UNIVERSITÀ DEGLI STUDI DI SIENA
Facoltà di Ingegneria

Corso di Laurea Magistrale in
INGEGNERIA INFORMATICA

Tesi Sperimentale

TECNICHE DI VIDEOSORVEGLIANZA MEDIANTE ESTRAZIONE DI FEATURE DINAMICHE E STRUTTURALI E METODI DI FUSIONE DATI: FEATURE STRUTTURALI

11 Dicembre 2006

Candidato:
STEFANO FROSONI

Relatore: ALESSANDRO MECOCCI

ANNO ACCADEMINO 2005-2006

RINGRAZIAMENTI

Prima di lasciare spazio alla tesi desidero ringraziare con tutto il cuore la mia fidanzata Emanuela che mi è sempre stata vicina e mia ha sempre aiutato anche nei momenti più difficili di questi anni.

Un ringraziamento particolare alla mia famiglia che mi ha sostenuto in questa lunga avventura senza perdere mai la fiducia nei miei confronti.

Un grazie particolare ad Antonio, un grande amico, grazie pure a Marco, al Professor. Alessandro Mecocci, la mente di questo progetto ed al Professor Marco Maggini che è stato la mia guida per affrontare questi anni universitari.

Ma se è vero che la tesi è solo l'ultima tappa di un percorso iniziato ben prima, non posso allora esimermi dal ringraziare tutte le persone e gli amici che, in un modo o nell'altro, mi sono state vicine nel corso di questi faticosi, ma splendidi ed unici anni della mia vita.

La nostra conoscenza, se paragonata alla realtà, è primitiva e infantile. Eppure è il bene più grande di cui disponiamo.
Albert Einstein

Non esistono grandi scoperte né reale progresso finché sulla terra esiste un bambino infelice.
Albert Einstein

Un uomo percorre tutte le strade del mondo per trovare ciò che gli serve, ma deve tornare a casa per scoprirlo.
George Edward Moore

INDICE

INTRODUZIONE

La crescente importanza attribuita nel corso degli ultimi anni al tema della sicurezza è stata di impulso per un notevole sviluppo delle applicazioni di videosorveglianza ed elaborazioni dell'immagine. Tale sviluppo c'è stato sia a livello di hardware che a livello di algoritmi matematici e di software. La loro più naturale evoluzione ha riguardato principalmente sistemi intelligenti, in grado di monitorare l'area o gli oggetti assegnati anche senza la presenza dell'operatore umano.

Si tratta visibilmente di sistemi di una certa complessità: sviluppare un apparato di videosorveglianza (eventualmente non presidiato) in grado di rilevare autonomamente oggetti e/o intrusi; significa di fatto realizzare per via artificiale i processi di osservazione visiva propri dell'intelligenza umana.

L'intrinseca difficoltà di questa realizzazione, è caricata d'ulteriori vincoli volti a garantire livelli ottimali o comunque accettabili di efficienza e robustezza. Considerato che tali requisiti impongono necessariamente l'adozione d'algoritmi complessi, per i sistemi di videosorveglianza che potremmo definire *intelligenti* ne deriva pertanto un lavoro computazionale considerevole.

Il problema principale che s'incontra in questi sistemi, è che nonostante gli algoritmi d'elaborazione delle immagini abbiano ormai raggiunto notevole complessità ed efficienza, i risultati che si ottengono per quanto riguarda il loro utilizzo in sistemi intelligenti, non sono per nulla soddisfacenti.
Le difficoltà maggiori sono proprio nella parte intelligente di questi sistemi, che quindi rimangono affiancati dall'uomo nell'azione di vigilanza.

Il sistema progettato in questa tesi insieme con altri vuole superare questa limitazione, fornendo una videosorveglianza intelligente che può essere sfruttato per svariati scopi: per esempio può essere utilizzato su una galleria di quadri di un museo per verificare se si sono verificati furti od atti vandalici. Lo scopo di questa tesi, quindi, è lo studio di un sistema di videosorveglianza non sorvegliato che riesca a determinare situazioni pericolose in ambienti dove il controllo e la sorveglianza sono di importanza decisiva.

Il progetto prevede l'utilizzo in modo parallelo di tre algoritmi d'estrazione delle feature; ciascuno di essi fornisce la propria stima in tempo reale sugli eventi che sono accaduti nel video, come per esempio lo spostamento di un quadro. Queste tre stime sono abbastanza efficienti, ma da sole non offrono un sistema intelligente poiché soffrono degli stessi problemi degli altri apparati già esistenti. La parte fondamentale del sistema è quella di comprendere un *super decisore* che esegue una successiva elaborazione sulle informazioni fornite dai singoli decisori per fornire un risultato più intelligente e più simile all'uomo.

Quindi, per esempio, il sistema riesce a capire da solo se quel particolare quadro è stato mosso o no, riuscendo a distinguere il movimento del furto da quello che invece riguarda le persone che gli passano davanti, tutta una serie di movimenti che potenzialmente non sono "pericolosi", e che in un sistema semplice sarebbero stati considerati come furto.

Nel primo capitolo vengono illustrate le tecniche classiche di videosorveglianza effettuando una panoramica sia sulla videosorveglianza analogica che digitale.

Il secondo capitolo espone una panoramica esaustiva sul mondo dell'elaborazione delle immagini e riporta una serie di algoritmi, utilizzati per la realizzazione del progetto tra i quali alcuni, come la trasformata di Hough e la mediana (estrattori di feature strutturali), che sono alcune delle tecniche principali per il sistema di controllo di ambienti.

Il terzo capitolo è quello più esteso e descrive il sistema nel suo complesso. Esso mostra prima di tutto un descrizione degli strumenti software utilizzati, quindi librerie ed ambiente di sviluppo. Poi passa ad analizzare in dettaglio i tre livelli di cui il sistema di videosorveglianza si compone: l'estrattore di feature con particolare rilevanza per le feature di tipo strutturale; e gli altri due livelli del sistema: le regioni di fusione dei dati e il decisore finale .

Il quarto ed ultimo capitolo raccoglie una serie di esperimenti con i risultati ottenuti, effettuati tramite il software sviluppato ed una videocamera gestita direttamente dal sistema o tramite dei filmati che vengono analizzati alla ricerca di situazioni a rischio. Tramite questi esperimenti si vuole dimostrare dei casi d'uso concreti del sistema. Gli esperimenti sono stati effettuati nel tipico scenario della pinacoteca dove andiamo a controllare dei quadri, (nulla vieta che il sistema possa essere sfruttato per controllare ambienti differenti) e dimostrano la bontà del sistema in questo tipo di contesto, rivelandosi decisamente efficace.

CAPITOLO 1

STATO DELL'ARTE

Scopo di questo primo capitolo è offrire una panoramica generale sulle problematiche della videosorveglianza. Saranno quindi presentati brevemente i principali algoritmi utilizzati per il riconoscimento d'oggetti e per l'estrazione di feature. Poi sarà presentata la struttura generale del sistema software progettato in questa tesi con la descrizione delle singole componenti in cui esso è strutturato. In particolare sarà descritto il primo livello del sistema ovvero quello comprendente i tre estrattori e gli algoritmi di stima. Invece saranno accennati solamente gli altri due livelli poiché verranno approfonditi nel secondo capitolo.

1.1 STRUTTURA GENERALE DEL SISTEMA DI VIDEOSORVEGLIANZA

Lo scopo principale del sistema è l'individuazione di *situazioni potenzialmente pericolose* in casi il più possibile generali ; nella parte di sperimentazione del sistema ci si è in particolar modo concentrati sulla sicurezza di opere d'arte per una pinacoteca. Per riuscire in questo scopo le immagini provenienti da una videocamera (o filmato digitale), sono analizzate fotogramma dopo fotogramma. L'analisi si compone di *tre parti fondamentali* che vengono elaborate di continuo tramite il flusso di fotogrammi *(frame-rate 1fps* circa).

Primo livello.

Questo livello è quello più basso e comprende i tre algoritmi d'estrazione delle immagini ed i metodi di stima eventualmente connessi.

L'intelligenza decisionale che si otterrebbe utilizzando solamente questo strato non sarebbe molto valida, infatti possono essere ritenute situazioni d'allarme anche situazioni che in realtà non lo sono; per fare un esempio, se si utilizza solamente il riconoscimento dei contorni su un quadro, e ci sono delle persone che si muovono davanti al quadro queste vengono rilevate come movimento di contorni e di conseguenza sono rilevate come situazione di allarme. Per quanto riguarda gli algoritmi d'estrazione delle feature si sono utilizzati due algoritmi per feature strutturali ed uno per feature dinamiche.

Il primo dei tre estrattori utilizza la trasformata di *Hough* per trovare linee rette che dovrebbero indicare i bordi dell'oggetto che si intende osservare, come per esempio le linee della cornice di un quadro. Si è utilizzata la trasformata di Hough per ricercare all'interno della scena i pixel che sono a *massima invarianza*. Questi punti, infatti ci

aiutano a capire se nella scena ci sono delle forti variazioni che possono essere sospette. Il fotogramma corrente viene prima di tutto sottoposto al filtro di Canny per riuscire a determinare degli edge nello scenario. Gli edge trovati sono poi sottoposti alla trasformata di Hough per determinare quelli maggiormente stabili. Per quanto riguarda il decisore implementato in quest'estrattore si è utilizzato un algoritmo di stima non parametrica, *il Parzen Window*. Le informazioni derivanti dal filtro di Hough sono state quindi utilizzate per addestrare e testare questo stimatore. Nei primi 50 frame il sistema è assunto pressoché stabile. In seguito viene creata una regione composta da questi fotogrammi, sulla quale vengono poi confrontati tutti i successivi, calcolando così una probabilità che ci dice quanto il frame corrente si avvicina alla stabilità. Si è scelto questo tipo di feature perché riesce in modo semplice ma accurato a determinare i rapidi cambiamenti nella scena. In questo modo si elimina tutta la ridondanza tipica degli edge che renderebbe computazionalmente molto pesante la ricerca delle variazioni all'interno dei filmati, concentrando così l'informazione.

Il secondo dei tre estrattori è una *mediana*: sono presi dei frame ad intervalli regolari e si calcola il valore mediano su un blocco di questi frame riordinati. In questo modo si cerca di ottenere un'immagine che è il valore intermedio tra molti frame ed indice di una variazione persistente nella realtà. Questa estrazione avviene in maniera sotto campionata; si cerca tramite il sotto campionamento di ridurre le variazioni "frame by frame" e di far calcolare alla mediana solo oggetti che rimangono stabili nel tempo. La mediana è formata da 5 frame in toni di grigio che vengono ordinate tramiti *l'algoritmo Bubblesort*, usando *il livello di grigio* associato a ciascun pixel come misura per i confronti. La mediana calcolata ogni 50 frame (fattore di sottocampionamento moltiplicato per numero di frame che compongono la mediana) viene aggiornata con la tecnica *FIFO (first in first out)*. Una volta calcolata la mediana, a questa viene sottratto il frame corrente, calcolato in valore assoluto, riuscendo così a determinare gli oggetti che si discostano dai frame che la mediana rappresenta. L'immagine differenza viene poi binarizzata e vengono inviate allo strato successivo solo le informazioni posizionali dei pixel chiari. Questo estrattore permette ad esempio di non considerare situazioni d'allarme spostamenti veloci, come quando una persona passeggia davanti ad un quadro. Le feature appena descritte sono definite feature strutturali.

Il terzo estrattore è un algoritmo di *Optical Flow* che cerca di percepire i movimenti che avvengono su ogni singolo frame. L'elaborazione è stata effettuata tramite *l'algoritmo iterativo di Lucas-Kanade* che riesce a rilevare quei punti nell'immagine che si spostano in due frame consecutivi. L'algoritmo viene calcolato su due frame consecutivi; uno è quello processato, l'altro è tenuto in memoria. Su questi due, tempo T e tempo T-dt viene calcolato l'algoritmo. L'immagine sorgente è stata suddivisa in tre porzioni identiche: *zona bassa*, *zona intermedia* e *zona alta*. Su ciascuna parte è stato applicato l'algoritmo di Lucas-Kanade. Ad ognuna di queste zone è data una particolare importanza. Per esempio se la videocamera è posta in alto ed il quadro è collocato ad una particolare altezza per la quale la zona alta non viene mai occupata da altre cose (tipo braccia o teste di persone) , allora un movimento in questa zona sarà molto importante. Diversamente un movimento della zona bassa sarà meno importante visto che a quest'altezza sarà frequente vedere movimenti di persone. Il decisore per questo livello funziona con un valore di soglia settato sul luogo, superato il quale l'algoritmo fornisce una situazione d'allarme al supervisore.

L'analisi attraverso queste tre componenti è solamente il primo livello del sistema. I tre stimatori forniscono delle visioni parziali e riduttive se presi singolarmente, ma se considerati in maniera unitaria riescono a darci un inquadramento molto preciso di ciò che sta avvenendo nella scena sorvegliata e questo è quello che avviene nei livelli successivi.

Secondo livello:

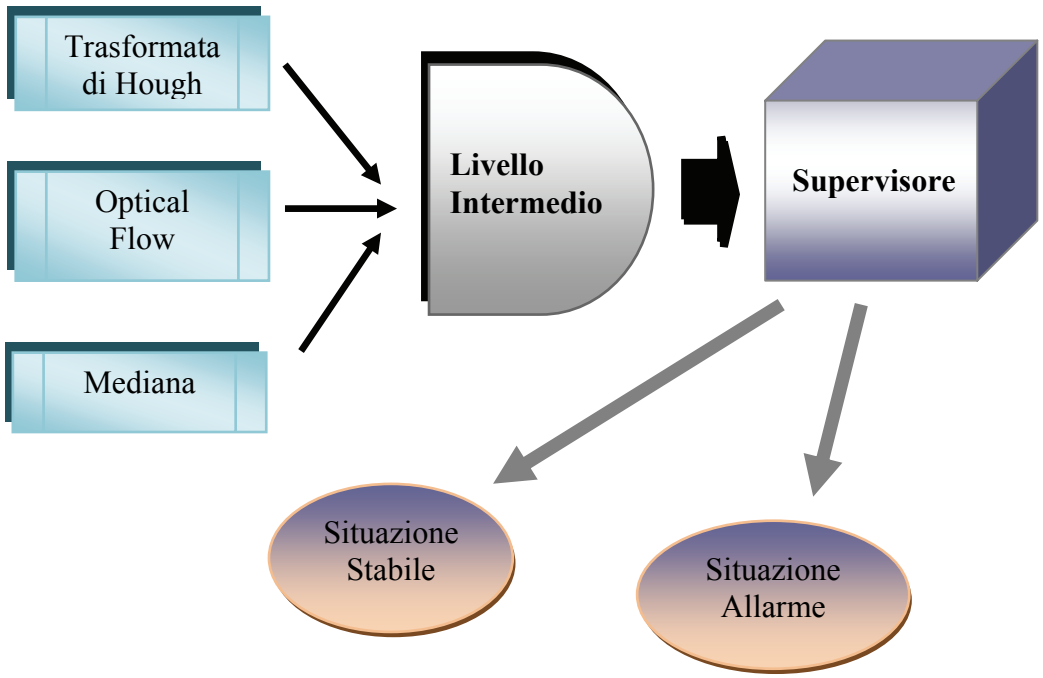
Il secondo livello raccoglie le informazioni dei tre estrattori e le classifica in base ad una *griglia*. Questa griglia suddivide l'informazione ottenuta dagli estrattori in base alla posizione spaziale che essi hanno nell'immagine ripresa.

Tramite questa griglia dividiamo l'immagine in 25 caselle, ciascuna etichettata per mezzo di un "*identificatore numerico"*. Queste caselle servono per riuscire a mantenere un conteggio separato per ciascuno dei tre analizzatori. Ogni qualvolta un pixel viene rilevato da un estrattore il conteggio relativo a quest'ultimo nella casella di appartenenza del pixel viene incrementato. Si genera così sull'immagine *una mappa di informazioni* sulle aree maggiormente stabili o instabili , informazioni sulla quale poi un super decisore dovrà stabilire se siamo in presenza o meno di una situazione potenzialmente pericolosa.

