5 VIDEOSORVEGLIANZA

5.1 Introduzione

Il percorso sin qui compiuto da questa ricerca ha contribuito alla maturazione di un'esigenza ora divenuta improcrastinabile: fuggire il più possibile i vincoli dettati dal contesto applicativo e porsi in un'ottica più generale, tale da poter comprendere l'essenza degli attori presenti in una scena, indipendentemente dalla loro natura e dal loro stato di quiete o moto. Questa esigenza trova asilo naturale nel cosiddetto contesto di ricerca sulla videosorveglianza. Il collegamento che mi ha portato a questo tema più ampio passa attraverso il tentativo di automatizzare gli impianti semaforici pedonali in base alla presenza dei pedoni ed al problema più fine di comprendere se questi siano casualmente presenti sulle isole di attesa, o se realmente intendano compiere l'attraversamento. Altro obiettivo per questi sistemi risulta essere il conteggio dei pedoni in attesa, per stabilire un peso da attribuire alle fasi semaforiche. Studi preliminari su questo argomento [MOR94] [RIC95] [ORE97] mettono in evidenza una moltitudine di scenari differenti per ciascuno dei quali esistono numerosi approcci possibili.

Tuttavia, problemi di questo tipo sono parimenti catalogabili come problemi specifici di videosorveglianza in cui sono presenti due classi di attori (persone e veicoli) a cui applicare tassonomie. La realizzazione di sistemi efficaci per la rivelazione dei pedoni passa attraverso la messa a fuoco di un obiettivo preciso e circoscritto [WRE97] [REA98]. Al fine di raggiungere una'ampia conoscenza sul più generale (la videosorveglianza) si è scelto di approfondire l'indagine sulle applicazioni che, a differenza di quelle citate, non

avessero un delimitato obiettivo ingegneristico ma affrontassero piuttosto problemi specifici in contesti indefiniti. A questo punto è necessaria una puntualizzazione: per i sistemi di videosorveglianza presenti sul mercato [COM00] [PRE00] [SHA00] esiste una chiara definizione, comune a produttori e fornitori, che proviene dal tradizionale tipo di applicazione con telecamere a circuito chiuso; tale concetto non deve però essere applicato direttamente alla definizione del contesto di ricerca che attualmente sviluppa temi di videosorveglianza in quanto i sistemi commerciali basati sul vecchio concetto di videosorveglianza presumono il controllo delle immagini da parte di un operatore umano che le deve interpretare, mentre l'interpretazione è l'obiettivo principale della ricerca in questo settore. Si parla quindi di due argomenti diversi sebbene questi abbiano il medesimo nome. A mio giudizio, l'unico punto di congiunzione sta in quei sistemi di videosorveglianza basati su personal computer che incorporano algoritmi di change detection in grado di avviare una videoregistrazione o di allertare un operatore. La sfida più prossima che si pone alla ricerca è quella di risolvere problemi specifici in contesti vincolati il meno possibile alla natura di oggetti di interesse ed ambiente. Manca ancora infatti un sistema di videosorveglianza automatico in grado di soddisfare questa esigenza, così come, a maggior ragione, non esiste un sistema di videosorveglianza che possa adattarsi ad un qualsiasi ambiente e che sia completo di algoritmi di rivelazione in grado di interpretare molteplici situazioni. Così si è preferito approfondire gli studi sulla parte di basso livello che è presente in ciascun sistema videosorveglianza basato su Computer Vision, allo scopo di indagare se sia possibile svilupparne uno indipendente dai vincoli sopra elencati. In questo senso un importante aiuto è stato fornito da chi [TOY99] ha classificato gli scenari più tipici per la videosorveglianza elencando le più comuni cause di rumore che ne possono pregiudicare il funzionamento. Altri spunti provengono da chi [GAV99] ha riassunto lo stato dell'arte della ricerca mettendo in chiara evidenza alcuni risultati poco soddisfacenti di progetti di sistemi di videosorveglianza "general purpose". Altre letture ([AMI98] [UTS98] [RIG99] [STA99] [BRA00] ed in particolare [HAR98] [HAR99] [HA299]) sono servite a

formulare chiare ipotesi sulle funzionalità dei moduli che costituiscono la struttura fondamentale dei sistemi di videosorveglianza.

5.2 Struttura modulare di un sistema per videosorveglianza (VSS)

La struttura di un generico VSS può essere descritta in tre [TOY99] livelli principali (Figura 5-1):

- Pixel Processing Level
- Frame Processing Level
- Tracking Level

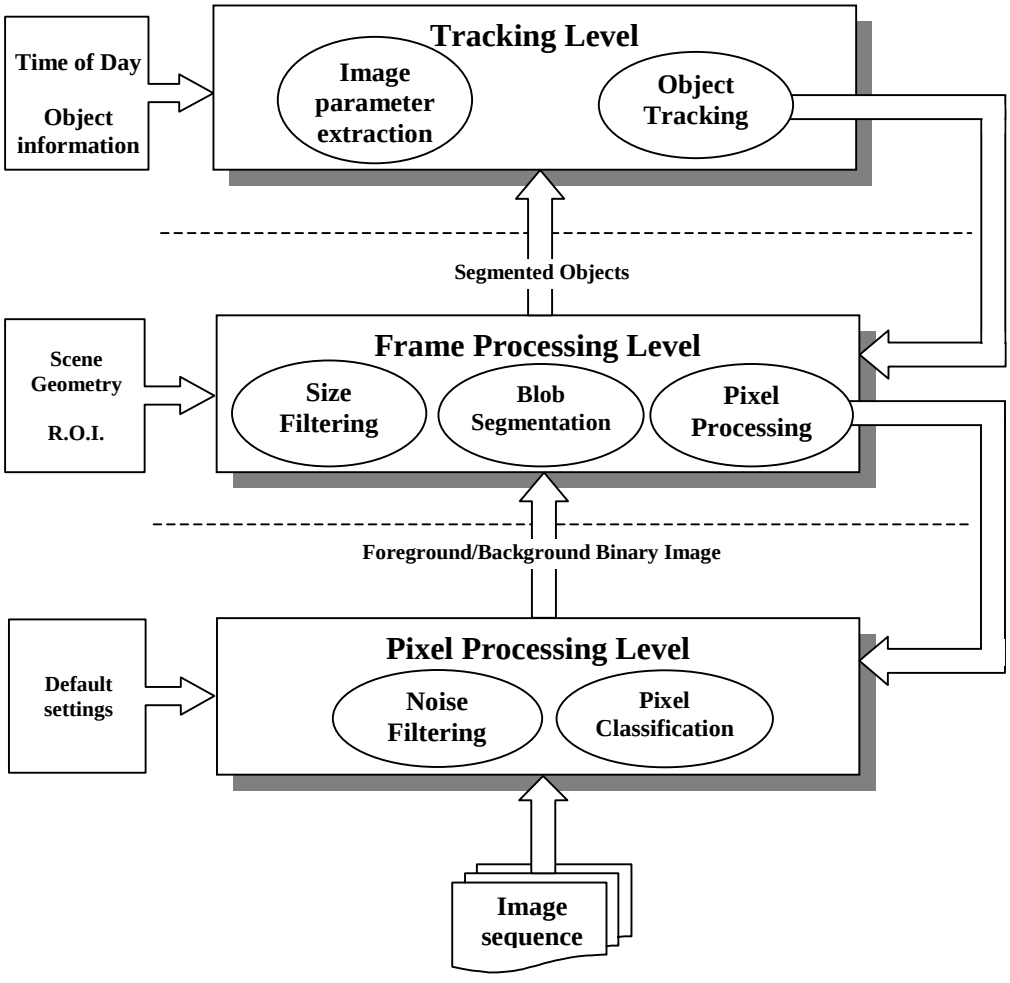

Figura 5-1 Livelli e moduli di un sistema di videosorveglianza.

Il livello “Pixel processing” contiene gli algoritmi di base che definiscono l’appartenenza dei pixel allo sfondo o ad un oggetto (stima del background). L'informazione prodotta a questo livello è di tipo binario: vengono ignorate la natura e il numero degli oggetti presenti e qualsiasi fenomeno che produca effetti non contemplati tra le possibili fonti di rumore eliminabili dall’algoritmo dà luogo a pixel oggetto. I parametri impostati a questo livello sono modificabili dai livelli superiori via via che il sistema perfeziona la propria conoscenza della scena.
Il livello di elaborazione “Frame processing” si occupa di una prima interpretazione dell’immagine binaria fornita dal livello sottostante. Si considerano le relazioni che intercorrono tra i gruppi di pixel individuati come oggetto e che possono aiutare a raffinare la classificazione fatta dagli algoritmi di base, si filtrano gruppi di pixel di piccoli dimensioni (size filtering) e si attua la classificazione dei pixel secondo l’appartenenza ad oggetti diversi (blob segmentation). A questo livello l’elaborazione è condizionata dalla geometria della scena e da informazioni supplementari come la definizione di aree di interesse (ROI Region of Interest).
Il livello più alto “Tracking”, si occupa della classificazione, del riconoscimento e dell’inseguimento degli oggetti individuati dal livello inferiore. Qui si utilizzano informazioni riguardanti il tipo di oggetto e le caratteristiche generali della scena ripresa (ora del giorno, condizioni di visibilità e metereologiche,…). Le informazioni riguardanti le posizioni precedenti degli oggetti possono essere usate al Frame Processing Level per migliorare la qualità della segmentazione.

5.3 Fattori critici per un VSS

Vi sono fenomeni che possono dare origine a tipi di rumore dannoso per i vari livelli di un sistema VSS. Ci occuperemo del rumore dannoso al livello Pixel Processing e cioè di quello che porta ad una errata classificazione dei pixel tra oggetto e sfondo: nel caso un pixel sia individuato erroneamente come appartenente ad un oggetto di parla di “falso positivo”, mentre se viene

erroneamente classificato come background l'errore commesso prende il nome di "falso negativo". Secondo la classificazione proposta in [TOY99], i principali fenomeni possono essere classificati come:

- Waveing trees
- Mimetizzazione
- Light change
- Foreground aperture

Con "waveing trees" si intendono quei fenomeni di moto che non hanno interesse ai fini della videosorveglianza. Un tipico esempio è rappresentato da foglie di alberi o fili sospesi mossi dal vento, oppure dallo "sfarfallio" dei monitor ripresi da una telecamera generato dalle due diverse frequenze di scansione. La "mimetizzazione" è quel fenomeno per il quale un oggetto ha caratteristiche cromatiche tali da essere confuso con lo sfondo causando così falsi negativi. I "light changes" indicano tutte le variazioni dovute a cambi di illuminazione della scena. Con "foreground aperture" si intende quel tipo di rumore che rende invisibile la parte di immagine comune a due frame dello stesso oggetto in movimento in una elaborazione derivativa (dato che tutte le analisi di questa ricerca sono state svolte su immagini a toni di grigio, intendiamo in realtà con "colore" l'intensità della scala di grigi associata ai pixel delle immagini).

