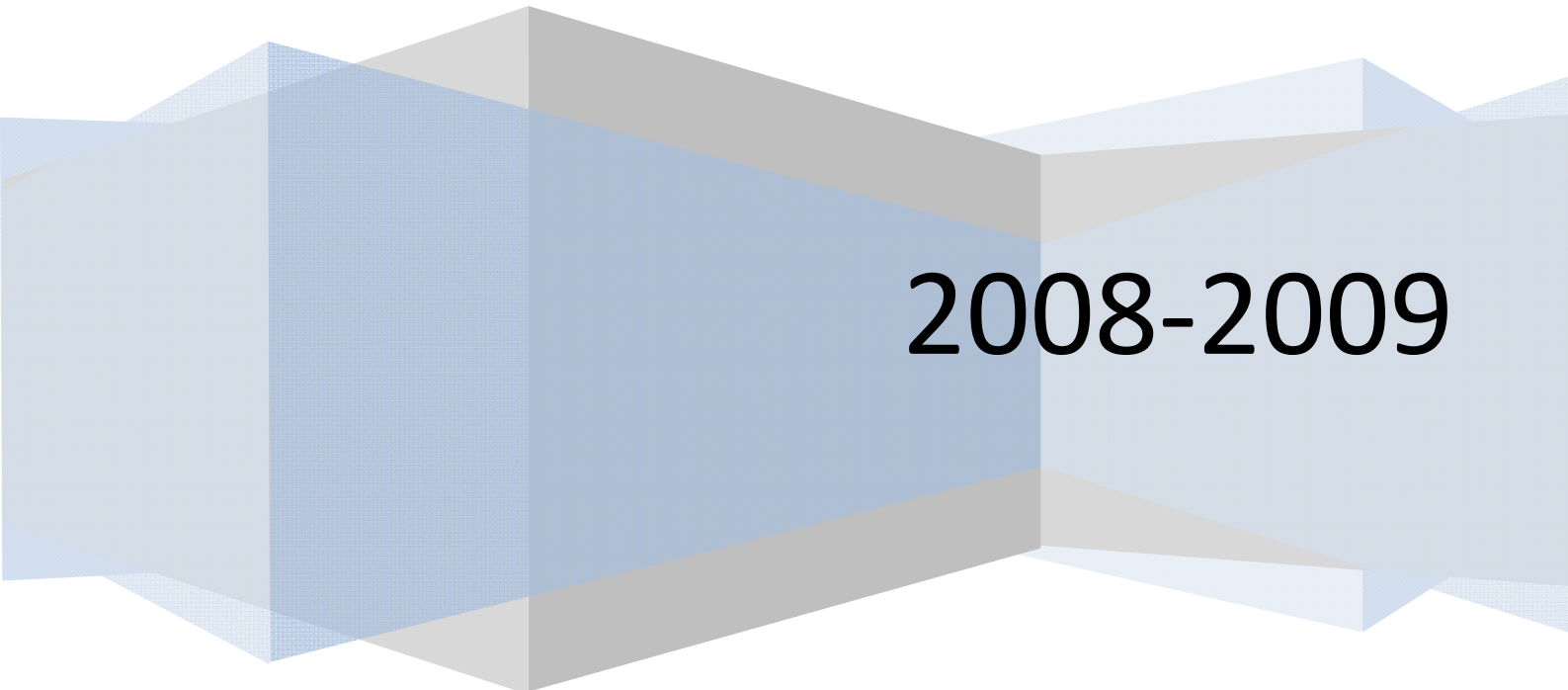


Università degli Studi di Bologna – Facoltà di Ingegneria

Change Detection in un Flusso Video

Elaborazione dell'Immagine LS

Alessio Della Motta



2008-2009

Introduzione

Il problema considerato richiede lo sviluppo di un sistema di visione in grado di individuare in un particolare flusso video (12 frame/s, 320x240 pixel, profondità 8 bit/pixel, compressione CODEC Cinepak Radius) la presenza di persone oppure di altri oggetti che vengono depositati o rimossi dalla scena. Caratteristiche rilevanti della sequenza video sono: telecamera statica, scena *indoor* con bassa profondità di campo, nessun frame permette una visione complessiva del *background* (almeno fra quelli iniziali), graduale cambiamento delle condizioni di illuminazione, range relativamente ampio di velocità dell'unico soggetto presente sulla scena.