Terzo livello:

Questo strato è il *supervisore*, che ha il compito di raccogliere tutte le informazioni che i due precedenti livelli gli hanno fornito per dare un'informazione intelligente dell'accaduto. Con intelligente s'intende che un sorvegliante artificiale sia il più possibile indipendente dall'intervento umano e le cui decisioni sia paragonabili a quelle di un operatore, che sta con i propri occhi ad osservare ciò che la videocamera riprende.

Di seguito lo schema rappresentativo del sistema con i suoi tre livelli:



CAPITOLO 2

TECNICHE DI ELABORAZIONE DELLE IMMAGINI

Questo secondo capitolo espone una panoramica esaustiva sul mondo dell'elaborazione delle immagini e riporta una serie di algoritmi, utilizzati per la realizzazione del progetto. Particolare attenzione posta per gli estrattori di feature strutturali.

2.1 ALGORITMI

2.1.1 Trasformata di Hough

Ideata da *Hough* nel 1962 era basata, nella sua prima versione, solo sul riconoscimento di segmenti rettilinei e si applicava solamente ad immagini binarie, ovvero immagini in cui sono presenti due soli livelli, bianco e nero ed in cui l'informazione associata ad un punto è rappresentata unicamente dalla sua posizione.

La trasformata di Hough (HT) è una tecnica che permette di riconoscere configurazioni globali presenti nell'immagine come segmenti, curve od altre forme prestabilite sfruttando la loro proiezione puntiforme in uno spazio (detto *spazio dei parametri* o delle caratteristiche) opportunamente definito.
In questo modo il problema di riconoscere una configurazione che può occupare una gran porzione dell'immagine e avere forma complessa si riduce a quello di riconoscere una configurazione locale, per lo più puntiforme, in un altro spazio. Inoltre essendo una tecnica globale (le decisioni vengono prese nello spazio dei parametri dove l'informazione è globale, piuttosto che nello spazio dell'immagine dove l'informazione è locale).

Quindi definita la curva che si intende cercare nella scena, per ogni punto dell'immagine si calcolano i parametri di tutte le curve che potrebbero passare per quel punto e si incrementano le celle di uno spazio dimensionale (con *n* numero dei parametri) che corrispondono alle varie curve. Si ottiene così una funzione di accumulazione definita nello spazio dei parametri.

Alla fine saranno i *massimi* di questa funzione, ovvero i punti nello spazio dei parametri che hanno accumulato il maggior numero di "voti" a rappresentare le curve che hanno probabilità elevata di essere presente nell'immagine, come se si trattasse di un'ipotesi avvalorata dal maggior numero di conferme sperimentali.
Lo stesso principio può essere applicato ad altre forme geometriche pur di scegliere come variabili trasformate opportuni parametri che definiscono la forma cercata. Ad esempio per i cerchi, occorrerà definire uno spazio di tre parametri (coordinate del centro e raggio, …) i cui punti ne rappresentino uno in particolare.
La trasformazione che porta dal *piano immagine* allo *spazio dei parametri* non è unica ma dipende dalla curva da ricercare e da come questa é parametrizzata. In particolare ogni punto *P* dell'immagine viene trasformato nella curva luogo di tutte quelle combinazioni di valori dei parametri che descrivono curve dello spazio immagine del tipo cercato e contenenti *P*.

Ad esempio se si stanno cercando delle ellissi, ogni punto *P* dell'immagine é trasformato nella curva luogo dei punti dello spazio dei parametri che corrispondono alle ellissi passanti per *P*.
La trasformata di Hough risulta molto robusta in quanto consente di ricercare le curve parametrizzate anche a partire da figure rumorose e dai contorni non continui. Infatti, l'altezza dei picchi nella matrice di accumulazione, dipende in maniera del tutto trascurabile da eventuali lacune presenti nella retta di partenza o dall'esistenza di punti spuri presenti nello spazio immagine.

Nella successione cronologica dell'evoluzione della trasformata, si hanno tre aspetti di riconoscimento :

1. Rilevamento di segmenti rettilinei.
2. Riconoscimento di forme geometriche.
3. Riconoscimento di forme qualsiasi.

Riconoscimento di rette

Sappiamo che é possibile descrivere analiticamente una retta in diverse forme. Consideriamo il punto (x', y') nel piano cartesiano (x , y) e la generica retta di equazione y'=mx'+c passante per il punto (Figura 1).

Al variare di m e c l'equazione rappresenta tutte le possibili rette del piano cartesiano passanti per il punto (x',y').

E' possibile ricavare ora la legge che da un punto (x', y') dello spazio immagine permetta di tracciare la curva nello spazio dei parametri (m,c). Dall'equazione y'=mx'+c si ottiene infatti c=-x'm+y' dove ora x' e y' sono costanti, e m e c variabili. Come si può notare in Figura 2, il fascio di rette passante per il punto P dell'immagine viene quindi rappresentato con una retta nello spazio (m,c).

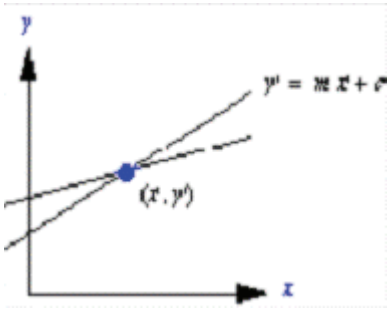


Figura 1: Rette passanti per il punto (x', y')

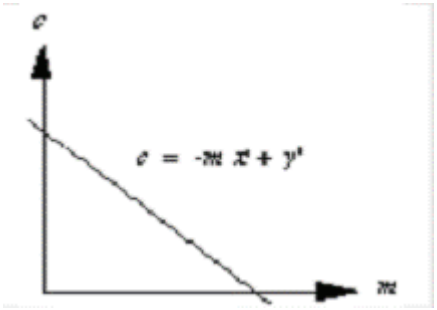


Figura 2: rappresentazione nello spazio (m,c).

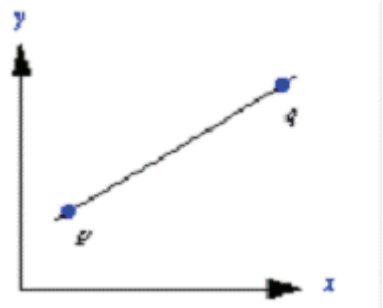


Figura 3

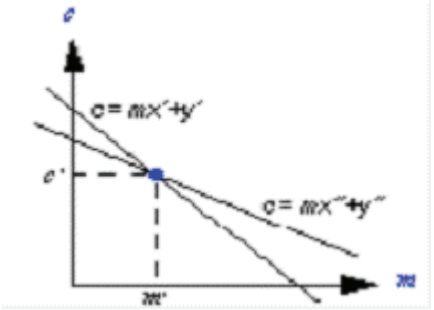


Figura 4

Se consideriamo ora due punti P(x',y') e Q(x",y") nello spazio (x,y) che giacciono sulla stessa retta, per ognuno di essi si ha un corrispondente fascio di rette che nello spazio immagine (m,c) viene rappresentato da una retta.

Si avrà allora che nello spazio dei parametri:

- la retta **c = - mx' + y'** descrive i valori di (m,c) relativi alle rette passanti per P;
- la retta **c = - mx'' + y''** descrive i valori di (m,c) relativi alle rette passanti per Q .

Il punto di intersezione (m',c') nello spazio dei parametri descrive i valori dei parametri (m,c) della retta passante per *p* e *q* nello spazio originale (x,y) ed e' quindi facile risalirvi.

Un problema legato alla prima versione della trasformata è che al variare di (m,c) si possono avere infinite rette.
Mentre per *c* si potrebbe fissare un valore massimo e uno minimo, il parametro *m* cresce smisuratamente, quando la retta diventa verticale (tendendo all'infinito).

Per questo è utile utilizzare una formula alternativa:

$$\rho = x\cos\theta + y\,\mathrm{sen}\,\theta$$

dove ρ è la distanza dall'origine della retta e θ è l'orientazione di ρ rispetto all'asse X.
In tal modo il campo risulta limitato.

Nelle figure che seguono possiamo vedere alcune trasformazioni di rette.

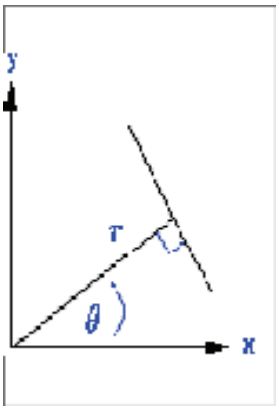


Figura 5

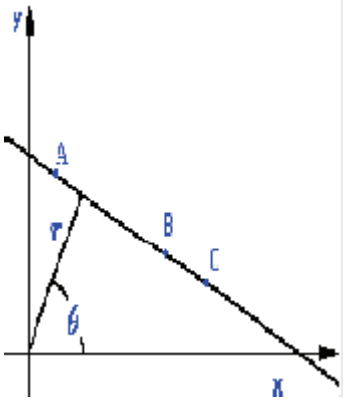


Figura 6

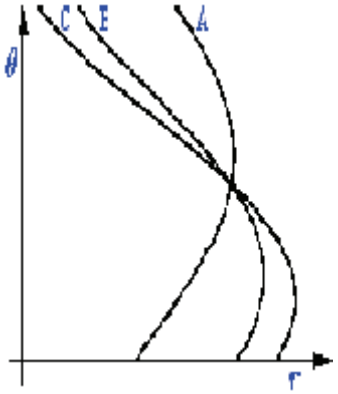


Figura 7

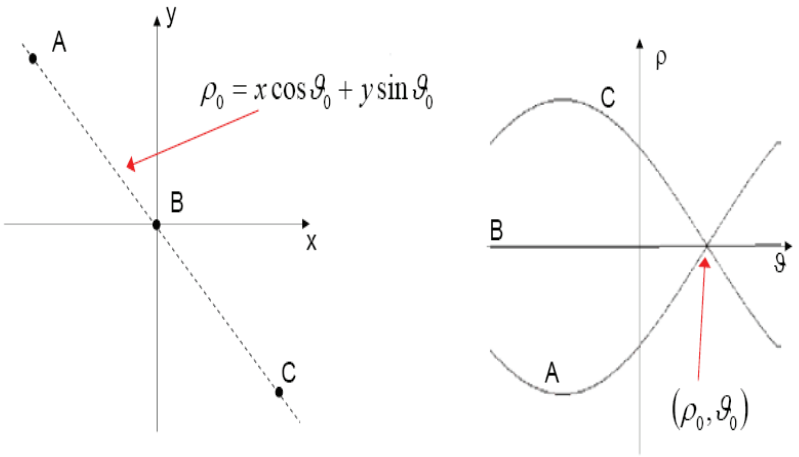


Figura 8: trasformazione di due punti di una retta per trovare la retta stessa.

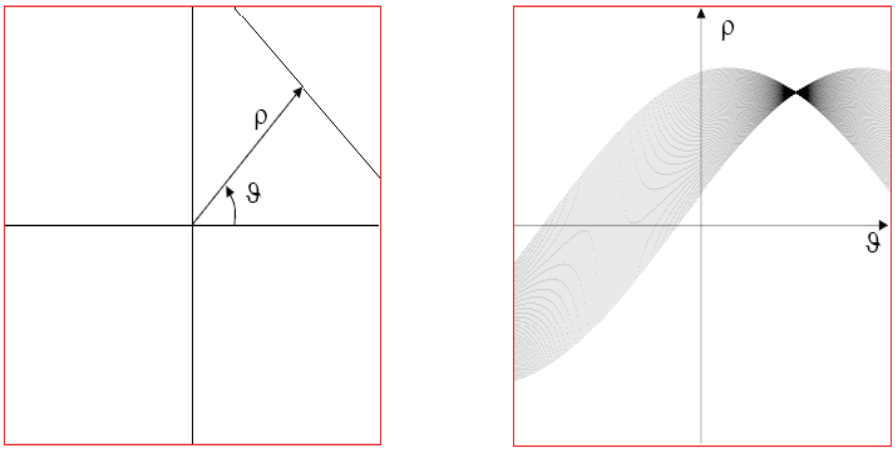


Figura 9: trasformazione dei punti della retta presenti nel riquadro a sinistra.

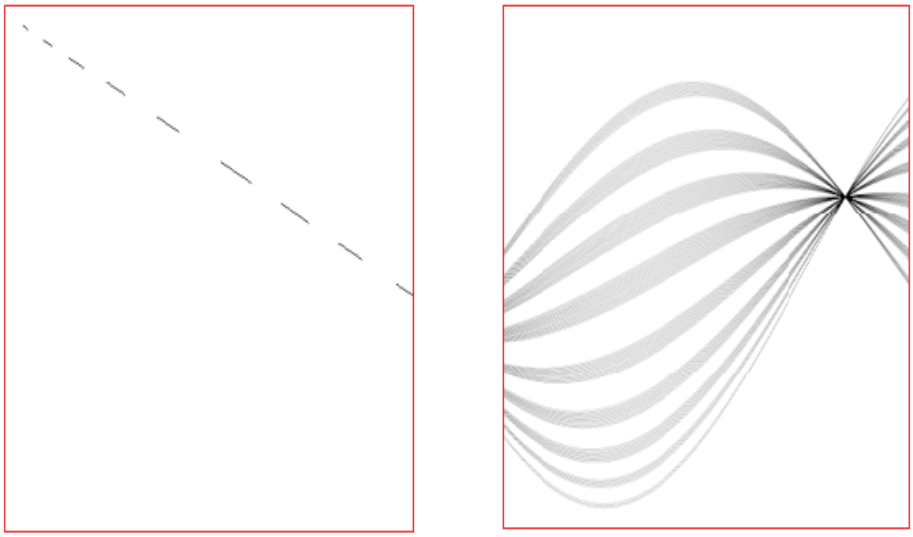


Figura 10: Retta Tratteggiata.

L'idea quindi per estrarre rette da un'immagine è dato dal seguente algoritmo:

- Rappresentare lo spazio (**ρ**,**θ**) attraverso una matrice di accumulazione A.

• Inizializzare tutti gli elementi di A(m,n) a zero.

• Per ciascun edge-pixel (x,y) nell'immagine, e per ogni $\boldsymbol{\theta}_n$, con $1 \leq n \leq N$,

calcolare: $\boldsymbol{\rho}_{(n)} = \mathbf{x} \cos \boldsymbol{\theta}_n + \mathbf{y} \operatorname{sen} \boldsymbol{\theta}_n$

($\boldsymbol{\rho}_{(n)}$ è il valore della retta per (x,y) nella direzione perpendicolare alla retta di direzione $\boldsymbol{\theta}_n$) .

• Determinare l'indice m (valore quantizzato di $\boldsymbol{\rho}_{(n)}$)

• Incrementare A(m,n).

• Ricercare in A(m,n) i punti di massimo (Un punto di massimo in tale matrice corrisponde ad una retta nello spazio immagine originale).

Trasformata di Hough generalizzata

Fino a questo punto è stato presa in considerazione solo la ricerca di rette nell'immagine. Ma per progettate un estrattore il più possibile generalizzato questo sistema di parametri non va più bene. Nel caso in cui non esiste una descrizione della forma da cercare in termini di curva parametrica, una tecnica è basata sulla *Trasformata di Hough Generalizzata* proposta da *Ballard* .

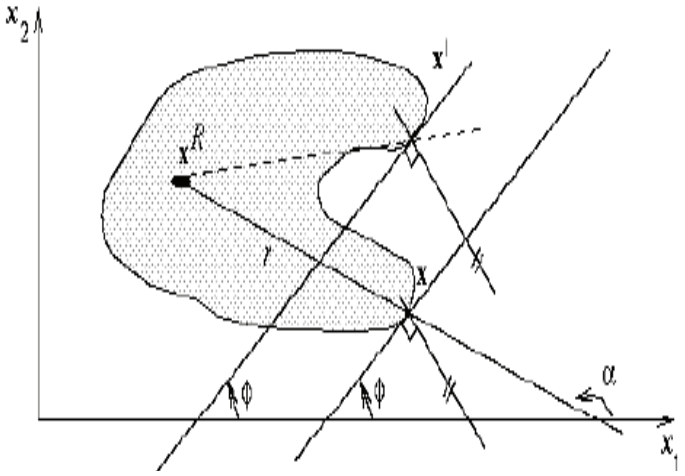


Figura 11: trasformata su una figura generica.