Il livello "Frame Processing" si occupa di queste incombenze:

- Distinzione di oggetti che si occludono a vicenda
- Riconoscimento e filtraggio delle ombre

Il livello "tracking" di un SVS realizza l'inseguimento degli oggetti e risolve di norma alcuni eventi tipici:

- Moved object
- Sleeping person
- Walking person

"Moved object" sono quegli oggetti normalmente in quiete ed appartenenti al background che possono animarsi, ma che non sono di interesse. Al contrario "sleeping person" è un oggetto che entra nella scena e poi si ferma: in questi casi la decisione se continuare a considerare l'oggetto per un tempo prolungato o annetterlo allo sfondo spetta ai livelli superiori. Il termine "walking person"

identifica quella categoria di oggetti inizialmente appartenenti allo sfondo (ad es. una macchina parcheggiata) che ad un dato istante si anima.
Nel corso di questa ricerca ci occuperemo strettamente dello sviluppo di algoritmi per il “pixel processing level”. Molte delle elaborazioni sono state eseguite con codice scritto in C++ su un personal computer dotato di un frame grabber per acquisire un segnale video nello standard PAL. Il codice fa parte di un applicativo “Sentinel” in grado di digitalizzare sequenze video e registrarle su disco rigido o di processare on-line le immagini provenienti dal frame grabber. Le sequenze campionate con il programma Sentinel sono state acquisite in modalità stereo da una coppia di telecamere i cui segnali sono multiplexati da harware dedicato, costruito appositamente nel Laboratorio di Visione Artificiale della Facoltà di Ingegneria Elettronica a Bologna. La scelta del formato video stereo nasce dall’intenzione futura di utilizzare le sequenze algoritmi che usino entrambi i campi per beneficiare anche dell’informazione sulla distanza. Nelle misure di cui sarà dato conto sarà utilizzato un solo campo dei due disponibili.

5.4 Classificazione degli algoritmi

Le numerose letture affrontate in fase preliminare hanno suggerito la classificazione rappresentata in Figura 5-2.

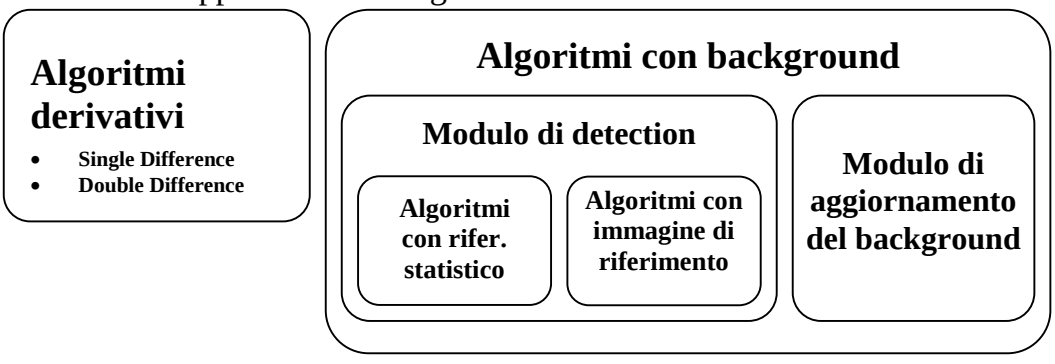

Figura 5-2 Classificazione degli algoritmi “pixel processing level”

In essa si notano due classi principali distinte dalla presenza di un riferimento (background). Nell’applicativo “Sentinel” sono stati implementati sia algoritmi che non ne fanno uso (“single” e “double difference”) sia altri che invece

utilizzano il riferimento. Tra gli ultimi è opportuna un'ulteriore distinzione tra i diversi tipi di moduli di "detection" e di "aggiornamento del background" suggerita grazie all'esperienza maturata già alla prima versione del "VIL" nella quale il background è aggiornato solo se non è rivelata la presenza di alcun veicolo sulla spira (Figura 3-2). Tale soluzione è risultata di primaria importanza per lo sviluppo di alcuni algoritmi innovativi facenti uso del background.

5.5 Algoritmi con differenze tra frame (derivativi)

Caratteristica principale di questa classe è quella di individuare gli oggetti in movimento in una sequenza video prendendo in considerazione solo le differenze presenti tra immagini successive. Tali algoritmi sono in grado di evidenziare oggetti in movimento in quanto un oggetto fermo non provoca differenze tra due frame consecutivi. Le possibili implementazioni si differenziano tra loro in base al numero di immagini consecutive considerato nelle elaborazioni. Due soluzioni efficaci "Single Difference" [TOR96] e "Double Difference" [CUC99] considerano rispettivamente le ultime due e le ultime tre immagini della sequenza video.

L'output generato da questi algoritmi è caratterizzato dal fatto di essere dipendente dalla velocità degli oggetti in movimento nella scena. Maggiore è la velocità di spostamento, maggiori sono le differenze tra immagini consecutive e quindi più ricco è l'output dell'algoritmo. Viceversa movimenti lenti introducono piccole variazioni, riducendo la qualità dell'elaborazione finale. Di conseguenza, in questa classe di algoritmi, oggetti che compaiono nella scena e poi si fermano cessano di essere individuati, quindi eventuali operazioni di inseguimento devono essere gestite da moduli di livello logico superiore che devono sempre tenere traccia della posizione dell'oggetto anche nel caso in cui questo si fermi.

Questa classe di algoritmi è sensibile al problema della "foreground aperture": la corretta individuazione della sagoma di un oggetto può essere compromessa se questo presenta zone a tinta uniforme che non producono cambiamenti in immagini successive.

Un pregio notevole di questi algoritmi è rappresentato invece dalla loro sostanziale insensibilità a variazioni luminose dell'ambiente. Ciò vale fintanto che le variazioni delle intensità dei pixel restano al di sotto della soglia di riconoscimento. Fenomeni di discontinuità luminosa (ad es. una lampada che si accende in una stanza) provocano errori molto contenuti e comunque limitati al termine del transitorio (ad es. accensione di una luce al neon).
Entrambi gli algoritmi di seguito descritti presentano pregi legati al carico computazionale ridotto.

5.5.1 "Single difference"

Il funzionamento di "Single difference" si basa sull'analisi delle differenze tra due immagini consecutive di una sequenza (*Immagine*$_{t-1}$ e *Immagine* $_t$). Calcolando il valore assoluto delle differenze tra ogni coppia di pixel delle due immagini si ottiene una nuova matrice di valori della stessa dimensione delle immagini di partenza che viene chiamata *Immagine Differenza Assoluta*:

$$ImmagineDifferenzaAssoluta(x,y) = |immagine_t(x,y) - immagine_{t-1}(x,y)|$$

E' molto probabile che il risultato sia affetto da una certa quantità di pixel "falsi positivi" dovuti a possibili fenomeni di moto limitati ad alcuni elementi dello sfondo (ad es. le foglie di alberi agitate dal vento). Per filtrare tutti i punti rumorosi, si opera esegue una binarizzazione con soglia:

$$ImmagineMovimento(x,y) = \begin{cases} 255, ImmagineDifferenzaAssoluta(x,y) > soglia \\ \\ 0, \text{altrimenti} \end{cases}$$

La matrice binaria così ottenuta prende il nome di *Immagine movimento* (Figura 5-3).

Figura 5-3 Secondo step di elaborazione nella Single Difference.

Un tipico esempio di elaborazione è quello rappresentato dalla Figura 5-4 che contiene un disco in movimento da sinistra a destra.

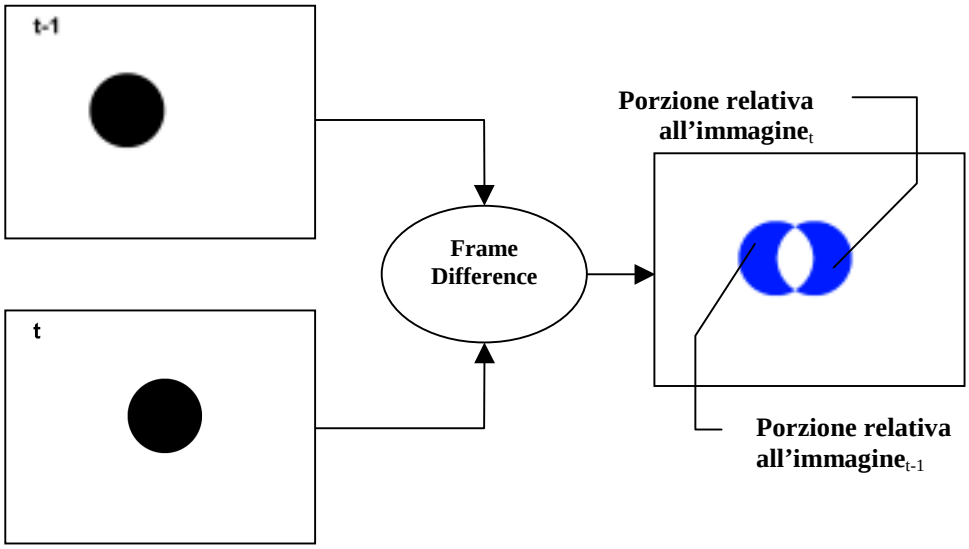

Figura 5-4 Risultato dell'elaborazione Single Difference.

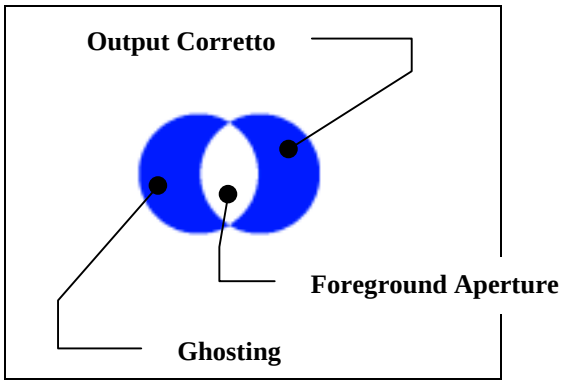

Figura 5-5 Output della Single Difference in presenza di oggetti uniformi.