Scelte di Implementazione

Per affrontare il problema della Change Detection si è deciso di utilizzare la tecnica di Background Subtraction, la quale permette di ovviare a problematiche quali *ghosting*, *foreground aperture* e non individuazione dei soggetti statici, tipiche degli approcci Two-Frame Difference e Three-Frame Difference.

Ottenuta una buona *Change Mask* (i dettagli sui procedimenti saranno illustrati nei paragrafi successivi), si è proceduto in modo classico attraverso la segmentazione di quest'ultima e la classificazione delle componenti connesse tramite appositi descrittori.

Per l'individuazione dei falsi oggetti si è fatto uso di edge detection, come descritto nell'ultimo paragrafo.

Gestione del Background

In questo paragrafo si descrivono le procedure messe in atto per utilizzare correttamente le informazioni sul background della scena.

Come primo passo, si è reso necessario generare dinamicamente il background, in quanto nessun frame iniziale della sequenza lo mostra nella sua integrità. Poiché l'operazione di *Background Initializing*, che corrisponde ad una sorta di training del sistema, può essere relativamente costosa¹ e durante la quale il sistema non svolge ancora le funzioni per le quali è stato installato, è buona norma utilizzare un basso numero di frame; il numero di frame utilizzato è stato

¹ In questa fase si devono raccogliere dati statistici su ogni pixel della scena e, nel caso di video di grosse dimensioni, ci potrebbe essere un'occupazione di memoria relativamente ampia. Inoltre, la computazione sui dati statistici raccolti (vista la mole di questi ultimi) tende a rallentare il sistema ed è bene che venga fatta una volta sola alla fine della raccolta dei dati. Ecco perché nel codice è presente il flag `BACKGROUND_TRAINING` che permette di selezionare se visualizzare o meno la costruzione del background frame per frame.

fissato a 55. Come funzione di approssimazione del valore dei pixel del background si è utilizzata la *mediana* che, come visto a lezione e sperimentalmente sul video considerato, fornisce la stima migliore con il minor numero di frame.

Il secondo problema da affrontare è stato l'aggiornamento del background (*Background Updating*), necessario per poter far fronte ad eventuali variazioni di illuminazione ed, in uno scenario più ampio, a modificazioni dello stesso (ad esempio tramite l'aggiunta o la rimozione di oggetti che dovranno permanere a lungo nella nuova locazione, così da poter essere considerati in futuro effettivamente parte del background). Quest'ultimo scenario è stato considerato, nonostante non fosse prioritario nel problema trattato.

Grazie a queste considerazioni si è sviluppata e decisa di utilizzare una variante della tecnica di Selective Background Updating. L'osservazione chiave è che non contribuiranno più all'aggiornamento del background solo i punti che nella Change Mask sono considerati *unchanged*, ma interverranno (in maniera meno incisiva) anche i punti *changed*; in questo modo, il background si riuscirà ad adattare ugualmente a modifiche permanenti. In formule:

$$\begin{cases} B_{t+1}(i,j) = \alpha F_t(i,j) + (1 - \alpha)F_t(i,j), & \text{se } BC_t(i,j) = 0 \\ B_{t+1}(i,j) = B_t(i,j) + \delta, & \text{se } BC_t(i,j) = P \\ B_{t+1}(i,j) = B_t(i,j) - \delta, & \text{se } BC_t(i,j) = N \end{cases}$$

Con B_t che indica il background all'istante t , F_t il frame all'istante t e BC_t una Change Mask apposta per il background, nella quale si tiene conto non solo del fatto che il pixel sia considerato *unchanged*, ma anche di che modifica esso abbia subito nel caso fosse *changed* (essendo il frame in scala di grigi, si terrà conto se esso sia diventato più chiaro o più scuro, rispettivamente $BC_t(i,j) = P$ e $BC_t(i,j) = N$). Aggiungere o rimuovere il valore δ permette di adattare in modo *fluid* il background² anche a fronte di modificazioni permanenti o rapide variazioni di illuminazione. Da notare che i ragionamenti fatti fino ad ora sono validi solo per immagini in scala di grigio; non è stato preso in considerazione lo scenario a colori. Inoltre, il caso presentato considera un δ che assuma valori continui; l'implementazione sviluppata si adatta, naturalmente, al caso reale discreto.

Change Mask Utilizzate

Come già accennato nel precedente paragrafo, oltre alla Change Mask (d'ora in poi FCM – Frame Change Mask) calcolata tramite *Background Difference*, vi è anche una Change Mask apposta (BCM – Background Change Mask) per l'aggiornamento del background.