A partire da un punto arbitrario x^R interno alla figura, si calcolano il modulo (r) e la direzione (α) dei segmenti che uniscono il punto di riferimento ai punti di contorno, in aggiunta si tiene conto della direzione ϕ del contorno in ciascun punto.

A partire da queste informazioni, eventualmente ordinate per ϕ si costruisce la *R*-table. Qui di seguito è costruita la R-table

$$
\begin{array}{ll}
\phi_1 & (r_1^1, \alpha_1^1), (r_1^2, \alpha_1^2), \ldots, (r_1^{n_1}, \alpha_1^{n_1}) \\
\phi_2 & (r_2^1, \alpha_2^1), (r_2^2, \alpha_2^2), \ldots, (r_2^{n_2}, \alpha_2^{n_2}) \\
\phi_3 & (r_3^1, \alpha_3^1), (r_3^2, \alpha_3^2), \ldots, (r_3^{n_3}, \alpha_3^{n_3}) \\
\ldots & \ldots \\
\phi_k & (r_k^1, \alpha_k^1), (r_k^2, \alpha_k^2), \ldots, (r_k^{n_k}, \alpha_k^{n_k})
\end{array}
$$

La *R*-table è una rappresentazione della forma, ciascun pixel $x = (x1,x2)$ con direzione del gradiente $\phi(\boldsymbol{x})$ ha come coordinate potenziali del punto di riferimento.

$$\left\{x_1 + r(\ddot{o})\cos[\acute{a}(\ddot{o})], x_2 + r(\ddot{o})\sin[\acute{a}(\ddot{o})]\right\}$$

E queste vanno calcolate per ogni entrata della tavola nella riga corrispondente alla direzione ϕ. Un ulteriore generalizzazione consiste nel considerare cambiamenti di rotazione (t) e scala (*S*) dell'oggetto di riferimento.

Tuttavia si ha lo svantaggio di un notevole onere computazionale, che cresce all'aumentare della complessità delle curve. Infatti, i *k* parametri necessari a descrivere e individuare una curva implicano un carico computazionale pari a *O(Nk)*, indicando con *N* la dimensione dell'immagine.

In ogni modo, i vantaggi insiti in questo metodo riguardano la tolleranza verso eventuali gap esistenti nell'immagine ottenuta da quella originale attraverso un'operazione di edge detection. Da questo, inoltre, ne consegue anche una maggiore robustezza al rumore.
Solitamente, l'edge detection avviene utilizzando il passaggio per lo zero del laplaciano, mentre la successiva applicazione della trasformata di Hough fornisce i parametri della curva che meglio approssima i contorni estratti. Tutto ciò consente di ottenere una curva continua per descrivere l'oggetto cercato. In altri termini, mentre l'edge detection localizza solamente gli oggetti da estrarre, la trasformata di Hough interpreta cosa gli oggetti estratti
rappresentino.

La versione generalizzata della trasformata di Hough viene utilizzano nei casi in cui gli oggetti cercati presentano una forma difficilmente rappresentabile da una semplice equazione analitica. In tali situazioni, pertanto, in luogo della rappresentazione parametrica della curva si fa uso di una tabella di look-up per definire la relazione tra le posizioni dei contorni e le orientazioni dei parametri di Hough.

L’uso di tali tabelle rende possibile l’individuazione di una serie di massimi locali di somiglianza con la curva di riferimento in corrispondenza dei quali sono localizzati gli oggetti cercati.

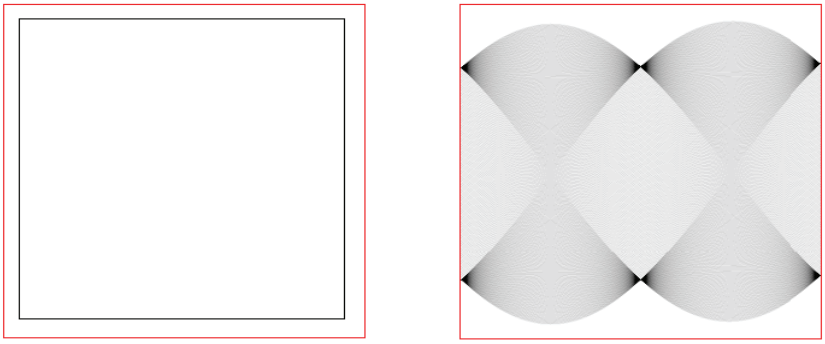


Figura 12: trasformata di hough su figura piana.

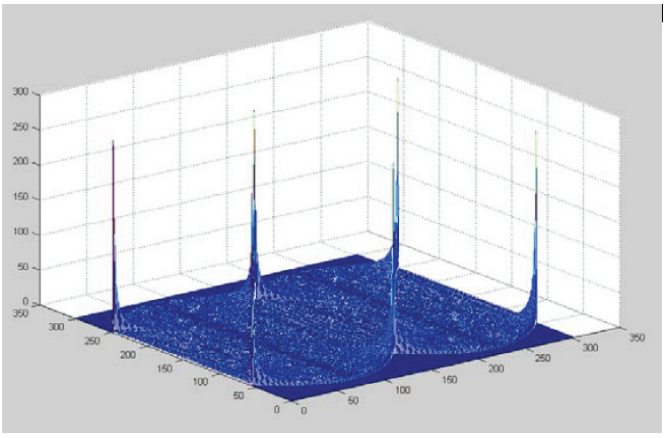


Figura 13: matrice di accumulazione per figura piana.

Figura 14: riprese effettuate presso un museo di Firenze.

Figura 15: applicazione della trasformata di hough per la ricerca di rette sul quadro di figura 14.

Riconoscimento dei picchi a massima invarianza

Si è utilizzata la trasformata di Hough per ricercare all'interno della scena gli edge a *massima invarianza* . Si è scelto questo tipo di feature perché riesce in modo semplice ma accurato a determinare i rapidi cambiamenti nella scena, eliminando tutta la ridondanza tipica degli edge che renderebbe computazionalmente molto pesante la ricerca delle variazioni all'interno dei filmati, concentrando così l'informazione.

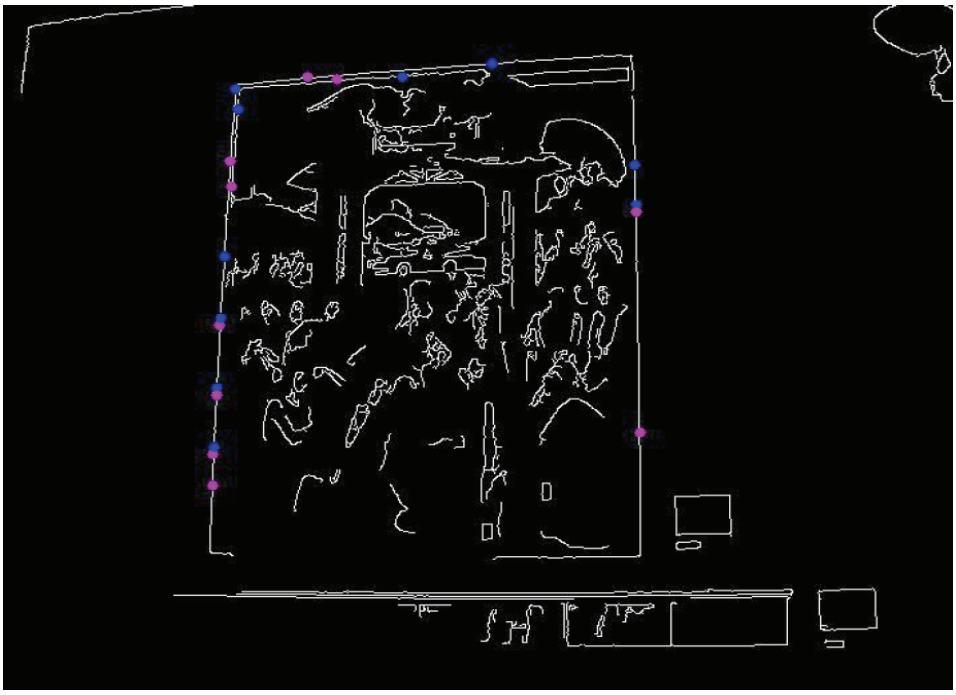


Figura 16: picchi a massima invarianza sul medesimo quadro.

2.1.2 Filtro di Canny

A livello computazionale un'immagine non è altro che una matrice di punti (chiamati *pixel*). La dimensione di un'immagine è il numero di pixel (in orizzontale verticale che compongono l'immagine). Sia *Im* un'immagine in bianco e nero, l'elemento *Im(x,y)* rappresenta il valore di luminosità del punto *x,y*. I valori di luminosità variano da 0 a *maxLum* (di solito 255). Il valore 0 corrisponde ad un punto nero il valore maxLum corrisponde ad un pixel bianco.

Adesso descriverò la sogliatura dell'immagine.

Data un'immagine di dimensione MxN i cui punti sono indicati con

$$P(x,y) \qquad \text{dove } 0 \le x \le M,\ 0 \le y \le N$$

si supponga che il valore di ogni elemento P(x,y) vari da 0 a 255, e sia **S0** il valore di soglia. L'immagine a cui viene applicata una soglia è quella i cui elementi sono definiti nel seguente modo:

$$Ps(x,y) = \begin{cases} 0 \text{ se } P(x,y) \le S0 \\ 1 \text{ se } P(x,y) > S0 \end{cases}$$

Lo scopo di applicare una soglia un'immagine è quello di eliminare tutte le parti che non interessano ai fini dell'elaborazione.

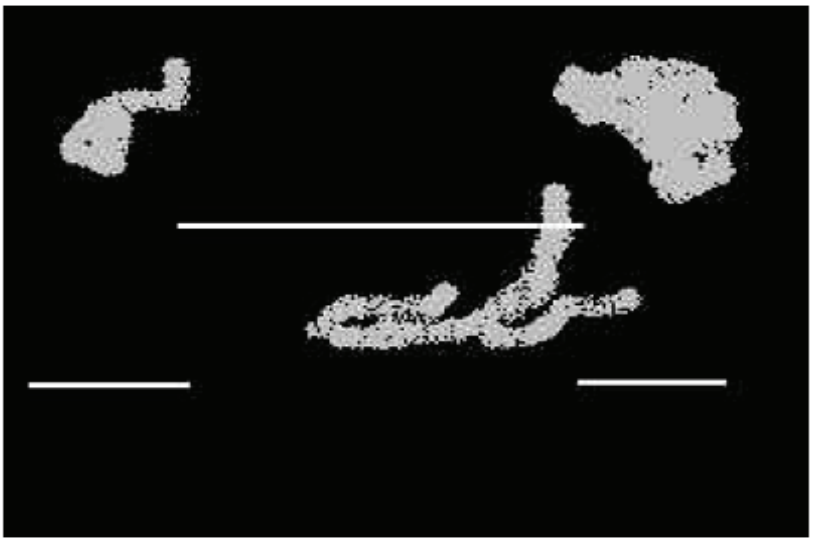


Figura 17: immagine linea laser.

La figura 17 mostra l'immagine di una linea laser, si noti come oltre alla linea chiara del laser siano presenti delle macchie chiare di intensità più bassa del laser ma più alta dello sfondo nero. Queste macchie potrebbero essere dovute per esempio a dei riflessi sull'oggetto o sull'ambiente circostante. Si supponga che il valore di luminosità dei pixel neri di sfondo sia pari a 0, quello del laser sia pari a 240 e quello delle macchie grigie sia pari a 140. Se come soglia si sceglie il valore S0 = 200 si ha che l'immagine sogliata risulta come nella figura 18.

Figura 18 : immagine linea laser senza macchie.

Si noti come sono state eliminate le macchie scure.

Se si fosse scelta una soglia paria a S0 = 100, l'immagine soglia sarebbe stata come da figura 19:

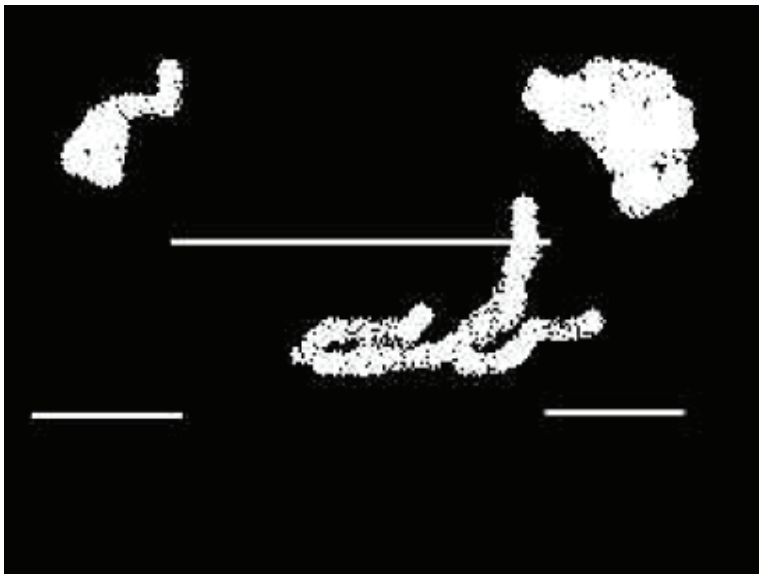


Figura 19

In questo caso si nota come l'immagine sia peggiorata in quanto le macchie grigie hanno ora lo stesso colore del laser. Un'immagine di questo tipo è difficilmente elaborabile.

La scelta del giusto valore della soglia è la principale difficoltà di questa tecnica di filtraggio. Essa risulta inoltre inutile in tutte quelle immagini nelle quali gli elementi di disturbo dell'immagine (es. macchie grigie) hanno lo stesso livello di luminosità degli elementi che si vogliono analizzare. Una maschera di convoluzione è un insieme bidimensionale di coefficienti i cui valori sono scelti per mettere in risalto una particolare caratteristica dell'immagine.

Si prenda come esempio una maschera di dimensione 3x3.

U1	U2	U3

(x-1,y-1)	(x,y-1)	(x+1,y-1)
U4 (x-1,y)	U5 (x,y)	U6 (x+1,y)
U7 (x-1,y+1)	U8 (x,y+1)	U9 (x+1,y+1)

Si consideri un'immagine I, e sia I(x,y) il valore del punto nella coordinata x,y. L'immagine filtrata tramite questa maschera si ricava dalla seguente relazione:

$$If(x,y) = U1*I(x-1,y-1) + U2*I(x,y-1) + U3*I(x+1,y-1) +$$
$$U4*I(x-1,y) + U5*I(x,y) + U6*I(x+1,y) +$$
$$U7*I(x-1,y+1) + U8*I(x,y+1) + U9*I(x+1,y+1)$$

Si noti che la precedente relazione non è applicabile ai bordi dell'immagine in quanto la maschera esce dall'immagine stessa. Per questo motivo le immagini filtrate con maschere di questo tipo sono più piccole di quelle di partenza.

Quando analizziamo un'immagine dobbiamo sempre tenere presente che essa rappresenta spesso una proiezione bidimensionale di una scena tridimensionale; essa è passata attraverso la rivelazione da parte di un sensore e il campionamento con conseguente discretizzazione nello spazio e nei valori. Ciò che generalmente si intende per contorno è il confine di passaggio da una zona con certe caratteristiche ad un'altra con caratteristiche apprezzabilmente diverse, inoltre si intende che questo passaggio debba avvenire in modo abbastanza brusco, ossia in uno spazio ristretto.

In questo senso, ciò che si cerca d'identificare e localizzare sono le discontinuità nelle caratteristiche locali e puntuali delle superfici visibili nella scena. Tali discontinuità segnano la separazione fra regioni omogenee, dove il termine "omogeneo" può significare "dello stesso colore", "dello stesso materiale", "con la stessa trama" e così via. I problemi più comuni per questo tipo di segmentazione (*edge detection*), causati in generale dalla presenza di rumore o da insufficienti informazioni sull'immagine, sono l'estrazione di bordi, dove non ci dovrebbero essere e viceversa, causando un'influenza negativa sui risultati finali.