Il risultato di Figura 5-5 è composto da tre aree: quella indicata con *Ghosting* risulta dalla porzione di sfondo scoperta dal disco in movimento (falsi positivi); i falsi negativi dell'area *Foreground aperture* derivano dalla assenza di variazioni in una zona dell'immagine occupata dall'oggetto a tinta uniforme in entrambi i

frame; vi è poi l'area *Output corretto* che rappresenta parte dell'output ideale (alla quale mancano i "falsi negativi"). I "falsi negativi" della zona *foreground aperture* sono comuni a tutti gli algoritmi derivativi. Tali lacune nascono in quanto non è possibile individuare differenze in questa zona se l'oggetto ha una tinta uniforme mentre il problema scompare se l'oggetto è dotato di texture (Figura 5-6)

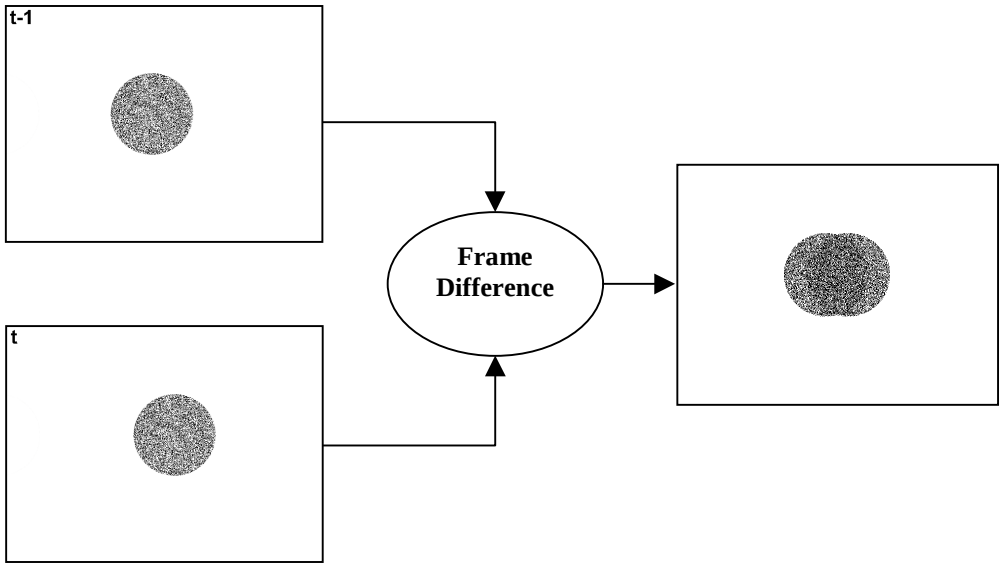

Figura 5-6 Elaborazione di Single Difference in presenza di *textured object*.

In generale, tanto più velocemente si muoverà l'oggetto, tanto maggiore sarà l'effetto del *ghosting* e viceversa.

5.5.2 "Double difference"

L'algoritmo "Double difference" nasce come variante della "Single difference" con l'obiettivo di eliminare l'effetto *ghosting*.

Questo algoritmo ha la peculiarità di prendere in considerazione tre immagini consecutive ($Immagine_{t-2}$, $Immagine_{t-1}$ e $Immagine_{t}$) ed eseguire due volte l'algoritmo Single Difference, prima sulla coppia $Immagine_{t-2}$ e $Immagine_{t-1}$ e poi sulla coppia $Immagine_{t-1}$ e $Immagine_{t}$, dando origine a due immagini binarie *ImmagineMovimento1* e *ImmagineMovimento2* (Figura 5-7).

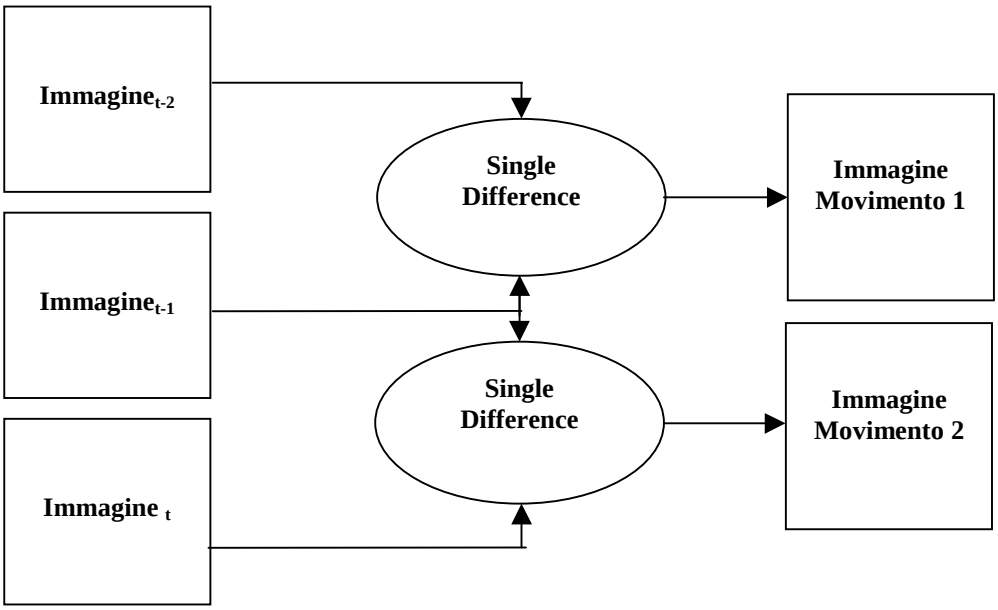

Figura 5-7 Primo passo di elaborazione della Double Difference.

Il secondo passo consiste in una operazione di AND pixel a pixel tra le due immagini binarie (Figura 5-8). Analiticamente questa operazione si può esprimere come segue:

$$ImmagineOutput(x,y) = \begin{cases} 255, ImmagineMovimento1(x,y) \wedge ImmagineMovimento2(x,y) \\ 0, \text{altrimenti} \end{cases}$$

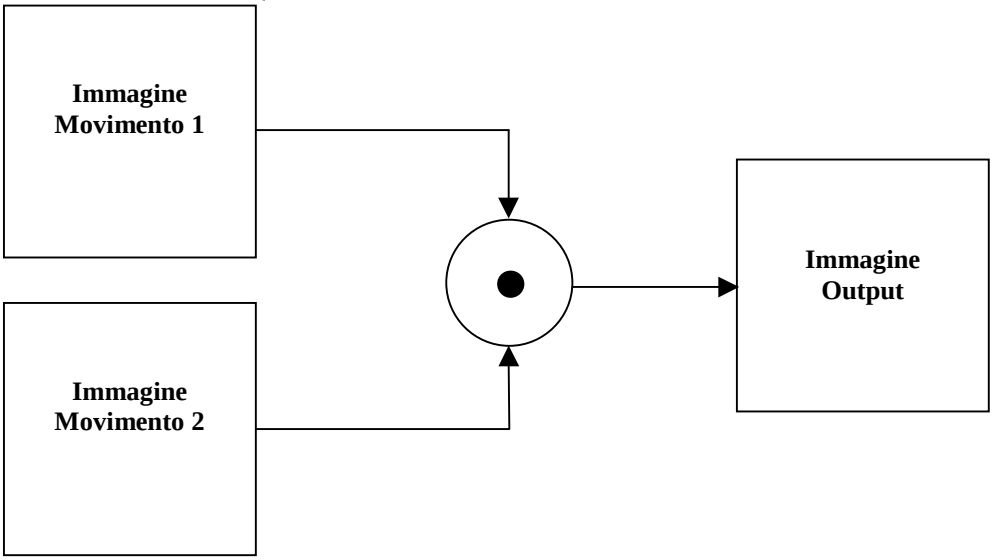

Figura 5-8 Secondo passo di elaborazione della Double Difference.

Il risultato, *ImmagineOutput* è ancora una matrice binaria. Tornando all'esempio del disco in movimento (questa volta su tre frame) si ottiene il risultato di Figura

5-9 in cui l'output si riferisce al frame *Immagine*$_{t-1}$ (e non all'ultima immagine). Rispetto all'algoritmo precedente la Double Difference è maggiormente affetta dal fenomeno *foreground aperture*: se l'oggetto è senza trama viene individuata soltanto la parte tra i contorni dell'oggetto nelle posizioni iniziale e finale, Figura 5-9

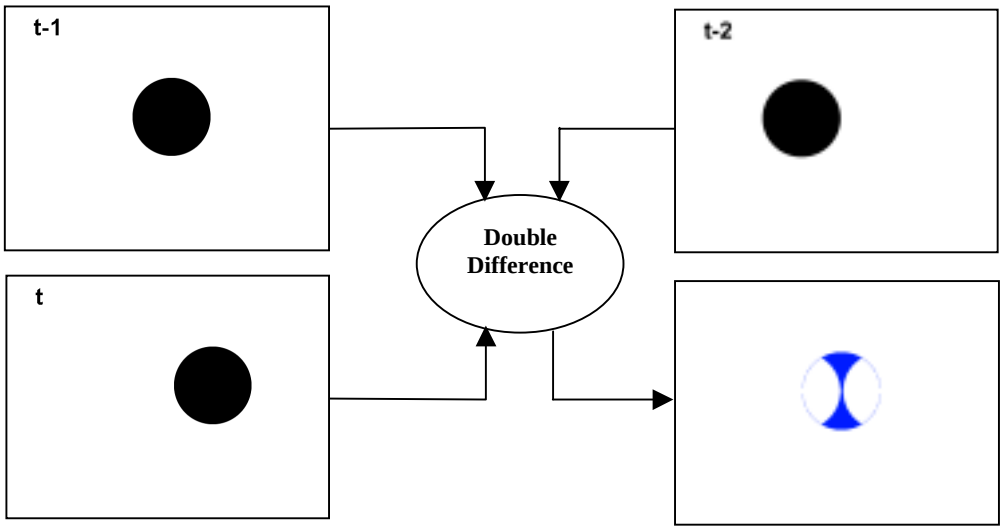

Figura 5-9 Double Difference su disco senza texture

I falsi negativi dovuti a *foreground aperture* scompaiono se si ha a che fare con oggetti dotati di trama (Figura 5-10).