² Nelle conclusioni sarà esplicitata un'ulteriore conseguenza di questo approccio, che in questa sezione si è deciso di omettere.

Questo permette di gestire con soglie diverse ciò che si vuole considerare come variato nel background e nella scena; l'output finale utilizzerà come soglia di FCM il valore 25, mentre come soglia di BCM il valore 10.

Un'ultima considerazione è che non si applicano operatori morfologici alla BCM: anche *outliers* possono intervenire nell'aggiornamento del background, ma il loro effetto sarà mediamente nullo o irrilevante (come in effetti si verifica); saranno applicati, invece, operatori morfologici alla FCM per ottenere una buona Change Mask.

Elaborazione della FCM

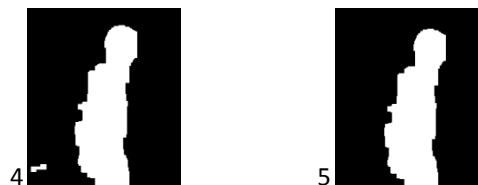
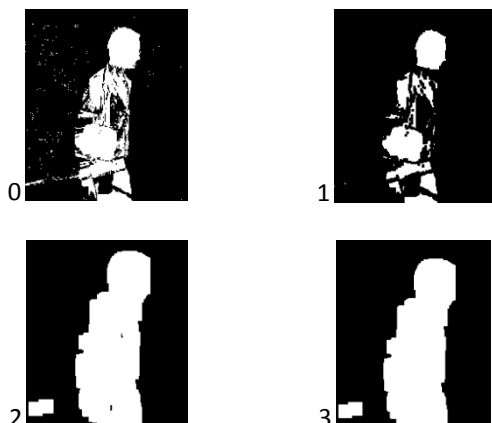
Dopo aver ottenuto la FCM tramite *Background Difference*, essa viene trattata appositamente per eliminare vari difetti, quali falsi positivi, falsi negativi, oggetti divisi in più parti, ecc.

La sequenza di operazioni applicate è la seguente:

1. Erosione preliminare per eliminare essenzialmente *outliers* falsi positivi, di modo che essi non intervengano nelle dilatazioni successive.
2. Dilatazione (ripetuta più volte) utilizzata, ad esempio, per unire parti di oggetti rilevate separate.
3. Area Opening dello sfondo, ovvero applicato sul negativo della FCM: in questo modo si eliminano le cavità degli oggetti (falsi negativi) con area maggiore di un certo valore costante.
4. Erosione per ridimensionare l'intervento delle precedenti dilatazioni, che hanno aumentato l'estensione degli oggetti rilevati.
5. Area Opening (diretta) per eliminare falsi positivi.

L'operazione di Area Opening viene fatta sempre a valle di una segmentazione e questo comporta un certo costo computazionale il quale, però, è stato ritenuto accettabile alla luce delle performance finali.

Vediamo un esempio della sequenza delle operazioni:



Vi è una controindicazione nell'utilizzo dell'area opening dello sfondo, come si può vedere dall'esempio seguente:



Si può notare come l'angolo in basso a sinistra venga inglobato nella sagoma della persona, perché possiede un'area inferiore alla soglia decisa per l'area opening dello sfondo; questo è un effetto che può manifestarsi in generale quando la persona si trova in prossimità dei bordi dell'immagine. Tale effetto è stato valutato e considerato tutto sommato accettabile alla luce dell'output complessivo.

Un'ultima osservazione: si è a conoscenza del fatto che gli oggetti da rilevare nel filmato non presentano cavità che possano essere in qualche modo rilevanti ai fini del problema; in altri scenari occorre prestare maggiore attenzione nell'utilizzo dell'area opening dello sfondo.

Segmentazione e Object Recognition

Dopo aver processato la FCM, si procede con la segmentazione e la classificazione delle componenti connesse presenti sulla scena.

L'algoritmo di *segmentazione* implementato è sostanzialmente quello visto a lezione, capace di effettuare il labeling in sole due scansioni dell'immagine, appoggiandosi ad una matrice delle equivalenze ed a una LUT. È stata fatta una leggera modifica (per la quale si rimanda al codice) che permette di restituire il numero effettivo di label assegnate (eliminando quelle temporanee equivalenti) e riassegnando i valori di tali label in maniera crescente cosicché, se nell'immagine fossero presenti 3 oggetti, essi sarebbero labellati rispettivamente con i valori uno, due e tre, qualunque sia la loro forma.