L'operatore di Canny è stato definito in base ad un modello matematico dei punti di bordo e del rumore, alla specifica dei criteri di prestazione da ottenere ed infine al filtro lineare che risponde meglio a tutte queste caratteristiche.
Per progettare *detector* per *edge* arbitrari, John Canny ha sviluppato una formulazione matematica per i seguenti due criteri:

- ottenimento di una percentuale di errore bassa, ovvero estrazione di bordi da rumore spurio molto ridotta.

- I punti che compongono i bordi degli oggetti presenti nell'immagine devono essere ben localizzati. In altre parole, la distanza tra i punti "marcati" dall'operatore e il "centro" del vero contorno deve essere minima.

I tre criteri proposti da Canny per valutare le prestazioni degli operatori di estrazione di contorni sono i seguenti:

- *Good Detection:* Deve esserci una bassa probabilità di errore nel riconoscere un vero punto di contorno e una bassa probabilità di riconoscere punti non appartenenti ai bordi (falsi contorni). Siccome entrambe le probabilità sono funzioni monotone decrescenti del rapporto segnale-rumore, questo criterio corrisponde alla sua massimizzazione. In assenza di ulteriori informazioni il rumore è considerato bianco e gaussiano.

- *Good Localization:* I punti marcati come contorni dall'operatore devono essere il più vicino possibile al centro del vero *edge*.

- *One responce to single edge:* Una risposta unica ad un singolo *edge*. Questo è implicitamente insito nel primo criterio e indica che quando ci sono due risposte allo stesso *edge*, uno di essi deve essere considerato falso. Questo criterio deve essere esplicitamente inserito

nella formulazione matematica del primo, dato che quest'ultimo non contempla la risposta multipla.

I bordi di immagini di intensità possono essere modellati in accordo con i loro profili di intensità. Per molti scopi pratici sono sufficienti pochi modelli per considerare tutti gli *edge* interessanti.

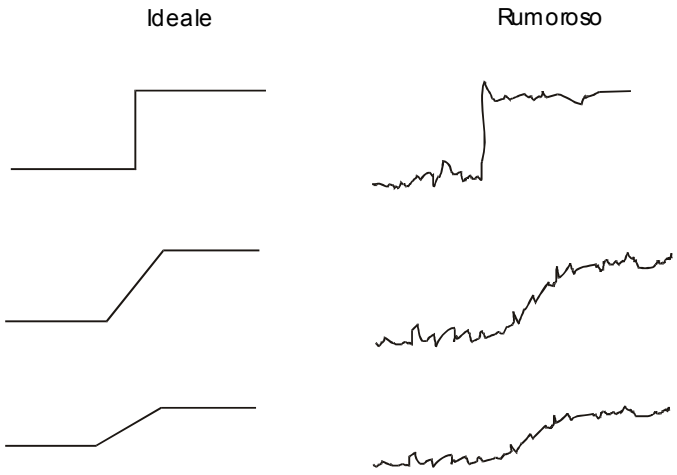


Figura 20: segnali monodimensionali.

Tali modelli vengono proposti come segnali monodimensionali, anche se ovviamente sono da considerarsi delle sezioni di immagini bidimensionali lungo una linea di orientazione arbitraria (non necessariamente una riga o colonna).

Gli *edge* a gradino sono probabilmente il tipo più comune in immagini di intensità. Essi esistono tipicamente sui contorni di regioni con differente livello di grigio. Se la transizione avviene lungo una serie di pixel, e non su uno solo, allora si parla di bordi a rampa (*ramp edges*). I bordi ripidi (*ridge edges*) sono generati da sottili linee e possono essere modellati da due bordi a gradino. I bordi "a punta" (*roof edges*) sono relativamente rari, ma possono apparire lungo l'intersezione di superfici. Da osservare

che i bordi a gradino e i bordi ripidi corrispondono a rapide variazioni nei valori d'intensità, mentre i *roof edges* corrispondono a rapide variazioni delle loro derivate prime.

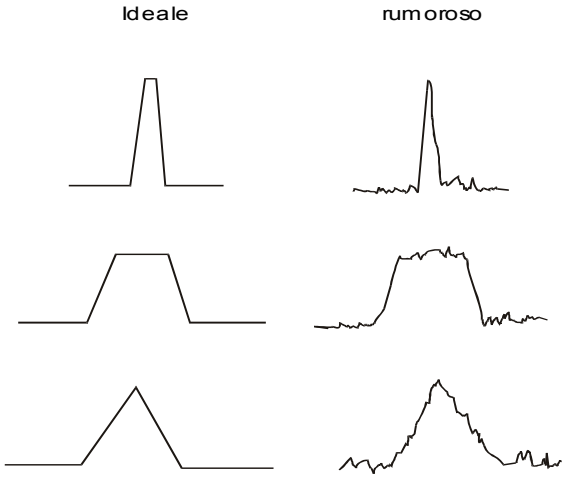


Figura 21: bordi e rispettive variazioni d'intensità.

L'operatore che realizza un'approssimazione molto buona dell'estrattore di bordi a gradino è la derivata prima di una Gaussiana.

Considerata la funzione Gaussiana G(x) come :

$$G(x) = e^{\frac{-x^2}{2\sigma^2}}$$

La risposta impulsiva del filtro ottenuto con la derivata prima G'(x) è:

$$G^{'}(x) = f(x) = -\frac{x}{\chi^2} e^{\frac{-x^2}{2\sigma^2}}$$

Con $\chi^2=\sigma_s$, dove σ_s, la deviazione standard, ha la seguente espressione :

$$\sigma s \quad = \sqrt{\int_{-W}^{+W} f^{''}(x)dx}$$

La chiave di generalizzazione per segnali 2-D è il considerare che il trattamento 1-D è applicato alla direzione della normale al contorno. Siccome tale normale non è conosciuta a priori, bisognerebbe calcolare per ogni pixel dell'immagine, la risposta del filtro direzionale 1-D per tutte le possibili direzioni. Questo laborioso metodo può essere notevolmente semplificato se si usa un filtro circolare Gaussiano, il quale opera una riduzione del rumore (gaussiano, appunto) e stima tutte le derivate direzionali del gradiente ottenuto da un preventivo calcolo. Questo metodo è meno accurato rispetto a quello che applica filtri direzionali, ma risulta adeguato per molti casi pratici. Si supponga di convolvere l'immagine con un operatore Gn, che è la derivata prima di una Gaussiana bidimensionale G, in una qualche direzione **n** cioè:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

e

$$G\bar{n} = \frac{\partial G}{\partial \bar{n}} = \bar{n} \bullet \nabla G$$

Idealmente, $\bar{n}$ deve essere orientata normalmente alla direzione dell'*edge* da determinare e comunque questa direzione non è nota a priori. Si può fornire una buona stima dalla direzione del gradiente "smoothed" (ossia filtrato):

$$\bar{n} = \frac{\nabla(G * I)}{\left\|\nabla(G * I)\right\|}$$

Un punto di *edge* è quindi definito punto di massimo locale (in direzione n) dell'operatore Gn applicato all'immagine I.

$$s(i, j) = \left\|\nabla(G * I)\right\| = \left\|Gn * I\right\|$$

A causa dell'associatività della convoluzione :

$$\nabla(G * I) = \nabla G * I$$

Quindi la prima fase dell'algoritmo, di esaltazione dei bordi per l'immagine I, consiste nei seguenti passi:

- L'ingresso è l'immagine di luminosità I corrotta da rumore. Sia G la funzione Gaussiana con valore medio nullo e deviazione standard σ.
- Applicazione del filtro gaussiano all'immagine I ottenendo J=I*G.
- Per ogni pixel (i,j):

  (a) calcolo delle componenti del gradiente Jx e Jy

  (b) Stima della forza del gradiente:

$$es(i,j) = \sqrt{J^2x(i,j) + J^2y(i,j)} \tag{4.17}$$

$$e0(i,j) = arc\tan\frac{Jy}{Jx}$$

  (c) Stima dell'orientazione della normale all'edge:

(4.18)

L'uscita è l'immagine dell'intensità del bordo *Es*, formata dai valori *es(i,j)* e dall'immagine di orientazione *E0*, formata dai valori *e0(i,j)*. La funzione gaussiana in due dimensioni definita in precedenza è caratterizzata dalla deviazione standard σ, che è il principale parametro di progetto. Aumentando o diminuendo σ, si ha un aumento o una relativa diminuzione del *kernel* "significativo" del filtro e di conseguenza l'effetto risultante sull'immagine *output* della convoluzione. Dato che la convoluzione con un nucleo gaussiano è un'operazione che coinvolge il pixel centrale ma anche quelli del suo intorno (la cui dimensione dipende da σ), si ha che l'*output* dell'operazione di convoluzione dell'immagine con questo filtro (filtraggio gaussiano) è un'immagine

"*smoothed*" ovvero che i picchi di luminosità sono appiattiti. In pratica il *kernel* gaussiano è una campana, la cui ampiezza è direttamente proporzionale alla deviazione standard. La realizzazione della convoluzione con il nucleo gaussiano è particolarmente semplice, grazie alla separabilità del nucleo stesso.

$$J = I * G = \sum_{h=\frac{-m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} G(h,k) I(i-h, j-k) =$$

$$= \sum_{h=\frac{-m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} e^{-\frac{h^2+k^2}{2\sigma^2}} I(i-h, j-k) = \sum_{h=\frac{-m}{2}}^{\frac{m}{2}} e^{-\frac{h^2}{2\sigma^2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} e^{-\frac{k^2}{2\sigma^2}} I(i-h, j-k)$$

Dove I è l'immagine corrotta da rumore di dimensione NxM, m è un numero dispari più piccolo di N ed M, e G è il nucleo del filtro gaussiano, caratterizzato dalla maschera mxm. Ciò significa che effettuare una convoluzione di un'immagine I con un filtro gaussiano è come effettuare prima la convoluzione con le sue righe poi con le sue colonne utilizzando una Gaussiana monodimensionale avente la stessa deviazione standard σ. Il vantaggio è il tempo di computazione che aumenta linearmente con la dimensione della maschera e non in maniera quadratica. L'algoritmo per operare la convoluzione di un'immagine I con un nucleo gaussiano bidimensionale G, di dimensione m x m con σ = σG è descritto di seguito:

- Costruzione di una maschera gaussiana monodimensionale g, di lunghezza m, con σg = σG.
- Convolvere ogni riga di I con g, ottenendo una nuova immagine Ir.
- Convolvere ogni colonna di I con g.

La relazione tra σ e la dimensione della maschera *w* (tipicamente un numero dispari), può essere ottenuta imponendo che *w* sottenda quasi tutta l'area sotto la campana gaussiana. Una scelta adeguata, quindi può essere $w = 5\,\sigma$, la quale sottende il 98,76% dell'area. La creazione del nucleo gaussiano è stata implementata dalla funzione *make_gaussian_kernel()*.

Il passo successivo dell'algoritmo di Canny è calcolare il gradiente dell'immagine risultato del filtraggio gaussiano passa basso precedente $J(x,y) = G * I$:

$$E(x,y) = \nabla J(x,y)$$

Il gradiente di un'immagine può essere calcolato in termini delle derivate lungo i due assi ortogonali in accordo con la seguente equazione :

$$E(x,y) = \frac{\partial J(x,y)}{\partial x}\bar{i} + \frac{\partial J(x,y)}{\partial y}\bar{j}$$

Dove i e j sono i versori degli assi cartesiani.

La condizione da soddisfare per garantire il calcolo del gradiente efficientemente, è di scegliere una buona approssimazione discreta delle due derivate. Il metodo più noto ed utilizzato per implementare le derivate, è come accennato sopra, quello di calcolare la differenza di pixel lungo le direzioni delle righe e delle colonne dell'immagine.

Le derivate di riga e di colonna sono state calcolate attraverso gli operatori di convoluzione sono:

$$El = J(l,c) * Hl(l,c)$$

$$Ec = J(l,c) * Hc(l,c)$$

Dove Hl(l,c) e Hc(l,c) sono matrici risposta impulsiva di riga e di colonna rispettivamente, definite come sotto :

$$Hc(l,c) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad Hl(l,c) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

La successiva fase dell'algoritmo di Canny, è denominata *non-maximal suppression*, e consiste nell'assottigliamento dei bordi fino a renderli spessi un solo pixel, mediante l'eliminazione di picchi situati nell'intorno dei massimi locali.

2.1.3 Mediana

La mediana è il valore (o l'insieme di valori) di una distribuzione X per cui la frequenza cumulata vale 0,5; pertanto bipartisce la distribuzione. Usualmente si indica la Mediana con Me. Se le modalità sono raggruppate in classi non si definisce un valore univoco, ma una classe mediana Xi − Xi + 1. La determinazione di tale classe avviene considerando le frequenze cumulate, indicando con Fi la generica frequenza cumulata dell'osservazione i-esima sarà: Fi + 1 > 0,5 e Fi < 0,5. Pur essendo corretto considerare un qualsiasi elemento dell'intervallo Xi − Xi + 1 un valore mediano si è soliti procedere, al fine di avere una misura unica del valore, a un'approssimazione della mediana con la seguente formula:

$$Me = X_i + (X_{i+1} - X_i)\frac{0,5 - F_i}{F_{i+1} - F_i}$$

La mediana è un indice di posizione e rientra nell'insieme delle statistiche di ordine.
Una proprietà della mediana è di rendere minima la somma dei valori assoluti degli scarti delle Xi da un generico valore.
In Statistica descrittiva la mediana è il valore che occupa la posizione centrale in un insieme ordinato di dati.
Per calcolare la mediana di *n* dati:

- Si ordinano gli *n* di dati in ordine crescente o decrescente;
- Se il numero di dati è dispari la mediana corrisponde al valore centrale, ovvero al valore che occupa la posizione (n + 1) / 2.
- Se il numero *n* di dati è pari, la mediana è stimata utilizzando i due valori che occupano le posizioni (n / 2) e (n / 2 + 1) (generalmente si sceglie la loro media aritmetica).

Ordinamento Bubble Sort

Per quanto riguarda l'ordinamento degli *n* dati, noi abbiamo ordinato le *n* immagini tramite Bubble Sort.

Il bubble sort o bubblesort (letteralmente: ordinamento a bollicine) è un semplice ed abbastanza efficiente algoritmo di ordinamento particolarmente indicato per l'ordinamento di array. Ha una complessità computazionale (misurata in termini di numero di confronti e assegnamenti) $O(n^2)$, superiore per esempio a quella del quicksort. Tuttavia è piuttosto noto e utilizzato (sia in ambito didattico che da parte di programmatori professionisti) in virtù della sua semplicità. Come tutti gli algoritmi di ordinamento, può essere usato per ordinare dati di un qualsiasi tipo su cui sia definita una relazione d'ordine. Il nome dell'algoritmo è dovuto al fatto che, durante l'applicazione del procedimento, i valori vengono spostati all'interno dell'array con una dinamica che ricorda il movimento delle bollicine in un bicchiere di champagne. In particolare, alcuni elementi attraversano l'array velocemente (come bollicine che emergono dal fondo del bicchiere), altri più lentamente (a differenza di quanto avviene nel caso del bicchiere di champagne, tuttavia, alcuni elementi salgono ma altri scendono).