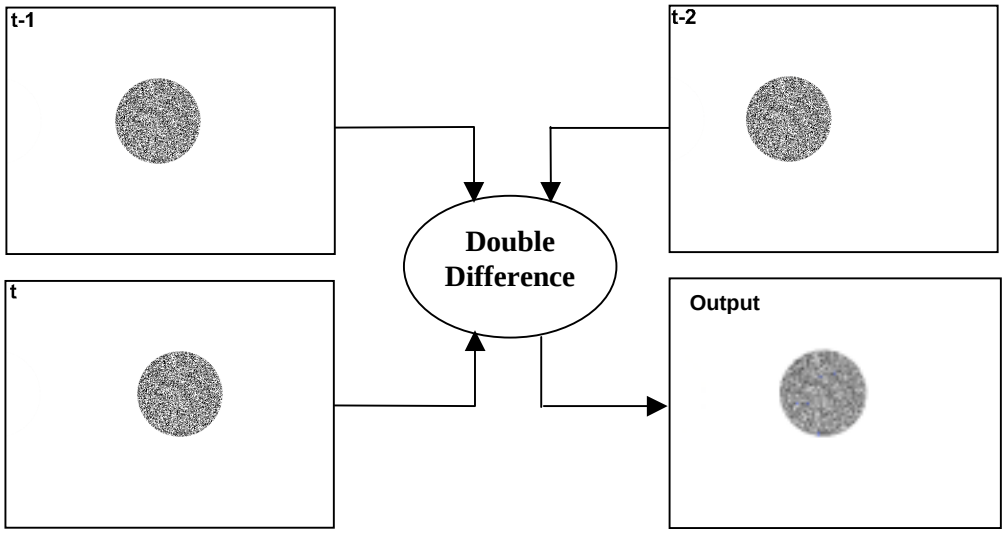

Figura 5-10 Double Difference su disco con trama

In generale la qualità dell'output dipende dalla velocità di movimento degli oggetti: oggetti che si muovono molto rapidamente favoriscono la Double Difference rispetto alla Single Difference in quanto l'assenza di *ghosting* e di sovrapposizioni dell'oggetto tra immagini consecutive (che eliminano la *foreground aperture*) permettono una individuazione corretta. Questa differenza tende a scendere però con il diminuire della velocità di movimento dell'oggetto che evidenzia la maggiore sensibilità al *foreground aperture* della Double Difference.

5.6 Algoritmi con riferimento "background"

In questa classe di algoritmi (si veda la Figura 5-2) l'individuazione degli oggetti si basa sul confronto dell'immagine corrente con un "background" di riferimento contenente informazioni sulla scena in assenza di oggetti di interesse. Questo tipo di approccio risulta più intuitivo (e storicamente più utilizzato [GAV99]) in quanto l'occhio umano cerca nella sequenza di immagini oggetti estranei avendo presente la scena vuota.

Punto di forza di questa classe rispetto alla precedente è che la qualità dell'output non dipende dalla velocità di movimento dell'oggetto e, in particolare, l'individuazione è garantita anche nel caso in cui questo diventi stazionario. Negli algoritmi con riferimento è assente il problema di *foreground aperture* ma esiste la possibilità che gli oggetti abbiano la stessa tonalità dello sfondo, nel qual caso siamo in presenza di falsi negativi originati da *mimetizzazione* (o *camouflage*).

La presenza di un background di riferimento fa venir meno l'indipendenza dell'algoritmo da variazioni globali della scena, come i cambiamenti di illuminazione. Il background "fotografa" la scena in un certo momento: se avvengono cambiamenti che modificano le condizioni iniziali i dati di riferimento relativi alla scena "fotografata" non sono più validi. Nasce quindi la necessità di mantenere i dati contenuti nel background di riferimento aggiornati rispetto all'evoluzione della scena mediante tecniche opportune.

Come anticipato nel Paragrafo 5.4 questa classe è composta da due moduli distinti:

- Modulo di detection.
- Modulo di aggiornamento del background.

Analizzando i diversi tipi di moduli di detection sono state individuate due categorie principali di approcci (Figura 5-11):

- Algoritmi di detection statistici.
- Algoritmi di detection con immagine di riferimento.

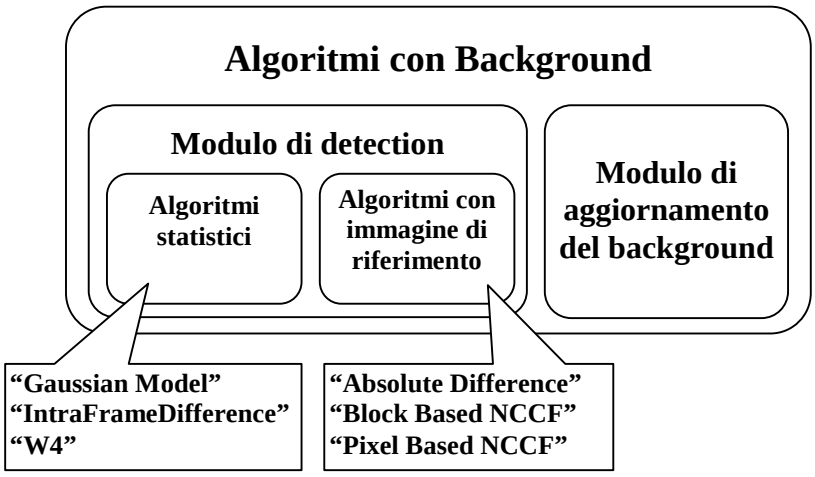

Figura 5-11 Struttura di un algoritmo con background.

5.6.1 Algoritmi di detection statistici.

Il modulo di "detection" di un algoritmo statistico classifica un pixel in base al confronto del suo valore con i dati statistici che lo riguardano. I pixel appartenenti allo sfondo sono soggetti a fluttuazioni continue. Queste variazioni sono spesso dovute, oltre che al rumore di digitalizzazione, a lievi movimenti di dettagli del background che possono causare l'individuazione di oggetti inesistenti. Per evitare questi inconvenienti, gli algoritmi eseguono una prima fase detta *training* in cui analizzano la scena in assenza di oggetti estranei per assorbire i fenomeni di "rumore motorio". Con questo termine si intendono tutti quei fenomeni di moto che non hanno alcun interesse ai fini dell'analisi degli

oggetti. In questo modo si costruirà un modello del background formato da un insieme di parametri statistici (non rappresentabili visivamente). In fase di rilevamento saranno considerati "punti di moto" soltanto i pixel i cui valori esulano dalla statistica del background. Gli algoritmi di questa classe mirano ad ottenere una detection robusta rispetto alla instabilità del background. Questo tipo di approccio può però presentare effetti indesiderati: se le fluttuazioni dei valori di un pixel generate da *waveing trees* durante il training sono ampie la sensibilità dell'algoritmo per quei pixel sarà limitata ai pochi valori non presenti nella statistica. Se un oggetto di interesse si dovesse muovere su una zona di sfondo affetta da *waveing trees*, potremmo vederne scomparire delle parti o, addirittura in casi sfortunati, potremmo non vedere affatto l'oggetto. Questi effetti variano molto a seconda del modello statistico scelto.
Gli algoritmi di detection statistici che sono stati esaminati sono : Gaussian Model, Intra Frame Difference e W4.

5.6.1.1 Rumore con distribuzione gaussiana ("Gaussian model")

Questo primo algoritmo statistico è stato implementato riducendo le ipotesi in [STA99] al solo caso gaussiano, supponendo che le fluttuazioni dei valori dei pixel a causa del rumore presente nelle immagini siano rappresentabili con una statistica gaussiana distribuita attorno ad un valore medio *M*. Sotto questa premessa, al termine della fase di training l'algoritmo stima i parametri della distribuzione gaussiana per ogni punto dell'immagine. In particolare il modello del background creato è costituito da una coppia di parametri (valore medio *M(x,y)* e deviazione standard $\sigma^2(x,y)$), che caratterizzano le fluttuazioni secondo il modello gaussiano. Il valore medio di un pixel alla colonna *x* e riga *y* è dato da:

$$M(x,y) = \frac{1}{N}\sum_{i=1}^{N} Immagine_i(x,y)$$

dove *N* indica il numero di frame che costituiscono il training e $Immagine_i$ indica la i-esima immagine tale fase. La varianza può essere espressa come:

$$\sigma^2(x,y) = \frac{1}{N^2}\sum_{i=0}^{N} Immagine_i(x,y)^2 - M(x,y)^2$$

Più ampie sono le escursioni attorno alla media, più la statistica risulterà distribuita e, di conseguenza, più grande sarà la deviazione standard.
Terminato il training, l'algoritmo classifica i pixel dell'immagine come appartenenti allo sfondo se il loro valore rientra in un intervallo pari al 96% della statistica stimata inizialmente. Viceversa un punto il cui valore ricade fuori da questo intervallo viene classificato come oggetto.
Questo risultato è ottenuto discriminando i pixel distano in valore assoluto dalla media *M(x,y)* meno della quantità $\sigma^2(x,y)$ (Figura 5-12):

$$immagineForeground(x,y) = \begin{cases} 255\ , & |immagine(x,y) - M(x,y)| > \sigma^2(x,y) \\ 0\ , & \text{altrimenti} \end{cases}$$

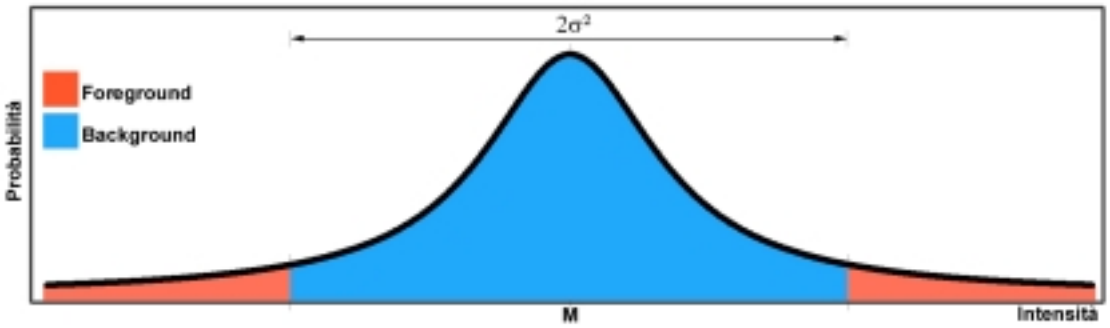

Figura 5-12 Soglia dell'elaborazione gaussiana.