L'algoritmo di *object-recognition* non fa altro che ciclare sul numero di label restituito dalla funzione di segmentazione e, per ciascuna di esse, calcolare i descrittori (vedi dopo) dell'oggetto identificato dalla label considerata mano a mano; l'assegnazione della classe all'oggetto specifico (ovvero discriminare se si tratta di un libro, una persona, altro) viene demandata

ad una funzione passata come parametro (si è utilizzato un puntatore a funzione) in modo da separare il calcolo dei descrittori dal loro utilizzo nella classificazione, mantenendo l'algoritmo valido in generale. Per ulteriori dettagli si consulti il codice.

Descrittori Calcolati ed Utilizzati

Sono state implementate funzioni per il calcolo dei seguenti descrittori: area, perimetro, compattezza, circolarità di Haralick, momento centrale di ordine generico.

Grazie alla funzione per il calcolo dei momenti, è stato possibile definire una serie di macro per calcolare altri descrittori in modo diretto, come il baricentro ed il primo invariante di Hu.

Le uniche considerazioni da fare riguardano il perimetro e la misura di circolarità di Haralick. Per quanto riguarda il primo, è stato implementato estraendo il contorno dell'oggetto ed "inseguendolo", senza utilizzare il metodo delle configurazioni dei pixel vicini. La circolarità di Haralick, invece, è stata calcolata come $\frac{\sigma_R}{\mu_R}$, anziché $\frac{\mu_R}{\sigma_R}$; questo perché considerando il caso, ad esempio, della circonferenza, essa ha media non nulla (precisamente pari al raggio) e varianza nulla, il che porterebbe ad una divisione per zero.

In realtà, nell'algoritmo di riconoscimento degli oggetti specifico si è sfruttato solamente il descrittore di area, il quale permette da solo un buon livello di correttezza nella classificazione; ciò non toglie che sia possibile perfezionare ancor più l'algoritmo per ridurre al minimo gli errori di rilevazione.

Individuazione dei Falsi Oggetti

Tipicamente un falso oggetto dovrebbe essere rilevato quando, a fronte della rimozione di un oggetto presente sulla scena, la background difference individua un cambiamento ed evidenzia l'area (o meglio il contorno) della zona in cui prima tale oggetto era collocato. Non di meno, il profilo di un oggetto realmente presente sulla scena dovrebbe provocare degli edge netti a valle di un'operazione di edge-detection, cosa che non ci si aspetta (se non in particolari condizioni della zona in cui l'oggetto era posizionato) per un falso oggetto.

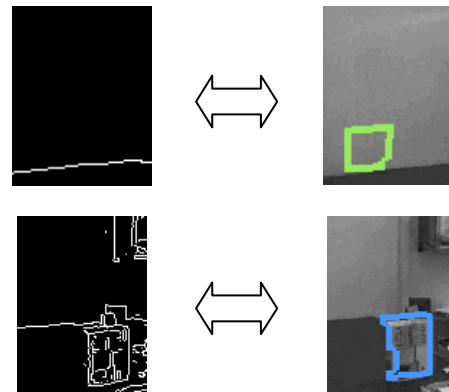
Preso nota di queste considerazioni, si è deciso di attuare la seguente procedura per riuscire a determinare gli oggetti non reali:

- Si applica al frame corrente una edge-detection (in particolare l'edge-detector di Canny) opportunamente tarata.
- Si calcola, per ogni oggetto, il numero di intersezioni tra il contorno costruito tramite

background difference (e relative elaborazioni) e gli edge individuati al passo precedente.

- Se il numero di intersezioni è superiore ad un certo valore di soglia si tratta di un vero oggetto, altrimenti esso sarà considerato come falso.

Il principio utilizzato è esemplificato dalle seguenti illustrazioni che rappresentano in ordine un falso oggetto ed un vero oggetto:



Infine, il seguente è un frame ricavato dall'output come esempio di funzionamento del sistema:



Conclusioni

Il sistema è stato tarato opportunamente per riuscire a classificare ed identificare correttamente tutti gli elementi della scena. Ciò non toglie che agendo ulteriormente sui parametri si possa migliorarne ancora l'output.

Come ultima nota, poiché il background mano a mano che il filmato procede si adatta alla nuova configurazione (ovvero non vi sarà più il libro sullo sfondo e comparirà quello in primo piano) i due elementi, falso oggetto e nuovo oggetto, saranno rilevati sulla scena per un tempo limitato, cioè fino a quando il background non sarà completamente trasformato. Questo fenomeno non è visibile nella simulazione, poiché il video non procede abbastanza per poter rendere evidente questo comportamento.