Il bubblesort è un algoritmo iterativo, ovvero basato sulla ripetizione di un procedimento fondamentale. La singola iterazione dell'algoritmo prevede che gli elementi dell'array siano confrontati a due a due, procedendo in un verso stabilito (se si scandisca l'array a partire dall'inizio in avanti, o a partire dal fondo all'indietro, è irrilevante; nel seguito ipotizzeremo che lo si scandisca partendo dall'inizio). Per esempio, saranno confrontati il primo e il secondo elemento, poi il secondo e il terzo, poi il terzo e il quarto, e così via fino al confronto fra penultimo e ultimo elemento. Per ogni confronto, se i due elementi confrontati non sono ordinati, essi vengono scambiati. Durante ogni iterazione, almeno un valore viene spostato rapidamente fino a raggiungere la sua collocazione definitiva; in particolare, alla prima iterazione il numero più grande raggiunge l'ultima posizione dell'array. Il motivo è semplice, e si può illustrare con un esempio. Supponiamo che l'array sia inizialmente disposto come segue:

15 6 4 10 11 2

Inizialmente 15 viene confrontato con 6, ed essendo 15>6, i due numeri vengono scambiati:

6 15 4 10 11 2

A questo punto il 15 viene confrontato col 4, e nuovamente scambiato:

6 4 15 10 11 2

Non è difficile osservare che, essendo 15 il numero massimo, ogni successivo confronto porterà a uno scambio e a un nuovo spostamento di 15, che terminerà nell'ultima cella dell'array. Per motivi analoghi, alla seconda iterazione è garantito che il secondo numero più grande raggiungerà la sua collocazione definitiva nella penultima cella dell'array, e via dicendo. Ne conseguono due considerazioni:

- Se i numeri sono in tutto N, dopo N-1 iterazioni si avrà la garanzia che l'array sia ordinato;
- Alla iterazione i-esima, le ultime i-1 celle dell'array ospitano i loro valori definitivi, per cui la sequenza di confronti può essere terminata col confronto dei valori alle posizioni N-1-i e N-i.

Ovviamente, ad ogni iterazione può accadere che più numeri vengano spostati; per cui, oltre a portare il numero più grande in fondo all'array, ogni singola iterazione può contribuire anche a un riordinamento parziale degli altri valori. Anche per questo motivo, può accadere (normalmente accade) che l'array risulti ordinato prima che si sia raggiunta la N-1esima iterazione.
Il bubble sort effettua all'incirca $N^2/ 2$ confronti ed $N^2/ 2$ scambi sia in media che nel caso peggiore. Si è deciso di utilizzare questo algoritmo come già detto per la sua semplicità. Anche se non è il migliore come prestazioni, è adatto al problema poichè si ha la necessità di ordinare array di piccole dimensioni da un minio di 5 ad un massimo di 10 o 15 elementi.

2.1.4 Optical Flow

Data una sequenza di immagini è possibile ricavare una grande quantità di informazioni spesso in maniera più semplice che non usando un'immagine "statica", come ad esempio rilevare un oggetto che si è spostato o calcolarne la posizione spaziale.
Scopo di questa trattazione è quello di rilevare il movimento degli oggetti in una scena e per fare questo ci si serve di una successione di immagini.
Diverse tecniche sono state implementate per risolvere questo genere di problema e quella seguita da noi si basa sull'*optical flow*.

Campo del moto

Prima di introdurre il procedimento vero e proprio è opportuno capire come un oggetto in movimento appare sul piano di un' immagine.
Quando un oggetto si sposta di fronte ad una videocamera vi è un corrispondente cambiamento nel piano dell'immagine.

Infatti dato il punto P_0 nello spazio e la sua relativa proiezione P_i, possiamo ricavare, data la velocità V_0 dell'oggetto, il vettore V_i relativo allo spostamento del punto P_i nell'immagine come si può vedere nella figura 22.

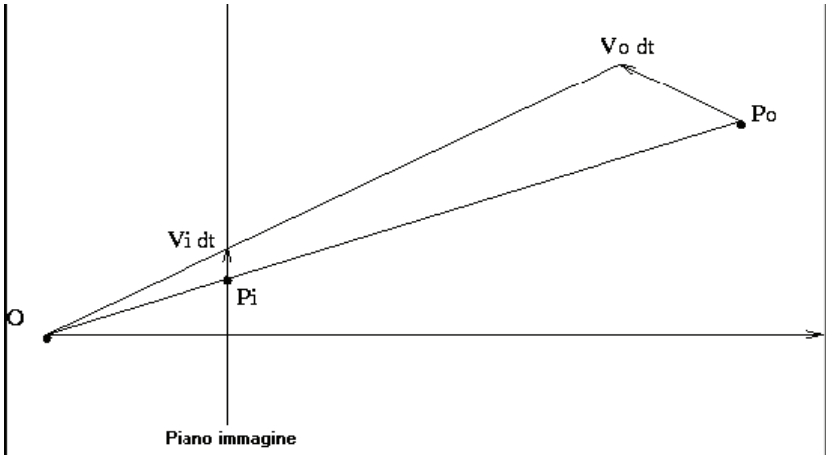


Figura 22

Dato un corpo rigido in movimento è quindi possibile costruire un "*campo di spostamenti*" determinato dall'insieme dei vettori spostamento V_i (vedi figura 23).

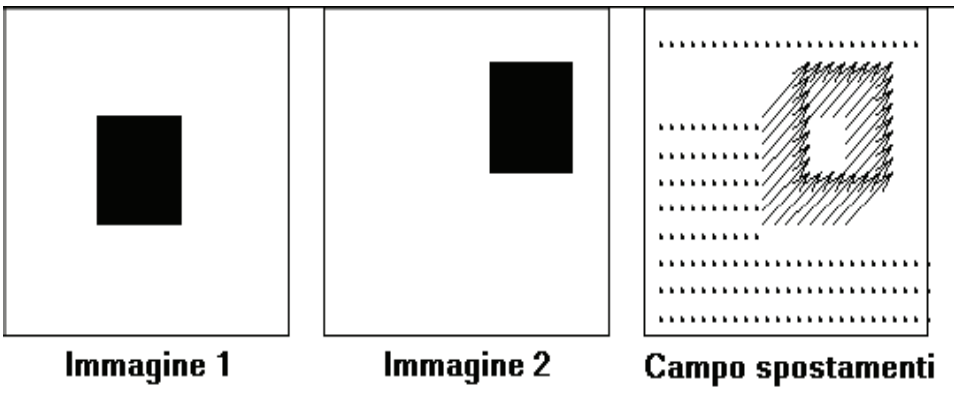


Figura 23

Descrizione Optical flow

L'optical flow è una tecnica che permette di determinare il campo degli spostamenti basandosi non tanto sullo spostamento dell'oggetto bensì sulla sua intensità luminosa.
Nella maggior parte dei casi il campo degli spostamenti coincide con il campo determinato mediante l'optical flow stesso ma questo non è sempre vero.
Questa incongruenza è dovuta al fatto che tale tecnica è in grado di misurare esclusivamente la componente del movimento nella direzione del gradiente dell'intensità luminosa come illustra l'immagine seguente (figura 24).

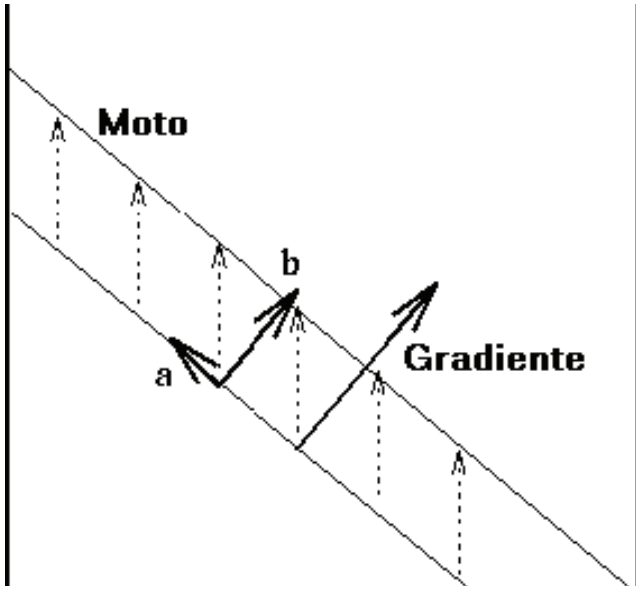


Figura 24

Denotiamo con I(x,y,t) l'intensità luminosa; questa è una funzione di tre variabile che denota la variazione spazio temporale del nostro segnale. Per poter vedere come cambia nel tempo bisogna effettuare la sua derivata rispetto t:

$$\frac{dI}{dt} = \frac{\delta I}{\delta x}\frac{dx}{dt} + \frac{\delta I}{\delta y}\frac{dy}{dt} + \frac{\delta I}{\delta t}$$

Assumendo che l'intensità luminosa di ogni punto dell'oggetto è invariante nel tempo (per esempio le ombre e l'illuminazione non cambiano durante lo spostamento infinitesimo) otteniamo:

$$\frac{dI}{dt} = 0$$

e questo implica che $\mathbf{I_x u + I_y v + I_t = 0}$, dove le derivate parziali di I sono indicate dai pedici e dove u e v sono le componenti x e y del vettore optical flow.
Questa ultima equazione è chiamata "*optical flow constraint equation*" poiché impone dei vincoli sulle componenti u e v.
Può anche essere riscritta nel modo seguente :

$$(I_x, I_y) \cdot (u, v) = -I_t$$

Per ricavare le componenti della velocità di spostamento nella direzione del gradiente dell'intensità luminosa si usa:

$$|| F || = \frac{-I_t}{\sqrt{I_x^2 + I_y^2}}$$

dove F e' la norma del vettore (u,v).

Come si può vedere non è un prodotto scalare perfetto poiché non è presente il valore dell'angolo ma ad ogni modo è sufficiente per indicarci la direzione di spostamento.

Metodo Di Lucas Kanade

I metodi utilizzati nell'optical flow cercano di calcolare il movimento in due frame presi al tempo t e al tempo t + δt su ciascuna posizione dei pixel.
Dato un pixel (x,y,z,t) con Intensità I(x,y,z,t), questo sarà spostato di un fattore δx, δy, δz e δt al frame successivo.

Possiamo quindi formulare la seguente equazione per il vincolo:

$$I(x,y,z,t) = I(x + \delta x, y + \delta y, z + \delta z, t + \delta t)$$

Assumendo che il movimento sia abbastanza piccolo, possiamo effettuare lo sviluppo in serie di Taylor della precedente equazione.

$$I(x+\delta x,y+\delta y,t+\delta t)=I(x,y,z,t)+\frac{\partial I}{\partial x}\delta x+\frac{\partial I}{\partial y}\delta y+\frac{\partial I}{\partial z}\delta z+\frac{\partial I}{\partial t}\delta t+H.O.T$$

Dove H.O.T sono i termini di ordine più alto che sono piccoli abbastanza e possono essere ignorati.

$$\frac{\delta I}{\delta x}\frac{\delta x}{\delta t}+\frac{\delta I}{\delta y}\frac{\delta y}{\delta t}+\frac{\delta I}{\delta z}\frac{\delta z}{\delta t}+\frac{\delta I}{\delta t}\frac{\delta t}{\delta t}=0$$

Che diventa:

$$\frac{\delta I}{\delta x}V_x+\frac{\delta I}{\delta y}V_y+\frac{\delta I}{\delta z}V_z+\frac{\delta I}{\delta t}=0$$

Dove V_x, V_y, V_z sono le componenti x,y,z della velocità (*optical flow)* di I(x,y,z,t) e

$$\frac{\delta I}{\delta x}\cdot\frac{\delta I}{\delta y}\cdot\frac{\delta I}{\delta z}\frac{\delta I}{\delta t}$$

sono le derivate dell'immagine a (x,y,z,t) nelle corrispondenti direzioni.

Scriveremo I_x, I_y, I_z e I_t al posto delle derivate, il tutto diventa:

$$I_x V_x + I_y V_y + I_z V_z = -I_t$$

Oppure

$$\nabla I \cdot \overline{V} = -I_t$$

Questa è un'equazione a tre incognite e non può essere risolta cosi com'è. Questo è conosciuto come *aperture problem* dell'algoritmo di optical flow. Per trovare l'optical flow abbiamo bisogno di altre equazioni date da vincoli aggiuntivi. La soluzione cosi come è stata data da Lucas e Kanade è un metodo non iterativo che assume un flusso localmente costante.

Assumendo che il flusso (V_x, V_y, V_z) sia costante in una piccola finestra di piccole dimensioni *m x m x m* dove m > 1 e centrato in (x,y,z) e numerando i pixel da 1 a n otteniamo un set di equazioni:

$$\begin{aligned}
&I_{x1} V_x + I_{y1} V_y + I_{z1} V_z = -I_{t1} \\
&I_{x2} V_x + I_{y2} V_y + I_{z2} V_z = -I_{t2} \\
&\vdots \\
&I_{xn} V_x + I_{yn} V_y + I_{zn} V_z = -I_{tn}
\end{aligned}$$

Con queste si hanno più di tre equazioni e tre incognite e quindi un sistema sovra-determinato:

$$\begin{bmatrix} I_{x1} & I_{y1} & I_{z1} \\ I_{x2} & I_{y2} & I_{z2} \\ \vdots & \vdots & \vdots \\ I_{xn} & I_{yn} & I_{zn} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}$$

Oppure

$$A\bar{v} = -b$$

Per risolvere un sistema cosi fatto usiamo il metodo dei minimi quadrati

$$A^T A\bar{v} = A^T(-b)$$

Oppure

$$\bar{v} = (A^T A)^{-1} A^T(-b)$$

In forma equivalente:

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}^{-1} \left(-A^T I_t\right)$$

In questo modo l'optical flow può essere trovato calcolando le derivate dell'immagine nelle quattro dimensioni.

Quando viene applicato ad una sequenza di immagini il metodo di Lucas-Kanade viene iterato in maniera "*coarse-to-fine*" in questo modo le derivate spaziali sono prima calcolate nella scala spaziale (piramide) più grezza, ciascuna delle immagini viene modificata tramite le deformazioni calcolate precedentemente, e in maniera iterativa vengono effettuati degli aggiornamenti che vengono applicati ai livelli più fini.

L'algoritmo di Lucas-Kanade é molto utilizzato perché molto robusto al rumore.

Figura 25: optical flow su riprese di un quadro dove si sposta la videocamera.

2.1.5 Filtro di Sobel

Si prenda come esempio una maschera di dimensione 3x3.

U1	U2	U3

(x-1,y-1)	(x,y-1)	(x+1,y-1)
U4 (x-1,y)	U5 (x,y)	U6 (x+1,y)
U7 (x-1,y+1)	U8 (x,y+1)	U9 (x+1,y+1)

Si consideri un'immagine I, e sia I(x,y) il valore del punto nella coordinata x,y. L'immagine filtrata tramite questa maschera si ricava dalla seguente relazione:

$$If(x,y) = U1*I(x-1,y-1) + U2*I(x,y-1) + U3*I(x+1,y-1) +$$

$$U4*I(x-1,y) + U5*I(x,y) + U6*I(x+1,y) +$$

$$U7*I(x-1,y+1) + U8*I(x,y+1) + U9*I(x+1,y+1)$$

Si noti che la precedente relazione non è applicabile ai bordi dell'immagine in quanto la maschera esce dall'immagine stessa. Per questo motivo le immagini filtrate con maschere di questo tipo sono in più piccole di quelle di partenza.

A titolo di esempio consideriamo la seguente maschera:

-1	-2	-1
0	0	0
1	2	1

Essa è chiamata è L'operatore di Sobel, corrisponde ad un operatore di tipo gradiente verticale e serve per mettere in evidenza i bordi orizzontali degli oggetti di un'immagine.

Si consideri infatti una porzione di immagine a intensità costante, caratterizzata quindi da pixel aventi lo stesso valore di intensità e vi si applichi l'operatore di Sobel:

```
100  100  100        -1   -2   -1
100  100  100   ⊗    0    0    0 = 100*(1+2+1)+100(-1-2-1) = 0
100  100  100        1    2    1
\----------------/   \----------------/
   Immagine            operatore Sobel
```

Si consideri ora una porzione di immagine caratterizzata da una linea orizzontale chiara:

```
10   10   10         -1   -2   -1
10   10   10    ⊗    0    0    0 = 220*(1+2+1)-10*(-1-2-1) = 840
220  220  220        1    2    1
\----------------/   \----------------/
```

Se si applica l'operatore all'immagine di un rettangolo, l'immagine risultante (in valore assoluto) è la seguente (figura 25).