A questo punto si ottiene un'immagine binaria che indica la presenza di oggetti estranei ma che risulta comunque affetta da una certa quantità di pixel rumorosi. Per migliorare la qualità dell'output viene eseguita un'operazione di filtraggio del rumore mediante un passo di erosione seguito da uno di dilatazione [CAN89] [PRA91].
Il risultato di una elaborazione è rappresentato in Figura 5-13.

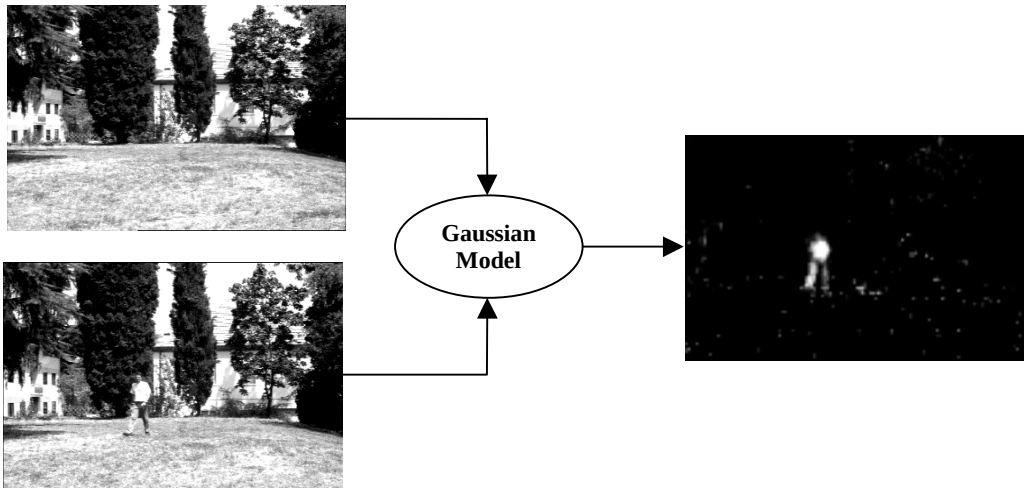

Figura 5-13 Esempio di elaborazione dell'algoritmo Gaussian Model.

In questo modo si attenuano gli inconvenienti dovuti ai fenomeni di *waveing trees*. La presenza di zone particolarmente rumorose può causare l'eccessiva riduzione della sensibilità dell'algoritmo, creando delle zone di "cecità" nelle quali può risultare difficile l'individuazione degli oggetti.
L'ipotesi di presenza di rumore distribuito in modo gaussiano e una soglia del tipo di Figura 5-12 possono risultare troppo penalizzanti nel caso che i valori siano distribuiti ampiamente a causa di fenomeni di *waveing trees*. Per attenuare questo inconveniente è implementato l'algoritmo di seguito descritto.

5.6.1.2 Massima differenza tra due frame ("Intra frame difference")

Per questo algoritmo è stato scelto un modello di background che utilizza parametri che garantiscono una sensibilità maggiore rispetto all'algoritmo precedente prescindendo dal modello gaussiano per la distribuzione del rumore nella sogliatura la deviazione standard è stata sostituita dalla massima variazione di intensità di un pixel tra due immagini consecutive, misurata durante il periodo di training. Il modello di background risulta costituito da due parametri per ogni punto dell'immagine:

- Il valore medio dell'intensità del pixel M*(x,y)*
- La massima variazione tra due immagini consecutive *d(x,y).*

Terminato il training, durante la fase di detection la soglia risulta come in Figura 5-14: un valore di grigio è considerato appartenente al background se dista, in

valore assoluto, meno della quantità *d(x,y)* dalla media *M(x,y)*. In caso contrario è considerato come punto oggetto.

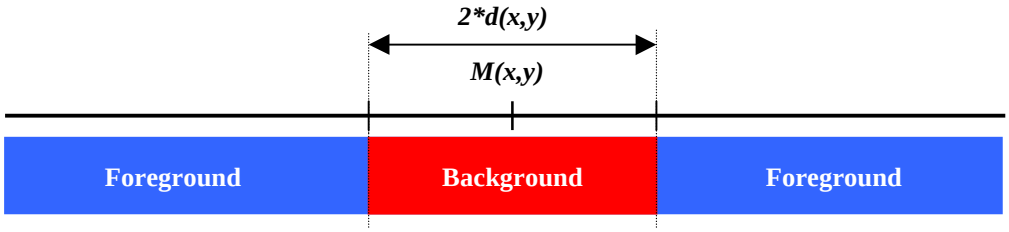

Figura 5-14 Soglia usata nell'Intra-Frame Difference model.

La soglia può essere anche formalizzata come segue:

$$immagineForeground(x,y) = \begin{cases} 255, |immagine(x,y) - M(x,y)| > d(x,y) \\ 0, \text{altrimenti} \end{cases}$$

Come già è avvenuto per i casi precedenti, l'immagine binaria così ottenuta non è ancora di qualità sufficiente per essere utilizzata dai livelli successivi del VSS. Così, per eliminare i pixel "rumorosi" si eseguono in successione un'operazione di erosione ed una di dilatazione per filtrare isolati pixel "falsi positivi".
L'elaborazione sinora descritta è sintetizzata in Figura 5-15

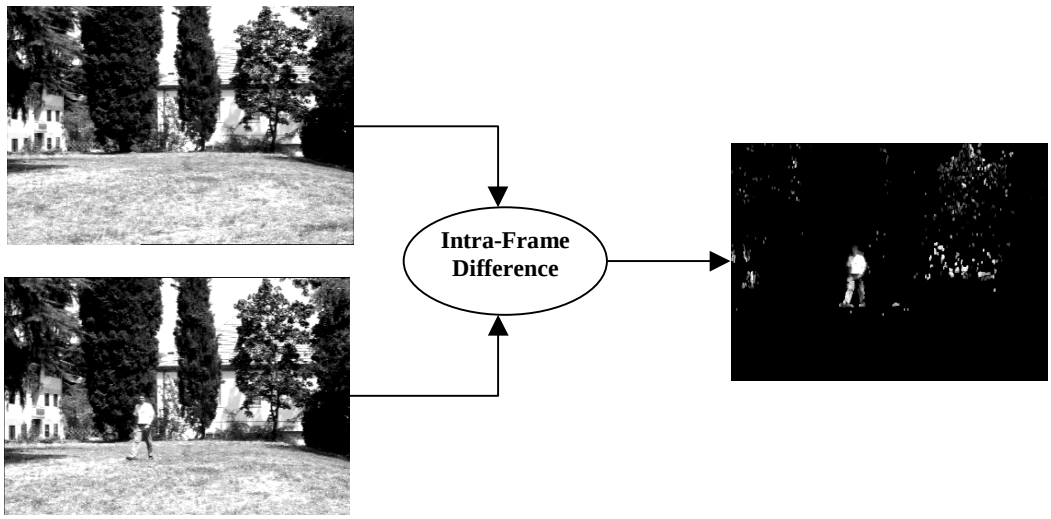

Figura 5-15 Esempio di elaborazione dell'algoritmo Intra-Frame Difference Model.

Confrontando la Figura 5-15 con la Figura 5-13 si nota come l'algoritmo "Intra frame difference" risulti più sensibile a fenomeni *waveing trees*, nella fattispecie costituiti dalle fluttuazioni delle foglie nella parte destra dello sfondo, ma filtri più efficacemente il rumore casualmente distribuito nelle rimanenti zone.

5.6.1.3 Algoritmo "W4"

Si è implementato l'algoritmo descritto in [HAR98] per il sistema denominato "W4-Hydra" al fine di migliorare la carenza di sensibilità nelle zone dove è presente rumore dovuto ai *waveing trees* mantenendo un buon filtraggio della componente del rumore. Infatti, come detto precedentemente, la sensibilità dell'algoritmo è confinata ai valori esterni alla statistica. Quindi se le fluttuazioni dei pixel sono caratterizzate da un'ampia distribuzione possono portare ad una eccessiva soppressione del rumore e quindi ad una mancanza di sensibilità locale o "cecità". Di conseguenza la corretta individuazione di un oggetto potrebbe risultare compromessa, in quanto l'oggetto verrebbe scambiato per sfondo.
Per limitare questo fenomeno, in W4 si introduce l'uso di una soglia doppia del tipo in Figura 5-16, derivante dalla somma di due intervalli.

Figura 5-16 Soglia utilizzata dall'algoritmo W4.

Nel caso il pixel non sia soggetto a fenomeni di *waveing trees*, ma solo a piccole oscillazioni introdotte dal rumore di digitalizzazione, i due intervalli di background definiti in rosso nella Figura 5-16 si sovrappongono dando origine ad una sogliatura simile a quella dei due algoritmi precedenti (Figura 5-12,Figura 5-14). Un esempio è mostrato in Figura 5-17 dove sono confrontate la distribuzione di un pixel generico e la relativa soglia calcolata dall'algoritmo W4.

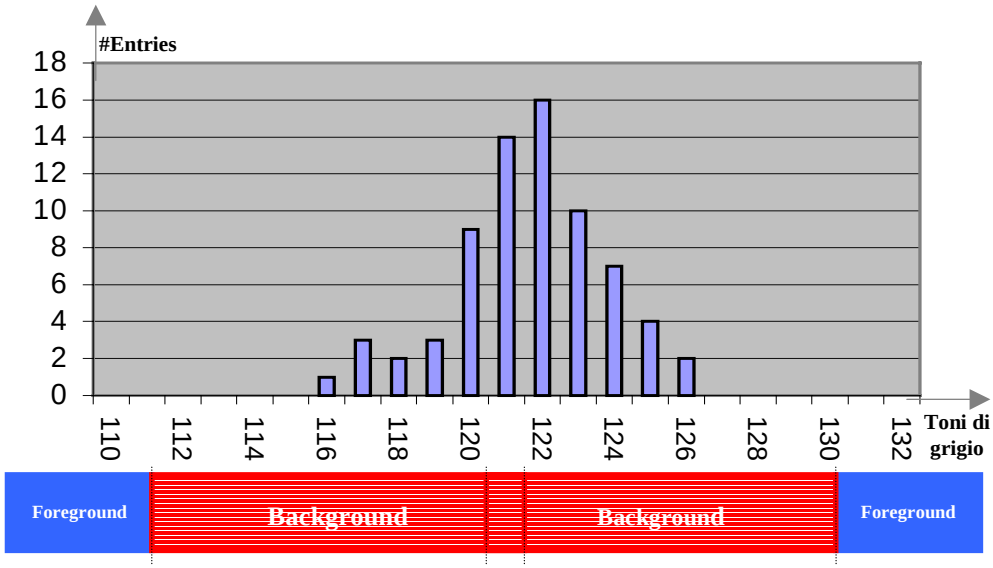

Figura 5-17 Esempio di distribuzione di un pixel con soglia calcolata dell'algoritmo W4. Caso di pixel affetto da rumore bianco.

