In particolare, Th_1 e Th_2 possono essere aumentate in modo da aumentare la sensibilità, mentre k può essere diminuito.
Un approccio simile è presente in [7], [8] e [10], 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 ([8]) 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 [5] 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:

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

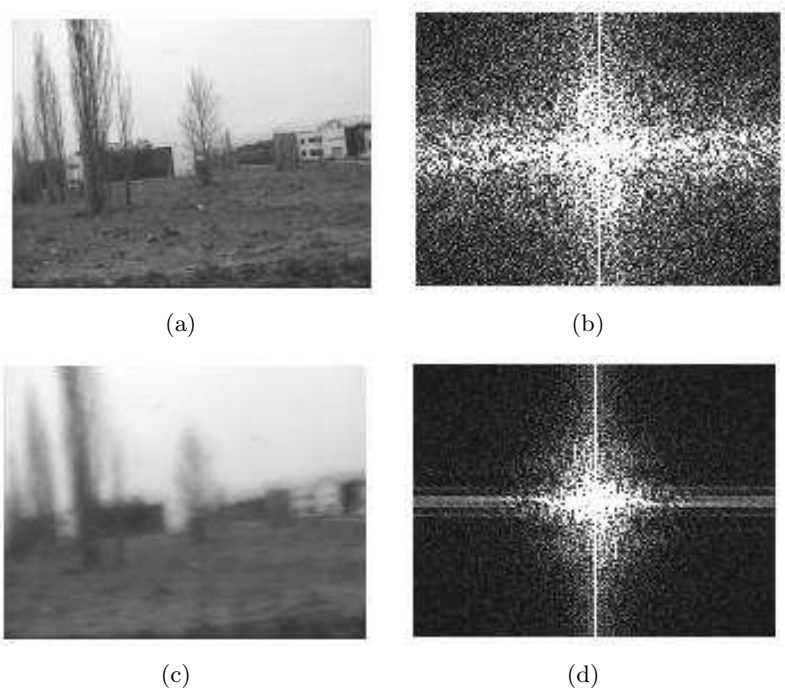
Lo spostamento della camera viene identificato quando $P > ThK$, dove $0 < Th < 1$ è un valore di soglia scelto in base alla sensibilità che si vuole dare all'algoritmo e K è il numero totale di pixel dell'immagine.
Un altro approccio è quello di utilizzare tecniche di *block matching*. In [8] e [7] il confronto viene fatto utilizzando la formula *zero-mean normalized cross correlation* (ZNCC) [11]:

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

dove μ_{z_i} e σ_{z_i} rappresentano la media e la deviazione standard della luminosità nell'immagine z_i. Un'altra cifra di merito, utilizzata in [10], 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 [12], con la differenza che, invece di analizzare la correlazione dei pixel, viene analizzata quella degli istogrammi.



(a) (b)

(c) (d)

Figura 2.8: 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 [4] e [5] questo fenomeno è associato a una diminuzione dell'energia ad alta frequenza. Per analizzare questo cambiamento, [4] confronta ciascun frame con il background nel dominio delle *wavelet* [13], mentre [5] utilizza il dominio della *trasformata di Fourier* [14]. Nella figura 2.8 vediamo un esempio di come si comporta la trasformata di Fourier nel caso di sfocature: nel passaggio da un frame nitido (Fig. 2.8(a)) a uno sfocato (Fig. 2.8(c)) abbiamo un crollo delle componenti ad alta frequenza nelle trasformate di Fourier (rispettivamente Fig. 2.8(b) e Fig. 2.8(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 [10], [8], [7] e [12], consiste nell'estrarre i contorni dalle