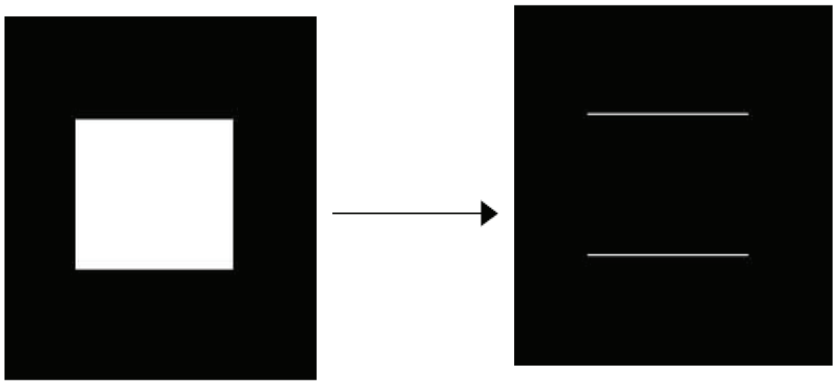


Figura 25: Applicazione filtro Sobel orizzontale ad un rettangolo.

Naturalmente esiste anche l'operatore di Sobel per la rilevazione delle righe verticali. Esso è ottenibile da quello presentato ruotando la matrice di
90 gradi.

Prendendo l'immagine ottenuta con l'operatore gradiente verticale e quella ottenuta con l'operatore gradiente orizzontale, e le si combina nel seguente
modo:

$$\text{Imagine Finale} = \text{Im}\,fin = \sqrt{\text{Im}\,gradiente\,verticale^2 + \text{Im}\,gradiente\,orizzontale^2}$$

Figura 26: Immagine Originale.

Figura 27: Immagine con applicato filtro Sobel.

2.2 STIMA

In questo progetto sono stati applicati degli algoritmi di stima per misurare le modifiche avvenute su ognuno dei tre estrattori.

Prima di passare in dettaglio i singoli sistemi vorrei introdurre un po' il concetto di stima e come è legato all'intelligenza artificiale per rendere tutto più comprensibile.
Si definisce Intelligenza Artificiale quell'attività che si propone di indagare i meccanismi che sono alla base della cognizione umana, in primo luogo il ragionamento logico-matematico, la capacità di risolvere problemi e la comprensione del linguaggio naturale, con il fine dichiarato di riprodurli per mezzo di elaboratori elettronici sufficientemente potenti. La posizione di partenza dell'intelligenza artificiale è costituita dal fatto che molti scienziati considerano il pensiero umano come una forma di calcolo e, in quanto tale, potenzialmente riproducibile tramite computer dotati di adeguati software.
Da questo punto di vista si possono distinguere due posizioni principali: quella dell'intelligenza artificiale forte e quella dell'intelligenza artificiale debole.
I sostenitori della tesi forte dell'intelligenza artificiale sono convinti che il pensiero sia interamente riconducibile a un processo di manipolazione di simboli che si attua utilizzando un gran numero di algoritmi. Pensare, in tale prospettiva equivale a calcolare, da cui deriva che la mente è equivalente al software di un calcolatore.
Per i sostenitori della versione debole dell'intelligenza artificiale, pensare non è sinonimo di calcolare, poiché le capacità del pensiero non si limitano a quelle logico-matematiche. Il calcolatore non è visto come l'analogo di un cervello biologico, ma come uno strumento che può essere di grande aiuto nella comprensione dei processi cognitivi.

E' necessario ora analizzare il processo attraverso il quale si ottengono i dati *significativi* o *feature* del problema.
Il problema di per sé può essere semplificato nel riconoscere all'interno di un insieme numeroso di eventi o transazioni quelli che costituiscono le cosiddette "anomalie". Quindi un evento dovrà necessariamente appartenere o alla classe degli eventi "normali" o a quella degli eventi "anomali", quindi per prima cosa si deve fare una distinzione chiara tra le caratteristiche proprie delle due classi che si sono identificate.
Questo processo di caratterizzazione dovrebbe permettere di assegnare in maniera univoca ogni evento alla classe corrispondente, e successivamente di dare un'etichetta ad ognuno legandolo alla classe a cui questi è stato assegnato. A tutto ciò segue una problematica definita come problema del riconoscimento. L'assegnazione di ciascun evento reale ad un insieme teorico in precedenza da noi definito in un certo modo serve a dare una codifica del mondo "riconoscendo" ogni dato all'interno di una categoria.

In generale il processo è suddiviso in tre fasi.

- *Evento:* Accade l'evento di cui non si conosce la classe di appartenenza ma si sa la famiglia di classi a cui appartiene (cioè quelle da noi scelte).

- *Estrazione delle Feature:* Si procede col digitalizzare l'evento con l'intento di creare una rappresentazione adeguata per la classificazione in uno spazio vettoriale. L'obbiettivo è ridurre la dimensione dell'evento fornendo maggior rilievo agli elementi che sono più ricchi di informazione.

- *Classificazione:* Se si suppone l'esistenza di un sistema esperto in grado di assegnare gli eventi alle diverse classi, si avvia il processo di riconoscimento delle feature digitalizzate, fornite in input, in modo da poter ricevere in output la classe di appartenenza relativa.

Il termine Classificare ha comprende una moltitudine di metodi, algoritmi e approcci mirati a suddividere la realtà degli eventi in insiemi di definizione a noi utili. Una tipologia d'approccio classica è legare l'idea di classificazione alla creazione di *n* funzioni, una per classe, che prendono in ingresso le feature e restituiscono in uscita un valore numerico significativo. In questo senso la dimensione del valore di output è

indice di quanto il dato sia vicino o appartenente alla classe; tanto è più grande quanto sarà maggiormente probabile che l'evento appartenga alla classe corrispondente a quella funzione. Quindi per risolvere un problema di classificazione è necessario inserire le feature dell'evento all'interno di tutte le *n* funzioni (se ho *n* classi) e verificare quale è l'output più grande; quella è la sua classe di appartenenza e tale funzione prende il nome di *funzione discriminante.*

A proposito, è utile considerare le densità di probabilità delle classi e dell'evento, in particolare il Teorema di Bayes può chiarire molto le cose:

$$P(w_i \mid x) = \frac{p(x \mid w_i) * P(w_i)}{p(x)}$$

Dove:

- $P(W_i \mid X)$: è la probabilità a posteriori dato l'evento *x* che questo appartenga alla classe *i.*
- $P(X \mid Wi)$: è la probabilità che data una certa classe *i,* si verifichi l'evento *x.*
- $P(W_i)$ è la probabilità a priori che un elemento della popolazione appartenga alla classe *i.*
- p(x) è probabilità che si verifichi l'evento *x.*

Quindi, si dovrebbe utilizzare come funzione discriminante il risultato del teorema di Bayes; la regola di decisione di Bayes indica, cioè la probabilità che l'evento, una volta avvenuto, appartenga a una determinata classe. Usando funzioni su *n* classi, logicamente si otterranno risultati che non sono altro che le probabilità di appartenenza ad ogni classe del singolo evento in questione. Per questo motivo si parla solitamente di massimizzare la funzione, infatti, a questo punto del processo si può dire, senza certezza assoluta però, che l'evento appartiene alla classe (funzione) che ha dato output più alto cioè che con maggiore probabilità ci assicura la classe all'evento.

Spesso la conoscenza della probabilità a posteriori $P(W_i \mid X)$ non è data. Tuttavia è sempre possibile descrivere questa probabilità a partire dalle informazioni che ci fornisce il teorema di Bayes. E' infatti possibile supporre la distribuzione delle classi $P(W_i)$, dato che le fissiamo noi, e degli eventi *p(x)* o comunque si possono aggiornare questi parametri durante il processo di classificazione.
Considerati appunto questi due termini come costanti, risulta possibile modellare la $P(W_i \mid X)$ sulla dipendenza dalla probabilità a priori $P(X \mid W_i)$. Per chiarire meglio il concetto di "a priori" riporto il classico esempio del classificatore di sesso dato un determinato campione di individui.

Supponiamo di avere 2 classi: F indica il sesso femminile e M indica il sesso maschile. Devo decidere se un nuovo individuo X è maschio o femmina. Una volta definite queste si passa a considerare la presenza o meno di feature estratte:

- Spazio delle feature vuoto: In questo procedimento l'unica funzione discriminante è la probabilità a priori delle classi. Se non posso "vedere" il nuovo individuo (misurarlo ovvero estrarne le feature), non mi resta che decidere in base alle probabilità a priori P(F) e P(M). Decido F se P(F) è maggiore di P (M), altrimenti scelgo M. In pratica si cerca di minimizzare la probabilità d'errore. Per esempio in occidente ho circa P(F) = 0.52 e P(M) = 0.48, e quindi decido sempre per ogni nuovo individuo che appartiene ad F .

- Spazio delle feature non vuoto: In questo caso il sistema possiede una classificazione del mondo secondo dei particolari criteri di scelta, e secondo questi riceve in input delle feature ad hoc. Un esempio di feature può essere l'altezza dell'individuo. Si creano così delle *Class Conditional Pdf*, cioè le probabilità a priori $P(X \mid W_i)$.
  Esse esprimono in funzione di un parametro, ad esempio l'altezza, la probabilità che data una classe l'evento che si verificherà gli appartenga.

L'utilizzo di una funzione discriminante non è l'unico metodo possibile per ottenere una buona classificazione. Al posto di muoversi sulla linea della regola di decisione di Bayes può essere molto utile cercare di capire a fondo la forma di queste funzioni di densità di probabilità.

Una volta addentratisi nel problema della stima è necessario operare una distinzione di fondo delle due grandi famiglie sotto le quali si racchiudono questi metodi:

• *Stima Parametrica:* metodo di stima che suppone di conoscere la forma della pdf e di poterne determinare le caratteristiche a partire da un vettore di parametri.

• *Stima Non Parametrica* al contrario non fa assunzioni di nessun tipo sulla forma della pdf né cerca quindi dei parametri ad essa relativi. Essa si basa unicamente nell'individuare dei metodi alternativi, come ad esempio intuizioni di tipo geometrico, per classificare gli eventi.

Sotto questo tipo di ipotesi, cioè sulla conoscenza della forma della pdf, si basa lo studio della distribuzione degli eventi messi in relazione con le loro classi di appartenenza. Esiste tuttavia un ulteriore distinzione da fare all'interno di ognuna di queste due classi di problemi, in base alla presenza o meno di campioni già etichettati. Infatti se è già presente un insieme di etichettature, caso Supervisionato, si hanno già i risultati di cosa si sta classificando. Diversamente quando si parla di *non Supervisione* non si ha effettivamente idea di come siano distribuiti i campioni sulle varie classi, che nella realtà si traduce nel non sapere di preciso i termini generici del problema o nel semplice non avere esempi di training con indicazione della classe di appartenenza.

Il metodo Parzen Window che è quello utilizzato nel nostro progetto fa parte delle stime parametriche non supervisionate. Tutti i metodi di questo tipo si propongono di ricostruire la funzione di distribuzione di probabilità a partire da semplici valori numerici probabilistici presi come media di certi intervalli di funzione.

Il *Parzen Window* (or *kernel density estimation*) deriva da Emanuel Parzen, ed è appunto un metodo per stimare la *probability density function* di una variabile casuale. Forniti alcuni dati di un campione di popolazione, il metodo del *Parzen Window* rende possibile estrapolare i dati dell'intera popolazione. Il Parzen Window utilizza una funzione finestra (Kernel), ovvero un ipercubo, ben centrato e di lato noto, associabile a una classe, al fine di calcolare il numero di campioni che cade al suo interno. Il kernel fa

quindi una stima delle Pdf attraverso un'interpolazione, cioè calcola una "media" sui campioni delle funzioni kernel corrispondenti, pesata sul volume dell'ipercubo.
Se $X_1,X_2,\ldots X_N$ è un campione di una variabile casuale, l'approssimazione Parzen Window della sua funzione probabilità di densità sarà data da:

$$\rho(x) = \frac{1}{N}\sum_{i=1}^{N} W(x - x_i)$$

dove W è un kernel stocastico.
Spesso W è una funzione gaussiana con media nulla e varianza σ^2.

$$W(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-x^2/2\sigma^2}$$

Nella figura 28 per esempio abbiamo sei gaussiane (in rosso) e le loro somme (in blu).

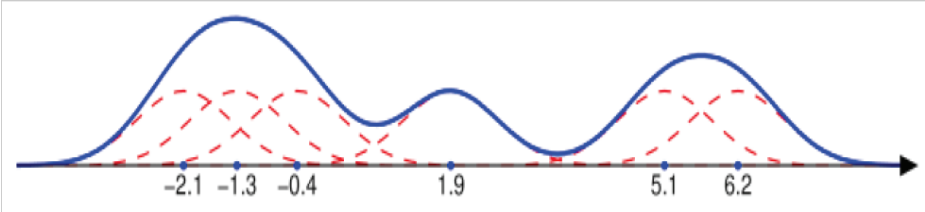


Figura 28

La stima Parzen Window della densità è ottenuta in come si vede, dividendo questa somma per 6, il numero di gaussiane . La varianza della gaussiana è stata posta a 0.5. Notare che dove i punti sono più densi, la stima di densità ha valori più alti.

CAPITOLO 3

ARCHITETTURA DEL SISTEMA

In questo capitolo verranno presentati tutti gli strumenti, sia hardware che software utilizzati per la progettazione del sistema. Quindi sarà presentato l'ambiente di sviluppo utilizzato,le librerie, le videocamere ed tutto quello che è stato utilizzato per la progettazione e in particolar modo tutta la progettazione software, l'implementazione dei vari algoritmi, il livello intermedio ed il super visore del sistema. A seguire sarà presentata la struttura del sistema composto dai tre livelli, con la descrizione in particolare del livello intermedio e del supervisore.

3.1 STRUMENTI SOFTWARE

3.1.1 Le Librerie

Gran parte del lavoro relativo alla progettazione del software viene fatto mediante l'utilizzo di librerie open source o a pagamento.La libreria permette di ottenere due cose e precisamente di ridurre drasticamente il tempo relativo allo sviluppo e quello di semplificare concettualmente la progettazione.
Molti algoritmi utilizzati nell'elaborazione delle immagini sono definiti pesantemente da modelli matematici specifici di una qualche teoria. Nell'ambito di tale settore esistono un'infinita di trasformate matematiche che vengono utilizzate per la riduzione della quantità di informazioni che si devono trattare. Il dovere tradurre in algoritmi tali sistemi matematici pretenderebbe un lunghissimo tempo di stesura e messa a punto.
Esistono molte librerie rilasciate tra l'open source che contengono un enorme patrimonio funzionale applicabile in questo settore.

OpenCV dell' Intel è un esempio di questo tipo. Ad esempio negli anni '80 c'era il compilatore Microsoft C, il sistema operativo era il DOS e quindi per la gestione dell' I/O era sufficiente utilizzare funzioni come printf e scanf. All'interno della libreria non erano neppure fornite funzioni per il posizionamento del cursore sul video per cui una stampa formattata doveva avvenire spostando il cursore mediante la stampa di tante linee e di tanti spazi.
Da allora è passata moltissima acqua sotto i ponti per cui di fatto risulta impensabile gestire anche una semplice formattazione video mediante stampe di spazi e di return. Con il passare del tempo windows diventò il sistema operativo per eccellenza e quindi tutta la gestione dell'interfaccia utente fu demandata a una serie di librerie definite con il termine di windows api. A quei tempi il linguaggio era il C quindi non si parlava di object oriented. Con il passare del tempo i metodo di analisi diventarono orientati a risolvere problemi più complessi perciò il linguaggio C++ diventò il linguaggio

d'obbligo. Il fatto di aver adottato l'object oriented permise di incapsulare le api all'interno delle *Microsoft Foundation Class*.

Queste pian piano diventarono le librerie fondamentali nell'ambito dello sviluppo in ambiente windows espandendosi sempre di più con l'uscire delle nuove versioni del sistema operativo e di conseguenza di quello di sviluppo. Al giorno d'oggi esiste un grossa diffusione dell'Open Source anche se di fatto questo è molto più diffuso in ambiente Linux che in ambiente Windows. L'utilizzo della libreria in questo caso permette di affrontare dal punto di vista funzionale il problema omettendo la trattazione matematica.