Nel caso di una distribuzione molto ampia dovuta ad intensi fenomeni di *waveing trees*, le due soglie risultano invece separate in modo tale che i punti che ricadono nella parte centrale della statistica vengano riconosciuti come appartenenti ad un oggetto (Figura 5-18). Questa scelta, apparentemente in contrasto con l'obbiettivo di rendere insensibile l'algoritmo ai fenomeni di *waveing trees*, ha lo scopo di aumentarne la sensibilità. Come conseguenza di questa scelta l'output presentato finora da W4 è affetto da un cospicuo numero di *falsi positivi* in corrispondenza delle zone dove i w*aveing trees* sono particolarmente intensi.

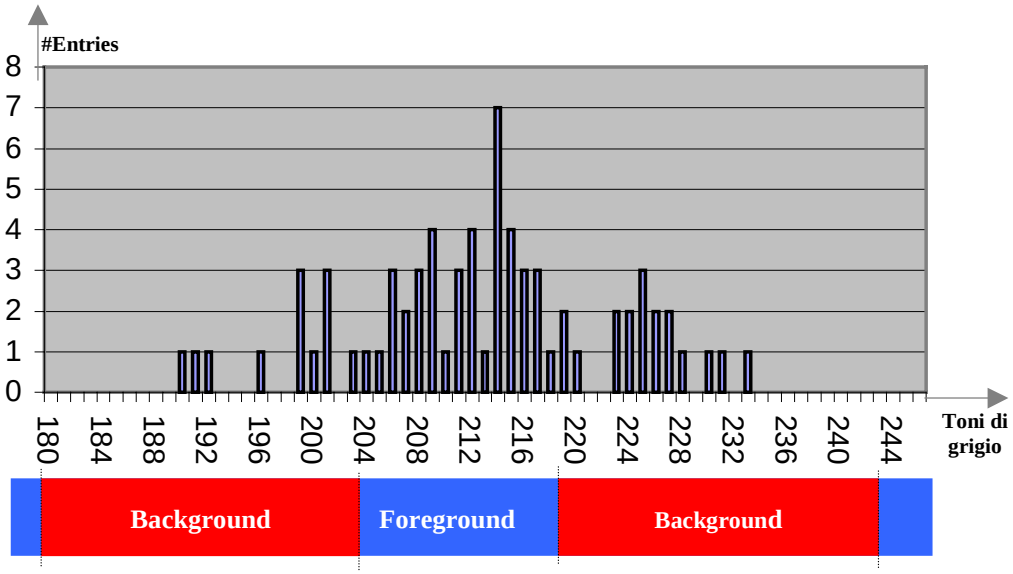

Figura 5-18 Esempio di distribuzione di un pixel con soglia calcolata dell'algoritmo W4. Caso di pixel affetto da intensi fenomeni di *waveing trees*.

Un esempio di output è mostrato in Figura 5-19: si può notare il numero elevato di pixel oggetto individuati erroneamente in corrispondenza degli alberi che si muovono nello sfondo.

Tuttavia è possibile eliminare i pixel *falsi positivi* dovuti a questo incremento di sensibilità sfruttando una ulteriore considerazione su questo tipo di rumore. Come si può notare in Figura 5-19, i pixel dovuti i *waveing trees* risultano per la maggior parte isolati tra di loro (non connessi). La casualità del rumore di digitalizzazione rende improbabile che pixel contigui di background ricadano contemporaneamente nell'intervallo "background" compreso tra i due "foreground", rendendo quindi improbabile che vengano a crearsi agglomerati di più punti di background contigui. Quindi è possibile filtrare statisticamente il rumore di digitalizzazione grazie alla particolare sogliatura di W4.

E' quindi sufficiente un passo di *opening* (*erosione* seguita da *dilatazione*) per eliminare la maggior parte dei *falsi positivi* presenti nell'immagine (Figura 5-19). Infine un'operazione di *closing* "coagula" i veri punti oggetto. Il risultato finale è mostrato in Figura 5-20.

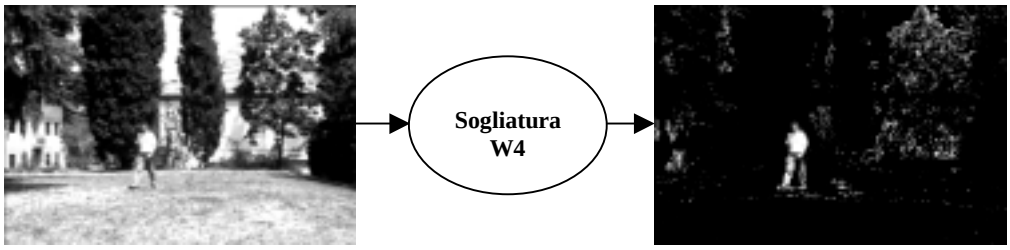

Figura 5-19 Primo passo di elaborazione di W4.

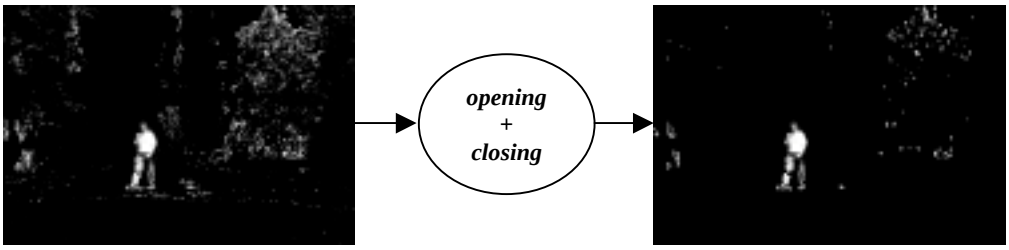

Figura 5-20 Secondo passo di elaborazione di W4.

Esaminando l'aspetto puramente implementativo, l'algoritmo, durante la fase di training, costruisce un modello del background a tre parametri per ogni punto dell'immagine :

- La minima intensità osservata durante la fase di training *M(x,y)*.
- La massima intensità osservata durante la fase di training *N(x,y)*.
- La massima differenza di intensità tra due immagini consecutive durante la fase di training *D(x,y)*.

I due passi di elaborazione spiegati poc'anzi permettono una binarizzazione dell'immagine. Un pixel è considerato foreground se soddisfa la condizione:

$$\left|M(x,y) - immagine(x,y)\right| > D(x,y) \wedge \left|N(x,y) - immagne(x,y)\right| > D(x,y)$$

In caso contrario i punti sono considerati appartenenti al background.

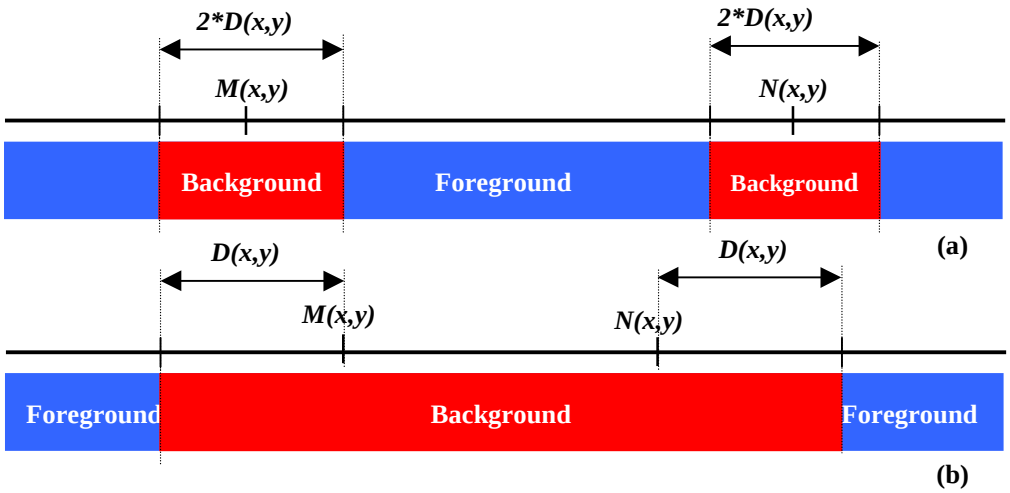

Figura 5-21 Soglie utilizzate per l'algoritmo W4 in caso di grandi (a) e piccole (b) distribuzioni del valore dei pixel.

Graficamente le configurazioni possibili per la soglia in W4 sono rappresentate in Figura 5-21, dove variano a seconda del tipo di distribuzione riscontrato durante il training.

In conclusione "W4" consente di raggiungere un buon filtraggio per *waveing trees* senza che questo degradi eccessivamente la sensibilità in tali aree.

5.6.2 Algoritmi di detection con immagine di riferimento.

La seconda sottoclasse di algoritmi (vedere Figura 5-11) usa un riferimento costituito da un'immagine che rappresenta la scena ripresa senza oggetti. L'individuazione degli oggetti è raggiunta valutando la somiglianza tra l'immagine corrente e quella nel background. Si possono applicare diverse tecniche di confronto ognuna delle quali presenta vantaggi ed effetti indesiderati in particolari condizioni.

Sono stati esaminati i seguenti moduli con detection basata su immagine di riferimento: Absolute Difference, Block based NCCF e Pixel Based NCCF.

5.6.2.1 Differenza dei valori assoluti (“Absolute difference”)

Questo primo modulo di detection rappresenta l’implementazione più semplice di un confronto con un’immagine di riferimento [GAV99]. Si calcola il valore assoluto della differenza tra ogni pixel dell’immagine corrente e del background: se questa è sufficientemente elevata il punto è classificato come oggetto (Figura 5-22).