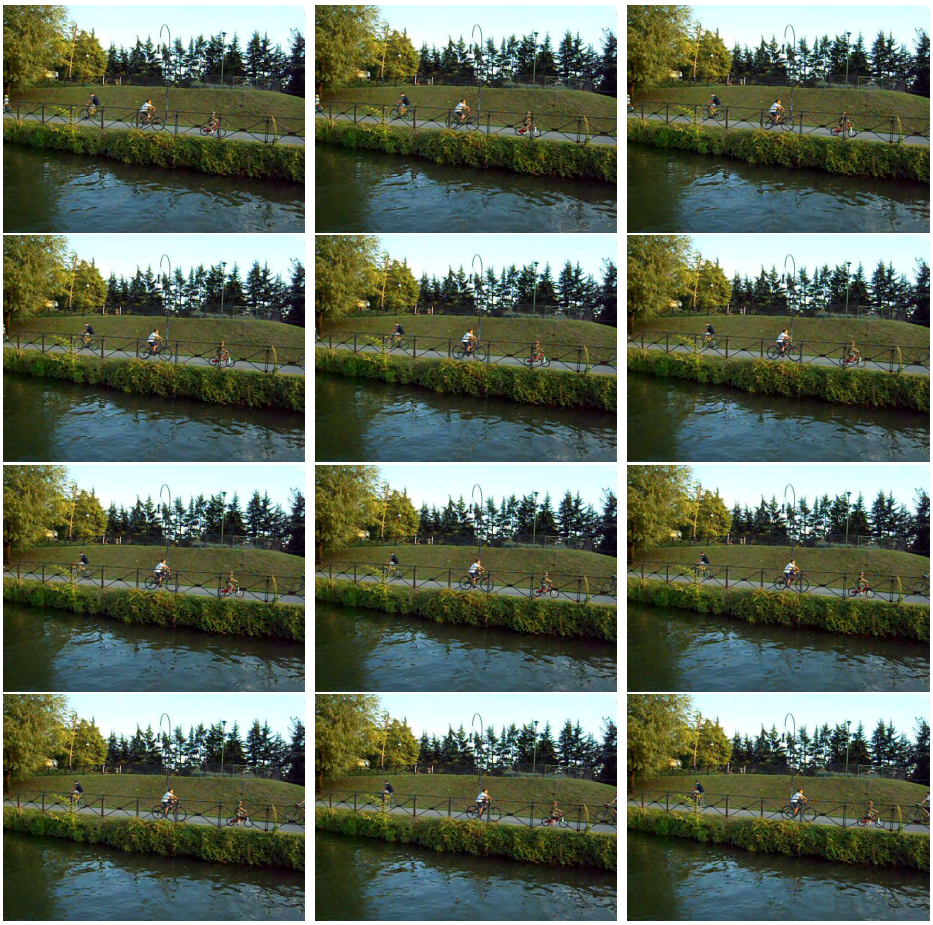


Figura 2.9: Sequenza di frame acquisiti a 30 fps

immagini secondo il metodo di Sobel [15] o di Canny [16], 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.

2.2.3 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

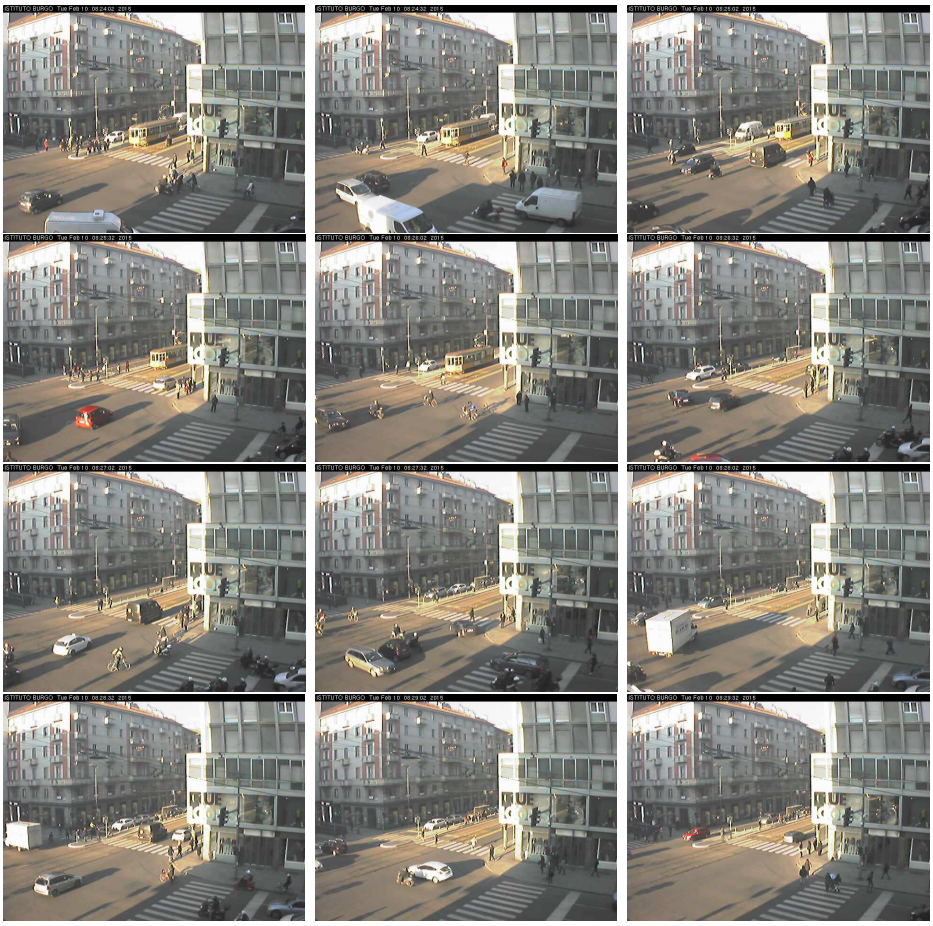


Figura 2.10: Sequenza di frame acquisiti ogni 30 secondi

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 2.9 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 2.10 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, che le tecniche viste finora richiedono che il sistema di monitoraggio possegga 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) [17] o *change-detection test* [18]. Troviamo alcuni esempi nell'identificazione di sfocature: in [6] la soluzione consiste nel monitorare nel tempo il *numero di SURF* [19], 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 [20], dove le sfocature vengono identificate monitorando l'*energia media del gradiente* delle immagini acquisite:

$$m_i = \mathcal{M}[z_i] = \int_{\mathcal{X}} \| \bigtriangledown z_i(x) \|_1 dx, \tag{2.9}$$