L'utilizzo della libreria permette la scrittura di molte complesse funzionalità software senza necessariamente dover trattare complesse formule matematiche. Ci sono varie tipologie di librerie: *LIB* librerie statiche collegate al programma in fase di linking, *DLL* librerie dinamiche lette al momento dell'uso, *OCX* activeX una specie di DLL, *BPL* librerie dinamiche Borland legate al mondo VCL (Visual Component Library) . Negli ultimi anni le librerie statiche hanno lasciato il posto a quelle dinamiche le quali non hanno solo il vantaggio di essere lette dinamicamente in memoria ma anche quello di essere condivise tra diversi programmi.

Ma come sono state utilizzate le librerie nel nostro progetto?

Per prima cosa le immagini sono acquisite da videocamere o da periferiche d'input per essere in seguito elaborate e preparate per il sistema vero e proprio di riconoscimento. Il sistema d'acquisizione permette di selezionare i driver di aggancio con un certo tipo di videocamere e da questo deve ricevere le immagini. Il sistema d'acquisizione può funzionare in background oppure può essere eseguito in modo che visualizzi le immagini a video. Poi un'immagine deve essere trattata mediante algoritmi grafici di preparazione. In particolare abbiamo utilizzato le librerie sia per l'interfacciamento a videocamere e l'acquisizione video, sia per la gestione a video e quindi la visualizzazione sulla finestra delle immagini acquisiste.
Nel campo della *computer vision* molte funzionalità sono definite da modelli matematici molti dei quali derivati da lunghi studi e sperimentazioni.

3.1.2 Libreria OpenCV

OpenCV è una libreria delll'*Intel*® rilasciata tra l'open source che contiene moltissimi insiemi di funzioni ciascuna delle quali adibita allo svolgimento di un determinato compito nell'ambito della computer vision.

La libreria OpenCV è presente su diverse piattaforme tra le quali windows e linux. Il sistema dispone di alcune librerie complementari che permettono la gestione di un windowing di base e la cattura delle immagini da videocamere. Questa libreria implementa una varietà di tool per l'interpretazione delle immagini. Essa è compatibile con l'*Intel*® *Image Processing Library* (IPL) che implementa una serie di operazioni a basso livello sulle immagini digitali. Le qualità principali della libreria, oltre ad essere molto funzionale, sono le proprie performance.

Gli algoritmi sono basati su *Data Structures* veramente flessibili (Dynamic Data Structures), compatibili con IPL Data Structures. La maggior parte delle funzioni sono *Assembler-optimized* e si avvantaggiano, per ottenere ottime prestazioni, dell'architettura *Intel*® (Pentium® MMX™, Pentium® Pro, Pentium® III, Pentium® 4).

La libreria OpenCV di Intel è stata progettata con l'obiettivo di creare una comunità open source per migliorare l'opportunità di utilizzare la computer vision nel sempre più crescente PC environment. La *OpenCV Library* è una *platform-independent interface* scritta in codice *ANSI C* .

Open CV è stata progettata per essere utilizzata insieme Intel® Image Processing Library (IPL) ed estende le sue funzionalità alle immagini e all'analisi di patterns. Inoltre OpenCV condivide lo stesso formato immagine (*IplImage*) con IPL. Oltre a ciò OpenCV utilizza Intel® Integrated Performance Primitives (IPP) a basso livello. IPP fornisce un'interfaccia *cross-platform* per maggiormente ottimizzare le funzioni di basso livello, che sono performanti su operazioni di dominio specifico: particolarmente nell'*image processing* e nella *computer vision.*

La libreria è distribuita in formato sorgente con i makefile utilizzabili per la compilazione sotto diversi sistemi operativi.

3.1.3 Libreria ImageEn

Altra libreria, L'*ImageEn* è stata utilizzata nel progetto per sfruttare le sue potenzialità di libreria di input/output di file nei più svariati tipi.
Una delle funzionalità più importanti è sicuramente quella che permette di acquisire immagini provenienti da una videocamera e quindi di trattarle al fine di poter fornirgli algoritmi d'elaborazione. ImageEn possiede un grosso numero di metodi adatti a svolgere questo tipo di applicazioni. Una volta effettuata l'installazione, il software crea diverse directory, una per ciascuna versione del compilatore. Quella d'interesse per il nostro progetto è la BCB5.

Esistono due tipi di file BPL e precisamente i files che iniziano per D, dove D sta per versione designer e quelli che iniziano per P i quali sono relativi alle librerie di runtime. Quelle di runtime dovranno essere distribuite con l'applicativo mentre quelle di design time dovranno essere installate all'interno del sistema di sviluppo. Una volta selezionate i due files .BPL si trova nella toolbar di lavoro la voce contenente gli oggetti di ImageEn. La suddivisione in due librerie è stata fatta in quanto nel primo tipo sono presenti tutti gli oggetti che gestiscono le immagini video prese da periferiche di I/O o da driver di acquisizione legati a videocamere.
Il secondo tipo invece gestisce immagini presenti dentro a campi di database. Il secondo tipo risulta essere veramente utile nell'istante in cui si vogliono gestire immagini direttamente all'interno di campi binari di sistemi informativi.

Il sistema di *ImageEn* è stato scritto con metodologie object oriented e grazie a queste deriva molti oggetti da altri di windows e dal sistema di *directx*. In effetti molti metodi e proprietà derivano direttamente da questi sistemi anche se di fatto grazie alla notevole potenza della libreria queste spesso sono completamente trasparenti.

ImageEn tratta le immagini sia in formato pixel che in formato vettoriale per cui alcuni oggetti si riferiscono a uguali funzioni ma di fatto a oggetti trattati in modalità differente. Ogni oggetto immagine può possedere tre classi abbinate ciascuna delle quali tratta la visualizzazione, l'I/O e il processing. Se ad esempio volessimo leggere da file un'immagine e visualizzarla a video dovremmo disporre sul video un oggetto di tipo *ImageEnView* a abbinargli un altro oggetto *ImageEnIO*.

All'interno dell' *Object Inspector* esiste una voce *AttachedTImage* che permette di selezionare quale campo di visualizzazione abbinare al blocco di I/O. Chiaramente ciascuna tipologia di oggetto possiede proprietà e metodi inerenti alla loro natura. Quelli di I/O possiedono metodi adatti a gestire l'I/O mentre quelli di processing sono adatti a modificare le caratteristiche dell'immagine come ad esempio i metodi di conversione, di equalizzazione, per la modifica della caratteristiche cromatiche.
Esistono alcuni oggetti che permettono di acquisire direttamente immagini video da sorgenti come videocamere, scanner e macchine fotografiche.
Si tratta di *ImageEn Video View*. Questi oggetti contengono due proprietà chiamate rispettivamente *ShowVideo* e *VideoSource*.
La Prima è un flag logico che dice se deve essere visualizzata l'immagine video mentre la seconda è il numero della videocamera sorgente.

L'uso di questi due parametri permette di vedere direttamente anche in fase di design time l'input da videocamera. Chiaramente questa funzione interattiva è utile per testare le funzionalità del programma senza necessariamente compilare il tutto. Si deve avere sempre bene presente che ImageEN utilizza a basso livello sia il metodop delle DirectX che quello del Video For Windows.

3.1.4 Ambiente di Sviluppo

Altri strumenti utilizzati in questo progetto che servono per estrapolare immagini da un video e quindi permetterne l'elaborazione sono i Frame Grabber.

I frame grabber si suddividono in due tipi: quelli digitali e quelli analogici.

Le *CCD* o le *CMOS* forniscono i dati secondo una logica parallela o seriale per cui l'interfacciamento con i computer necessita di sistemi di conversione che possano fornire i dati seguendo un standard che può essere quello USB o quello parallelo.

FireWire è la marca della *Apple* per lo standard *IEEE1394*.

Lo standard IEEE 1394 ha la sua origine nelle idee degli ingegneri della Apple. Durante la metà degli anni 80, cercavano un bus che potesse essere molto più pratico e facile da usare del bus SCSI. Questa facilità di impiego è caratterizzata da:

• Cavi lunghi e flessibili

• Nessuna terminazione bus necessaria.

• Nessun indirizzamento dei dispositivi tramite interruttori DIP.

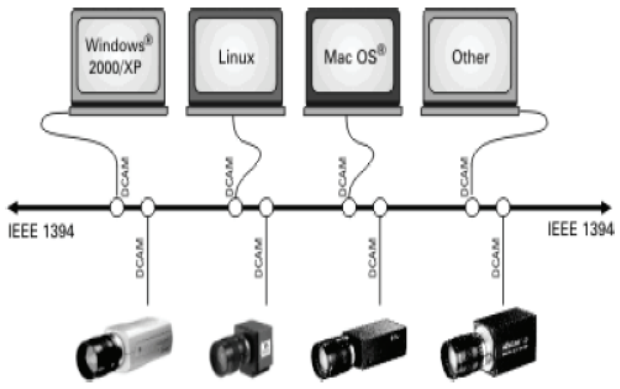


Figura 29

Quindi l'IEEE 1394 è decisamente più di un collegamento video digitale. IEEE 1394 è anche conosciuto come FireWire e 'i.LINK.
Molti grabber si installano come sistemi PCI o PCMCIA. Molti di questi dispongono di caratteristiche avanzate come ad esempio sistemi di compressione hardware che permettono di memorizzare 1000 ore su un disco di 80 Gbytes. Il DCAM è un protocollo per telecamere FireWire . Definisce il trasferimento dei dati delle immagini dalla telecamera (senza compressione e audio) il trasferimento dei parametri alla telecamera (il tempo di esposizione per esempio). L'interfacciamento avviene sempre mediante driver. I driver delle sorgenti d'immagini, come le telecamere FireWire, i convertitori Video-to-FireWire, i frame grabber etc, usati nei campi d'applicazione come la sicurezza, la tecnica, la scienza e la medicina, seguono una di queste due strategie:

- L'hardware viene fornito con un driver di unità e un kit di sviluppo di software (SDK), che permette al programmatore di accedere allo stesso driver. Questo modo di procedere è anacronistico e quindi sensato solo in casi eccezionali.

- L'hardware viene fornito con un driver che 'incorpora' questo hardware nel sistema operativo. In questo modo il software applicativo è indipendente dall'hardware perché non accede più ad un hardware speciale, ma utilizza il sistema operativo come 'intermediario'.

Molte ditte come *The Imaging Source* offre i cosiddetti driver *WDM Stream Class* per varie sorgenti video. *WDM* sta per *Windows Driver Model*. I driver WDM Stream Class rappresentano il modo 'ufficiale' per incorporare dispositivi audio e video in Windows.
L'interfacciamneto di C++Builder e ImageeN con la videocamera può avvenire tramite Video For Windows o mediante DirectX.
In ogni caso il sistema setta una funzione alla quale arrivavano tutte le chiamate in ricorrenza di un nuovo frame. Molti grabber evoluti dispongono di funzioni di compressione interni e oltre a questo sono inseriti su bus per cui il trasferimento dati tra grabber e disco potrebbe avvenire tramite canali diretti senza necessariamente far fare da passamano al software.

Molti grabber evoluti dispongono di SDK per la loro programmazione e per la gestione di funzioni diretta come ad esempio quelle di salvataggio. Il grabber per questo uso deve possedere un bus in modo da porte utilizzare questo per trasferire i dati su disco e in più dovrebbe disporre dell'algoritmo di compressione hardware.
I grabber scrivono i dati su bus perciò per molte funzioni non occupano risorse macchina rendendo quindi possibile la scrittura di software.
L'acquisizione immagini da videocamera, nel caso di quelle di rete, non utilizzano schede elettroniche
d'interfacciamento. L'uso di queste videocamere sta diventando sempre più massiccio nell'ambito della videosorveglianza tanto da essere diventato il modello più diffuso.

Come strumento per la programmazione è stato utilizzato *Borland c++ Builder 6* .
Borland C++Builder 6 rivoluziona il panorama C++ abbinando Internet allo sviluppo C++. La release 6 unisce l'ambiente di sviluppo C++ di Borland agli ultimi standard Internet, *XML* e *HTML 4*, offrendo la possibilità di realizzare applicativi Web server ad altissime prestazioni.
C++Builder 6 garantisce che i progetti sviluppati oggi - sia quelli creati ex novo che quelli fondati su progetti già implementati - risulteranno funzionanti anche in futuro grazie alla totale integrazione del *CORBA ORB VisiBroker* per C++.
C++Builder 6 è l'unico compilatore C++ che integra entrambi gli standard per oggetti distribuiti *COM* e *CORBA* all'interno dello stesso ambiente di sviluppo, semplificando radicalmente lo sviluppo di complicate soluzioni distribuite ed assicurando l'interoperabilità con gli oggetti e i client Windows, Linux, UNIX, e Java.

3.2 PROGETTAZIONE SOFTWARE

3.2.1 Primo Livello

La videosorveglianza come spiegato precedentemente sta diventando un'attività di grande interesse, specialmente se utilizzata in ambienti molto "sensibili" come i musei, dove sono conservate innumerevoli opere d'arte che rendono ancora più meraviglioso il nostro "Belpaese". Il sistema progettato in questa tesi può essere utilizzato per la videosorveglianza in svariate situazioni , ma per la sua sperimentazione ci si è dedicati alla sicurezza di un museo di Firenze dove si è potuto testare il funzionamento del sistema nella pratica.

La stragrande maggioranza di questi musei già possiede una qualche forma di videosorveglianza passiva; proprio per questo motivo che il nostro sistema può essere utilizzato in una *duplice modalità*. Una modalità che riesce a leggere dei filmati in formato digitale, (filmati che possono provenire dalle videocamere già esistenti all'interno del museo) oppure in modalità diretta in tempo reale, per avere dei riscontri istantanei. Lo scopo principale del sistema è l'individuazione di *situazioni potenzialmente pericolose* per le opere d'arte conservate. Per riuscire in questo scopo le immagini provenienti da una videocamera (o filmato digitale), sono analizzate fotogramma dopo fotogramma.

L'analisi si compone di *tre parti fondamentali* che vengono elaborate di continuo tramite il flusso di fotogrammi *(frame-rate 1fps* circa).

1) La prima parte è quella realizzata tramite *la trasformata di Hough* alla ricerca dei pixel che sono *a massima invarianza nella scena*. Questi punti, infatti ci aiutano a capire se nella scena ci sono delle forti variazioni che possono essere sospette. Il fotogramma corrente viene prima di tutto sottoposto al filtro di Canny per riuscire a determinare degli edge nello scenario. Gli edge trovati sono poi sottoposti alla trasformata di Hough per determinare quelli maggiormente stabili. Le variazioni in una scena ripresa in tempo reale sono molteplici e possono avvenire anche per situazioni ben lontane dall'essere pericolose (o di furto); proprio per questo motivo le informazioni derivanti dal filtro di Hough sono state utilizzate per addestrare e testare dopo 50 frame uno stimatore Parzen Window. Nei primi 50 frame il sistema è assunto pressoché stabile. In seguito viene creata una regione composta da questi fotogrammi, sulla quale vengono poi confrontati tutti i successivi, calcolando così una probabilità (numero compreso tra 0 e 1) che ci dice quanto il frame corrente si avvicina alla stabilità. Chiaramente un valore pari a 0.5 (50%) come soglia presupporrebbe uno stimatore "oracolare*"* che non è possibile , quindi è stata scelta una soglia tra 0.6 e 0.7 che fornisce risultati abbastanza veritieri. I picchi rilevati vengono segnalati con dei puntini colorati sull'immagine rappresentata dagli edge.