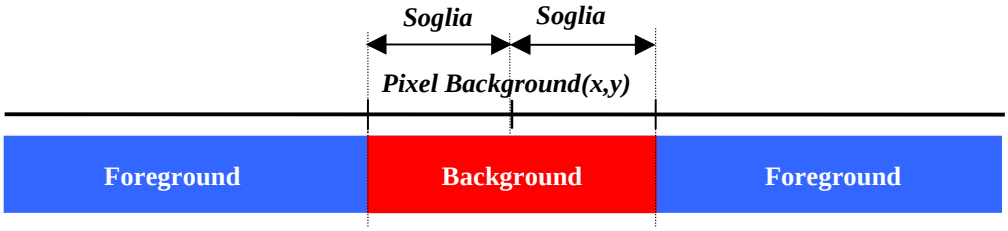

Figura 5-22 Soglia usata nell'algoritmo Absolure Difference.

La soglia di discriminazione dei pixel va necessariamente preselezionata dall’utente a seconda delle caratteristiche della scena. Una soglia bassa garantisce un’alta sensibilità ed è adatta ad immagini poco rumorose. Per riprese esterne, normalmente caratterizzate da rumore elevato (dovuto ad esempio ad alberi o cavi elettrici in movimento sotto l’azione del vento) la soglia va impostata ad un valore elevato. In tutti i casi si tratta di trovare il valore che consenta il giusto compromesso tra l’ammontare di pixel *falsi negativi* (sensibilità algoritmo) e di *falsi positivi* (filtraggio del rumore).
Dal punto di vista analitico tale condizione si può esprimere come:

$$immagineForeground(x,y) = \begin{cases} 255, |immagine(x,y) - background(x,y)| > soglia \\ 0, \text{altrimenti} \end{cases}$$

Un miglioramento della qualità dell’output si ottiene con un’operazione di *opening* seguita da una di *closing* in modo tale da filtrare in un primo tempo i pixel isolati dovuti al rumore per poi “amalgamare” quelli relativi ad oggetti. Un tipico esempio di elaborazione è rappresentato in Figura 5-23.

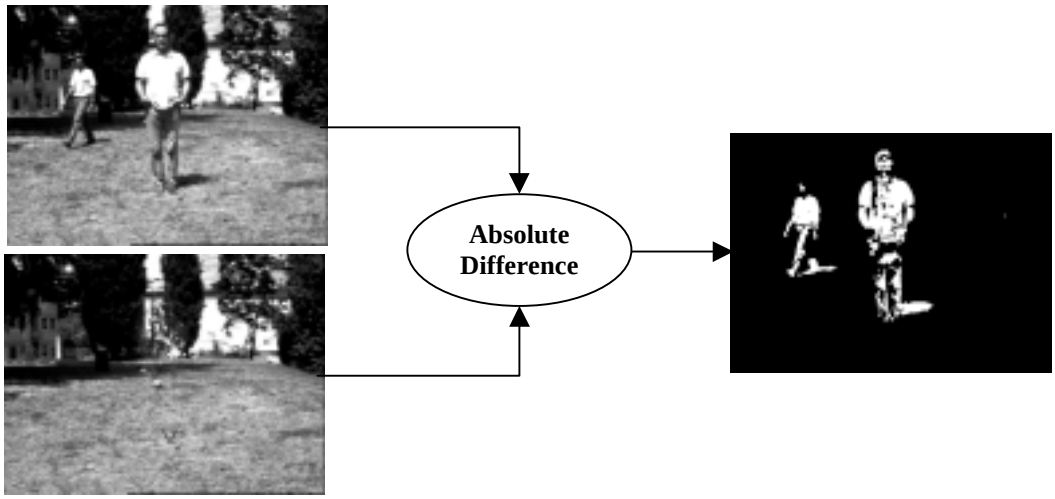

Figura 5-23 Risultato dell'elaborazione di Absolute Frame Difference.

Osservando l'output dell'esempio si vede che l'adozione di una soglia adeguata consente di ottenere un buon bilanciamento tra soppressione dei *falsi positivi* e buona sensibilità nella detection.

5.6.2.2 Algoritmo basato su NCC a blocchi ("Block based NCCF")

Come ampiamente discusso nei paragrafi 3-3 e segg. il confronto tra due immagini basato sulla funzione di Cross Correlazione Normalizzata (NCCF) consente una valutazione delle sole variazioni di texture a fronte di una buona immunità nei confronti delle variazioni di luminosità. L'immagine corrente ed il background sono divise in blocchi (Figura 4-1) applicando la NCCF tra i blocchi corrispondenti delle due immagini.

Un effetto indesiderato di questo approccio è l'inevitabile riduzione di risoluzione dell'output, conseguenza del fatto che l'elemento minimo di confronto è rappresentato dai blocchi in cui è stata suddivisa l'immagine. Tuttavia l'uso di una funzione particolarmente sensibile alla texture può risultare dannoso in presenza di rumore motorio del background che ha l'effetto di mutare la trama del blocco in esame. Anche in questo caso, come nel precedente, bisogna impostare una soglia per la binarizzazione il cui valore va scelto in base alle caratteristiche della scena ripresa:

$$blocco(x,y)=\begin{cases}\text{foreground}, CrossCorrelazioneBloocco(x,y) < soglia \\ \\ \text{background}, \text{altrimenti}\end{cases}$$

Un tipico esempio di elaborazione è mostrato in Figura 5-24.

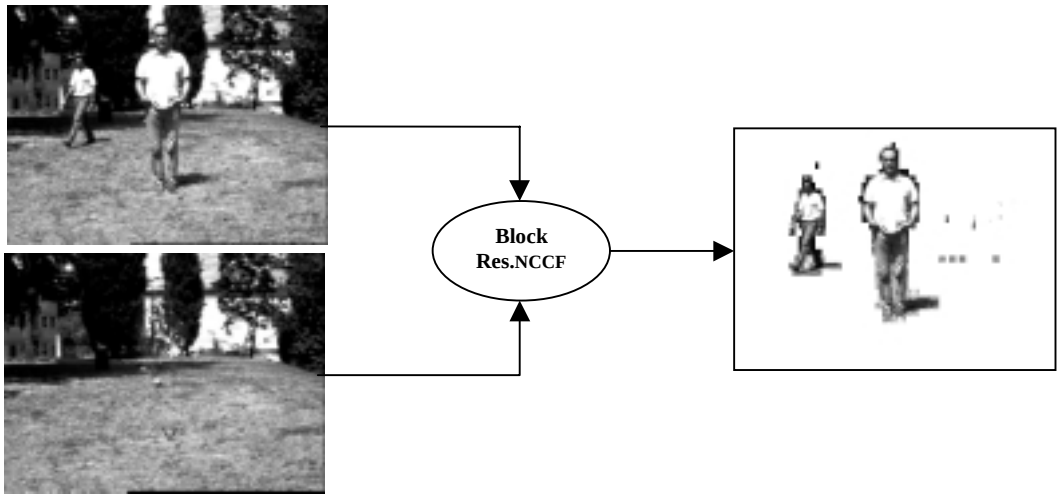

Figura 5-24 Risultato dell'elaborazione della Block Res. Nccf.

5.6.2.3 Algoritmo basato su NCC pixel a pixel ("Pixel based NCCF")

Questo algoritmo è stato concepito per sopperire alla scarsa risoluzione di quello precedente con l'intento di conservare i vantaggi derivanti dall'uso della NCCF. Per ottenere questo obiettivo, la correlazione tra l'immagine di riferimento e quella corrente è valutata pixel per pixel tra la coppia di blocchi centrati sui pixel in esame.

$$immagineOutput(x,y)=\begin{cases}\text{foreground}, Ncc(BloccoBackground(x,y), BloccoImmagineCorrente(x,y)) > soglia \\ \\ \text{background}, \text{altrimenti}\end{cases}$$

La classificazione di un pixel si effettua valutandone anche l'intorno. Questo porta ad una maggiorazione della sagoma degli oggetti individuati. Questo "bordo" aggiuntivo dipende dalle dimensioni finite dell'intorno su cui è calcolata la correlazione ed è da imputarsi al fatto che la presenza di un oggetto influenza la classificazione dei pixel esterni fintanto che la sagoma ricade all'interno del blocco di correlazione.

In Figura 5-25 il “bordo” è evidenziato in rosso.

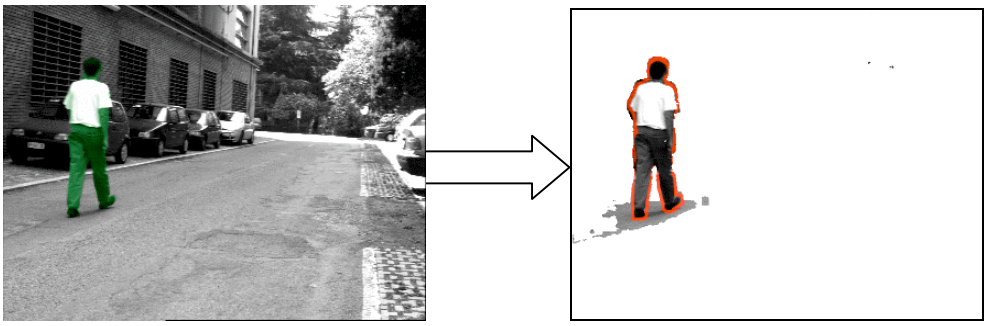

Figura 5-25 Effetto dell'algoritmo sulle sagome individuate.

Anche in questo caso è conveniente eseguire un passo di erosione e dilatazione (*opening*) al fine di filtrare ulteriormente i pixel rumorosi.
Un tipico output è rappresentato dall’immagine in Figura 5-26.