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) [21].
Il test statistico utilizzato non richiede alcuna informazione *a priori* del processo che si sta monitorando, e sfrutta una sequenza iniziale $\{m_i\}, i = 1, \ldots, T$ di indicatori estratti da frame non sfocati, in modo da configurare automaticamente i suoi parametri. Tale sequenza prende il nome di *training set*, e permette di stimare la *densità probabilistica* (*probability density function* (pdf)) di m_i in assenza di sfocature, (i.e. 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, [20] consiglia di utilizzare un training set ampio, ad esempio $T > 400$. Il test opera su *sotto-sequenze* di misure m_i (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_i 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_i 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

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

supera una certa soglia h.
L'utilizzo di una tecnica sequenziale per l'identificazione di sfocature 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) [22]. D'altro canto, confrontato con le tecniche descritte nel paragrafo 2.2.2, questo metodo 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 [3], 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 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

(a) (b)

Figura 3.1: Esempi di sfocature

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 la trattazione presente in [20], 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 η 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*,

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

Figura 3.2: Sequenza di otto frame consecutivi acquisiti ogni minuto

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 $h(\cdot)$ 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 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 ($\mathcal{B}_i \neq \mathcal{B}_{i+1}$).

3.1.2 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

[1] Il 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

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 $\{y_i\}$ 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 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 (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.

3.1.3 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,

(a) (b)

Figura 3.4: Esempi di occlusione

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

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 $\mathcal{S}$.
Consideriamo il caso generale di una sequenza di frame $\{z_i\}, i = 1, \ldots, N$. Per $i < T^*$ abbiamo che i frame vengono acquisiti in condizioni di funzionamento normale; all'istante di tempo $i = T^*$ avviene un evento di tampering, il quale compromette i frame per $T^* < i \leq N$. Nel caso dell'operatore di degradazione $\mathcal{D}$, con riferimento al modello descritto nell'equazione (3.5), pos-

(a) (b)

Figura 3.5: Esempi di guasti

siamo considerare che, in condizioni di funzionamento normale, l'operatore di sfocatura $\mathcal{B}$ si comporti come un *filtro passa-tutto*, ovvero

$$\mathcal{B}_i[y](x) = y_i(x), \text{ per } i = 1, \ldots, T^*,$$

e che all'istante T^* esso cambi, diventando come nel caso dell'equazione (3.3). L'obiettivo dell'algoritmo è quello di stimare l'istante T^* di cambiamento dell'operatore di sfocatura.
Nel caso dell'operatore di spostamento $\mathcal{S}$, tenendo come riferimento la formula (3.6), l'obiettivo dell'algoritmo è quello di stimare l'istante di tempo T^* in cui avviene il passaggio dalla sequenza di immagini $\{y_i\}$ a $\{w_i\}$.
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 è, quindi, possibile utilizzare le tecniche basate su background che sono state descritte nel paragrafo 2.2.2. Nel prossimo capitolo vedremo la soluzione che abbiamo deciso di utilizzare per lo sviluppo dell'algoritmo di tampering detection.

Capitolo 4

Soluzione proposta

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.

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+* [23] con relativo *modulo camera* [24]. Il Raspberry Pi è un *single-board computer* (ovvero un computer implementato su una singola scheda elettronica) basato su un *system-on-chip* (SoC) *Broadcom BCM2835* [25], che incorpora un processore *ARM1176JZF-S* [26], una GPU *VideoCore IV* [27] e 512 MB di memoria. Utilizza un sistema operativo *Debian Linux* realizzato per processori ARM chiamato *Raspbian* [28]. Le ridotte dimensioni e il basso consumo di potenza hanno permesso di utilizzarlo per fare delle acquisizioni in ambienti esterni, utilizzando una batteria.

Estrazione frame dal web

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] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] David A Forsyth and Jean Ponce. Computer vision: A modern approach. 2002.

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

[4] 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.

[5] 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.

[6] 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.

[7] Sebastien Harasse, Laurent Bonnaud, Alice Caplier, and Michel Desvignes. Automated camera dysfunctions detection. In *Image Analysis and Interpretation, 2004. 6th IEEE Southwest Symposium on*, pages 36–40. IEEE, 2004.

[8] 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.

[9] Evan Ribnick, Stefan Atev, Osama Masoud, Nikolaos Papanikolopoulos, and Richard Voyles. Real-time detection of camera tampe-

ring. In *Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on*, pages 10–10. IEEE, 2006.

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

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

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

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

[20] 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.

[21] 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.

[22] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51(4):921–960, 2007.

[23] `http://www.raspberrypi.org/products/model-b-plus/`.

[24] `http://www.raspberrypi.org/products/camera-module/`.

[25] `http://www.broadcom.com/products/BCM2835/`.

[26] `http://www.arm.com/products/processors/classic/arm11/arm1176.php/`.

[27] `http://www.broadcom.com/products/technology/mobmm_videocore.php/`.

[28] `http://www.raspbian.org/`.