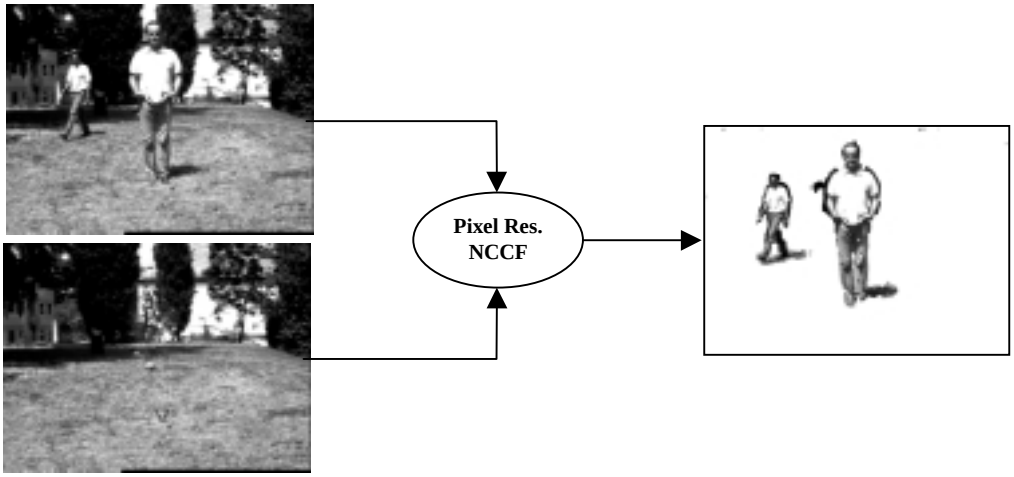

Figura 5-26 Risultato dell'elaborazione Pixel Res. Nccf.

5.7 Moduli di aggiornamento del background

Tutti gli algoritmi che fanno uso di un riferimento necessitano anche di uno stadio che lo tenga aggiornato rispetto alle variazioni accadute nella scena. Il funzionamento di tale modulo dipende dalla natura (dati statistici o immagine) del riferimento utilizzato. Inoltre ci possono essere situazioni in cui non è

possibile acquisire il background in maniera esatta, ovvero in assenza di oggetti nella scena. Ciò non deve pregiudicare in modo irrimediabile il funzionamento dell'algoritmo. E' quindi necessario che lo stadio di aggiornamento sia in grado di "recuperare" eventuali errori commessi in precedenza. Per filtrare eventuali picchi di rumore l'aggiornamento deve avvenire a fronte di variazioni persistenti e non può essere istantaneo.
Grazie al modulo di detection le informazioni possedute a questo livello riguardano la natura dei pixel dei quali sappiamo se rappresentano oggetti o sfondo. Il contributo dato dalla nostra ricerca consiste nell'utilizzare questa informazione per condizionarne l'aggiornamento, in particolare evitando che questo avvenga in corrispondenza dei pixel che il modulo di detection ha individuato come appartenenti ad oggetti (Figura 5-27).

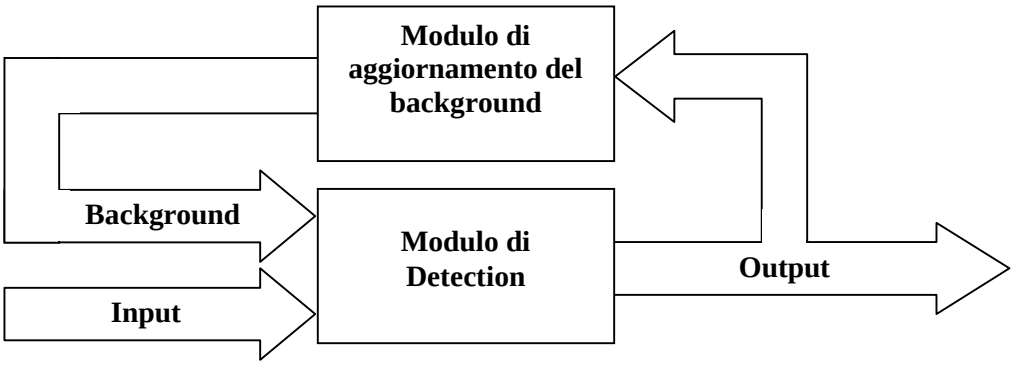

Figura 5-27 Funzionamento dei moduli in un algoritmo con background.

Il modulo di detection si occupa di estrarre oggetti di interesse dall'immagine corrente mediante un confronto con il background di riferimento. Quello di aggiornamento si occupa della manutenzione dei dati contenuti nel background in funzione dell'output generato dal precedente. Tanto più il riferimento sarà fedele allo stato attuale della scena senza oggetti, tanto più l'output sarà di migliore qualità. Allo stesso modo quanto migliore è la detection, tanto più l'aggiornamento potrà essere effettuato in maniera efficace.
Chiameremo questo metodo "aggiornamento a due pesi" per distinguerlo da quello classico detto "aggiornamento ad un peso" in cui la velocità di aggiornamento è costante ed indipendente dal significato semantico del pixel.

5.7.1 Aggiornamento del background “a un peso”

Come discusso nel Paragrafo 3-5, un modo intuitivo per aggiornare il background consiste nell’introdurre un peso per lo stato attuale della scena ed il coefficiente complementare per quello precedentemente memorizzato. Variando il peso si può scegliere di dare maggiore importanza allo stato pregresso del background, facendo in modo che le variazioni della scena attuale contribuiscano in maniera leggera a modificare il riferimento stesso. In questo modo però, se la scena subisce una improvvisa variazione (ad es. di luminosità), a causa del lento adeguarsi dello sfondo si generano molti falsi positivi. Viceversa si può scegliere di impostare il peso in modo da dare maggiore importanza allo stato presente, aggiornando in modo veloce il background: in questo modo l’algoritmo riesce a seguire eventuali variazioni della scena ma presenta alcuni inconvenienti. Le informazioni su da oggetti che si muovono lentamente potrebbero compromettere la fedeltà del background rispetto alla scena attuale a causa dell’elevata velocità di aggiornamento. Non sarebbe oltremodo possibile tenere traccia di quegli oggetti annessi al background che, dopo aver fatto il loro ingresso, si fossero fermati nella scena.

La soluzione in questo caso sta nel trovare un peso “di compromesso” adatto al tempo di permanenza caratteristico degli oggetti nella scena ed alla rapidità delle variazioni del background

Sotto l’aspetto implementativo il tipo di aggiornamento da eseguire varia a seconda che si tratti di un background costituito da un’immagine o da dati statistici.

Nel primo caso, il nuovo valore di un pixel del background ($PixelBackground_{n+1}$) è calcolato effettuando una media pesata tra il suo valore precedente ($PixelBackground_n$) e quello del corrispondente nell’immagine corrente ($Immagine_n$):

$$PixelBackground_{n+1}(x,y) = PixelBackground_n(x,y) \cdot peso + Immagine_n(x,y) \cdot (1 - peso)$$

dove la variabile *peso* assume valori compresi tra 0 e 1. Un valore vicino all'unità garantirebbe un aggiornamento veloce, viceversa un valore vicino allo zero ne produrrebbe uno lento. Nel caso di background di tipo statistico l'insieme di dati stocastici associati a un punto del background deve essere invece aggiornato in base ai valori assunti in un certo periodo di osservazione comprendente gli ultimi frame: variando la sua lunghezza si può decidere se effettuare un aggiornamento rapido o lento. Considerando l'andamento del pixel per un numero limitato di frame si ottiene un aggiornamento rapido; viceversa tenendo traccia di un maggiore (e quindi più distribuito) numero di valori, questi ultimi avranno un impatto minore sulla statistica, variandone di poco alla volta la distribuzione.

5.7.2 Aggiornamento background "a 2 pesi".

Il limite più evidente del modulo di aggiornamento a un peso appena descritto sta nella difficoltà a trovare pesi che siano davvero "di compromesso" in quanto spesso non è dato di sapere a priori né quanto veloci possano essere le variazioni del background, né quanto rapidamente un oggetto si muoverà nella scena. L'aggiunta di un secondo peso va in direzione di eliminare questo ostacolo, sfruttando l'informazione che si ha sul valore semantico (oggetto o sfondo) di ciascun pixel della scena: in corrispondenza di oggetti l'aggiornamento del background sarà "lento" per evitare di includere informazioni errate nel riferimento, viceversa per gli altri punti viene effettuato "velocemente" per garantire un aggiornamento fedele in presenza di variazioni nella scena. L'aggiornamento di un background-immagine è effettuato con una media pesata tra il valore precedente di un pixel del background ($PixelBackground_n$) e quello del corrispondente nell'immagine corrente ($Immagine_n$):

$$PixelBackground_{n+1}(x,y) = PixelBackground_n(x,y) \cdot PesoAgg(x,y) + Immagine_n(x,y) \cdot (1 - PesoAgg(x,y)$$

Questa volta però il peso di aggiornamento varia a seconda che il punto in questione appartenga ad un oggetto di interesse o allo sfondo:

$$PesoAgg(x,y)=\begin{cases} PesoVeloce, & Pixel(x,y)\notin oggetto \\ \\ PesoLento, & Pixel(x,y)\in oggetto \end{cases}$$

Nel caso di un background statistico (come nel caso ad un peso) la velocità di aggiornamento può essere cambiata variando l'intervallo di valori considerati: in caso il punto appartenga allo sfondo, l'aggiornamento può essere mantenuto veloce considerando un numero limitato di frame. Viceversa, se il punto rappresenta un oggetto, il numero di valori deve risultare maggiore per avere un aggiornamento lento della statistica.

Anche l'aggiornamento a due pesi presenta qualche inconveniente. Per come è concepito, questo algoritmo aggiorna in maniera molto lenta le informazioni nelle aree di background momentaneamente coperte da un oggetto, e d'altra parte non sarebbe possibile fare altrimenti proprio perché il background è nascosto dall'oggetto. In questo modo, se è in atto una variazione globale background (per es. dovuta ad un cambio di illuminazione) mentre un oggetto è presente nella scena, le zone da esso occluse manterranno inevitabilmente i valori antecedenti la l'occlusione. Nel caso l'oggetto riprendesse a muoversi rivelerebbe di nuovo lo sfondo che, a causa del mancato aggiornamento, potrebbe presentare differenze tali da determinare l'individuazione di oggetti inesistenti (*falsi positivi*). La correzione di errori di questo avviene grazie al peso per l'aggiornamento "lento" che deve essere mantenuto tale da permetterne il recupero in tempi ragionevoli.

5.8 Conclusioni

Abbiamo descritto la struttura degli algoritmi implementati e dei quali intendiamo valutare le prestazioni per mezzo di misure i cui risultati saranno discussi nel prossimo capitolo. Tra quelli descritti ve ne sono alcuni originali basati sul concetto da noi introdotto di "aggiornamento a due pesi".