2) La seconda analisi è effettuata è quella dell'*immagine mediana*. A differenza degli altri due estrattori (analizzatori), questo avviene in maniera sottocampionata; infatti i fotogrammi che compongono la mediana, vengono catturati ogni 10-20 frame, questo perché lo scopo di questa elaborazione è quello di fornirci un'idea più generale di quello che stiamo osservando. Cerchiamo tramite il sotto campionamento di ridurre le variazioni "frame by frame" e di far calcolare alla mediana solo oggetti che rimangono stabili nel tempo. La mediana è formata da 5 frame in toni di grigio che vengono ordinate tramiti *l'algoritmo Bubblesort* , usando *il livello di grigio* associato a ciascun pixel come misura per i confronti. Non sono state usate informazioni sul colore per ridurre la complessità del problema, (avremmo dovuto calcolare una mediana vettoriale, così invece è scalare) mantenendo comunque una buona accuratezza. La mediana calcolata ogni 50 frame (fattore di sottocampionamento moltiplicato per numero di frame che compongono la mediana) viene aggiornata con la tecnica *FIFO (first in first out)*. Una volta calcolata la mediana (che rappresenta 50 frame), a questa viene sottratto il frame corrente tutto calcolato in valore assoluto, riuscendo così a determinare gli oggetti che si discostano dai frame che la mediana rappresenta. L'immagine differenza viene poi binarizzata e vengono inviate allo strato successivo solo le informazioni posizionali dei pixel chiari. Quest'ultimo dei tre estrattori incorpora anche gli altri perché con buone probabilità gli oggetti che saltano fuori nell'immagine differenza, sono legati sia al movimento che ai cambiamenti della scena.

3) La terza analisi è quella relativa *al movimento*, questo tipo di analisi della scena, serve per individuare tutti quegli oggetti che si spostano con una certa velocità in ogni direzione. L'elaborazione è stata effettuata tramite *l'algoritmo iterativo di Lucas-Kanade* che riesce a rilevare quei punti nell'immagine che si spostano in due frame consecutivi. L'immagine sorgente è stata suddivisa in tre porzioni identiche (alta, media, bassa) e su ciascuna è stato applicato l'algoritmo di Lucas-Kanade in modo da poter distinguere quale delle tre zone ha movimento e di poter così decidere più facilmente se si tratta di movimenti consentiti e

naturali oppure no . In questa analisi non è stato utilizzato nessun tipo di stimatore, ma si è scelto di usare una soglia di velocità che può essere modificata tramite un apposito controllo sull'interfaccia del software. I pixel che subiscono movimento vengono segnalati nel sistema tramite delle frecce colorate che rappresentano l'intensità del movimento tramite la loro lunghezza . Questo tipo di estrattore si presenta molto efficace ed è del tutto complementare a quello di Hough descritto prima.

L'analisi attraverso queste tre componenti è solamente il primo livello del sistema. I nostri tre stimatori ci forniscono delle visioni parziali e riduttive se presi singolarmente, ma se considerati in maniera unitaria riescono a darci un inquadramento molto preciso di ciò che sta avvenendo nella scena sorvegliata. Grande importanza quindi,viene rivestita da un livello (layer) centrale che unifica le informazioni dei nostri tre estrattori e dal successivo livello decisionale.

3.2.2 Feature Strutturali e Scelte progettuali

Le feature necessarie alla creazione di questo sistema di videosorveglianza dovevano essere di natura differente, alcune di *tipo strutturale*, altre di *tipo dinamico*. Le prime necessarie per comprendere le relazioni spaziali all'interno dell'immagine riprese, cercare cioè di capire i vari elementi presenti all'interno della scena. Le altre per fornirci elementi sufficienti a capire, se e come un oggetto presente nella scena si sposta. Riuscire ad estrarre le feature necessarie per il sistema ha richiesto una serie di prove sperimentali dove sono state testate tecniche differenti dell'elaborazione delle immagini per raggiungere i risultati ottimali.

Per quanto riguarda le feature di natura strutturale la linea guida è stata quella di riuscire a determinare i punti a minima varianza nell'ambiente ripreso, in modo da poter sfruttare queste informazioni per determinare se avvengono cambiamenti importanti nella scena. La prima tecnica utilizzata è stata la ricerca degli edge, seguita dalla ricerca degli angoli delle immagini, ipotizzando che i bordi delle figure rimanessero stabili nel tempo, ma questa idea è stata subito vanificata sul campo, gli angoli infatti sebbene abbastanza stabili al "rumore" (tutto ciò a cui il sistema non è interessato), non fornivano sufficienti informazioni sulla distribuzione degli oggetti principali delle immagini. Un'altra tecnica che si rilevata per alcuni aspetti interessanti è stata la ricerca dei contorni degli oggetti; questo procedimento però aveva delle limitazioni, forniva infatti una quantità ridondante di informazioni (in alcuni casi i contorni erano difficili da ricavare e buona parte dell'immagine poteva esserlo) difficilmente rappresentabile successivamente in modo simbolico. La scelta quindi è ricaduta sulla trasformata di Hough e sulla mediana temporale per le feature di tipo strutturale. La trasformata di Hough successiva alla determinazione degli edge, si rivela molto robusta nel determinare dei picchi a massima invarianza che rappresentano la nostra scena stabile in modo compatto e simbolico invariante a piccoli movimenti provocati dal rumore nel

sistema. La struttura dell'immagine ripresa realizzata tramite le distanze (calcolate tramite le coordinate univoche dei pixel) tra tutti questi picchi, viene poi fornita in ingresso ad un analizzatore di tipo *Parzen Window*. Quest'ultimo è stato scelto per la sua semplicità di stimatore non parametrico; potevano essere utilizzati approcci molto più sofisticati (ANN, HMM, SVM) ma l'eccessivo richiesta computazionale (il sistema deve essere real-time) e il lungo percorso di training, hanno spinto verso questo semplice stimatore che si è comportato discretamente bene, anche in termini di memoria utilizzata. L'analizzatore è riuscito a fornire un'informazione probabilistica sull'appartenenza o meno della struttura simbolica fornita da Hough a situazioni di stabilità del sistema.

L'altro estrattore di feature costituenti la scena è una *mediana di frame*. Sono presi dei frame ad intervalli regolari e si calcola il valore mediano su un blocco di questi frame riordinati. Tra l'immagine mediana ed il frame attuale viene poi calcolata la differenza in valore assoluto. In questo modo si cerca di ottenere un'immagine che è il valore intermedio tra molti frame ed indice di una variazione persistente nella realtà. Questo estrattore permette ad esempio di non considerare situazioni d'allarme spostamenti veloci come quando una persona passeggia davanti ad un quadro o in contesti dove c'è un via vai continuo di persone. Questo tipo di estrattore è stato ritenuto adatto al sistema, proprio perché riesce a fornire delle regioni che rappresentano gli oggetti che sono diversi dal "normale " (rappresentato dall'immagine mediana).

I due estrattori riescono insieme a fornirci una visione della scena del tutto completa, infatti tramite l'estrattore di Hough riusciamo a creare delle strutture simboliche che rappresentano l'opera sorvegliata e attraverso la mediana di immagini riusciamo ad inserire queste strutture in regioni che rimangono stabili ai cambiamenti temporali. Un tipico esempio esplicativo che ci spiega l'uso delle feature strutturali, potrebbe essere questo: La videocamera di sorveglianza riprende il quadro (o una qualsivoglia opera d'arte) e determina una struttura costituita da nodi che sono stabili nel tempo, non appena uno di questi nodi, non è più presente nella scena il primo estrattore notifica l'anomalia agli strati successivi, ma se durante l'intervallo di frame che compongono la mediana nella zona dove è scomparso il nodo non ci sono stati cambiamenti, il sistema non genera nessun allarme. In questo modo riusciamo a ritrovare tipiche situazioni di

furto ma anche di vandalismo o qualsiasi altro tipo di azione, volta a danneggiare l'opera o l'ambiente in questione.

Figura 30 (Estrattore tramite trasformata di Hough).

Figura 31 (Differenza in valore assoluto dalla mediana di immagini).

3.2.3 Livello Intermedio

Una caratteristica che rende il sistema progettato innovativo, è proprio la sua capacità di fondere insieme dati, provenienti da tre differenti punti di vista della scena videosorveglianza, da qui l'importanza di questo livello che è deputato proprio ad unificare i dati provenienti dai tre estrattori.

Il livello centrale *o middle layer* si compone di una *griglia logica* costruita sul frame corrente. Tramite questa griglia dividiamo l'immagine in 25 caselle formate da 80X58 pixel ciascuna, tutte etichettate tramite un *identificatore numerico (ID)*. Queste caselle servono per riuscire a mantenere un conteggio separato per ciascuno dei tre analizzatori. Ogni qualvolta un pixel viene rilevato da un estrattore il conteggio relativo a quest'ultimo nella casella di appartenenza del pixel viene incrementato. La griglia o tabella viene realizzata fisicamente come array multidimensionale dove la prima dimensione rappresenta le 25 caselle, l'altra formata da 3 posizioni serve a mantenere il conteggio degli estrattori su ciascuna casella. Si genera così sull'immagine *una mappa di informazioni* sulle aree maggiormente stabili o instabili , informazioni sulla quale poi un super decisore dovrà stabilire se siamo in presenza o meno di una situazione potenzialmente pericolosa.

3.2.4 Supervisore

Il supervisore è lo stadio finale del sistema; esso ha il compito di raccogliere tutte le informazioni provenienti dallo stadio intermedio ed analizzarle per formulare una decisione sulla situazione ambientale osservata. Il supervisore del sistema si basa su una grammatica il cui alfabeto è composto da nove caselle (il video ripreso è stato suddiviso in 9 riquadri) e dalle condizioni ambientali su di ognuna verificate. Il linguaggio formale che è stato costruito si basa su cicli condizionali "*If Then*" che valutano in modo opportuno le caselle e la condizione a ciascuna di esse associata.

Una *grammatica formale* è una struttura astratta che descrive un linguaggio formale in modo preciso, è cioè un sistema di regole che delineano matematicamente un sistema (di solito infinito) di lunghezze finite (stringhe) usando un alfabeto (di solito finito). Le grammatiche formali sono chiamate così per analogia con la grammatica delle lingue umane. Per linguaggio formale si intende un insieme di stringhe di lunghezza finita costruite sopra un alfabeto finito, cioè sopra un insieme finito di oggetti tendenzialmente semplici che vengono chiamati simboli, che nel nostro caso sono appunto le caselle e lo stato di ciascuna. In particolare l'unione di ciascuna casella con lo stato ad essa associato e con lo stato delle altre caselle ci permette di ottenere le stringhe. L' insieme di queste stringhe permette di ottenere il linguaggio che sta alla base del sistema decisionale di quest'ultimo livello. Quindi il superdecisore analizzando le stringhe così formatesi in base alle regole del linguaggio riesce a valutare la situazione globale. Infatti analizzando tutte le combinazioni di stringhe possibili (appartenenti al linguaggio) il supervisore va a confrontarle con delle clausole condizionali che gli permettono di trarre un decisione globale sullo stato del sistema.

Il supervisore risulta abbastanza funzionale, anche se potrebbe essere migliorato soprattutto utilizzando altre grammatiche che potrebbero funzionare meglio. Infatti questa tesi è ancora in evoluzione per quanto riguarda l'aspetto del decisore globale e un altro tesista sta continuando il lavoro fin qui svolto per migliorarlo. La decisone di utilizzare nove caselle e non sedici oppure quattro deriva da un'intensa attività sperimentale. Quest'attività ci ha permesso, dopo aver testato i vari casi, di definire nove caselle il miglior compromesso tra accuratezza di osservazione e semplicità decisionale.

CAPITOLO 4

RISULTATI SPERIMENTALI

Il sistema di videosorveglianza intelligente proposto in questa tesi è applicabile a molte situazioni di videosorveglianza, ma è stato principalmente pensato come supporto "intelligente" nel controllare la sicurezza di musei e pinacoteche; quindi è venuto spontaneo effettuare dei test proprio in questa tipologia di ambienti. Il museo scelto per effettuare delle prove è stato il museo degli Innocenti di Firenze, museo che contiene molte opere d'arte, tutte di notevole valore. Nel museo non potendo lavorare direttamente con il sistema e la videocamera per motivi tecnici si è scelto di riprendere tramite una videocamera un'ora e mezza della "vita" di questo ambiente in una sala dove sono contenute parecchie opere d'arte. Sono stati ripresi gli spostamenti della gente, il via vai del custode e tutti gli altri movimenti che il sistema progettato può trovarsi di fronte nella normale attività quotidiana; attraverso questo filmato sono stati poi realizzati esperimenti volti a raffinare il sistema.

Ulteriori esperimenti sono stati effettuati tramite una Webcam Creative direttamente collegata al software; tramite questa serie di esperimenti sono state valutate le capacità real-time del sistema, dimostrandosi un sistema molto efficiente e quasi sempre all'altezza della situazione.

Parlare di risultati in una tesi così sperimentale e che riguarda la rilevazione di situazioni pericolose intese in senso del tutto generale non è molto semplice, anzi si rivela assai complesso. Comunque il sistema ha reagito molto bene alla serie di filmati realizzati al museo degli Innocenti, mostrando le situazioni dove qualche turista mostrava troppa insistenza nello stare davanti ai quadri oppure si avvicinava troppo all'area del quadro. Buoni risultati sono stati rilevati anche in quelle situazioni dove per simulare il furto del quadro e quindi il suo spostamento, veniva spostata la

videocamera, dopo pochi secondi il software reagiva facendo suonare l'allarme di rilevazione di pericolo. Negli esperimenti effettuati in real-time tramite la webcam, si visto che sebbene la reattività del sistema non fosse paragonabile all'analisi dei filmati già ripresi, comunque in ambito di uso comune poteva dimostrarsi ancora più che soddisfacente.

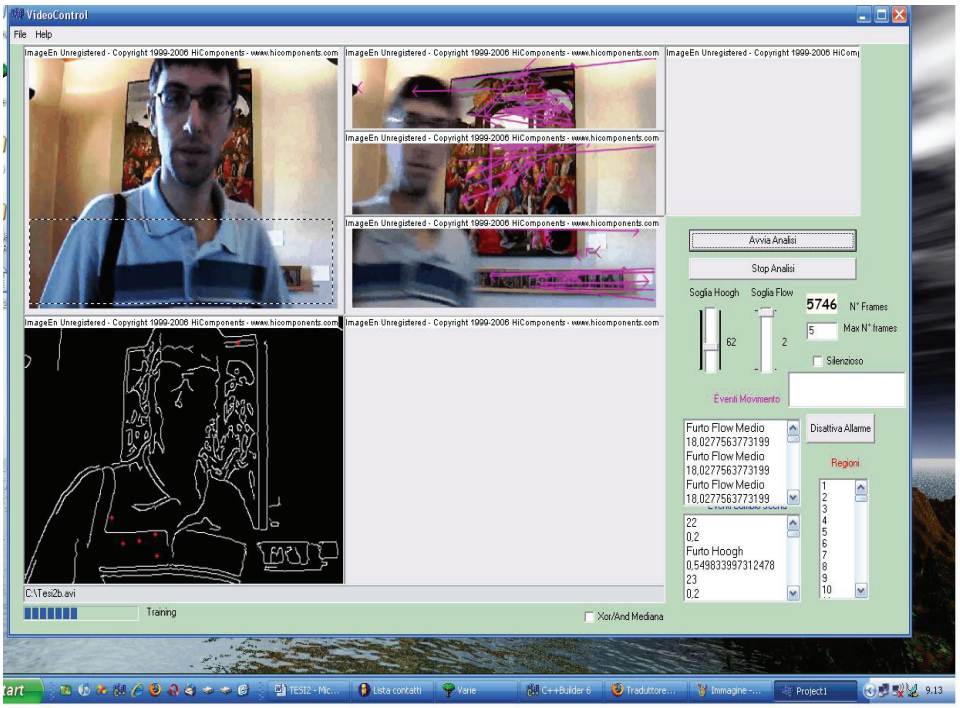


Figura 32

Nella figura 32 si vede in che modo il sistema lavora su un filmato ripreso al museo dove un mio collega si è davanti al quadro insistentemente, ma ancora non è stata calcolata la mediana di immagini. Si può vedere l'interfaccia nel suo complesso comprendente le quattro finestre di visualizzazione.

Nella prima in alto a sinistra si vede il video ripreso; in quella accanto vediamo l'applicazione dell'optical flow con le frecce che indicano la direzione e l'intensità (lunghezza freccia) del moto. In quella in basso a sinistra invece sono visualizzati gli edge e si possono ben vedere i contorni del soggetto ripreso. Nell'ultima ancora non è visualizzato nulla perché la mediana come detto non è ancora stata calcolata.

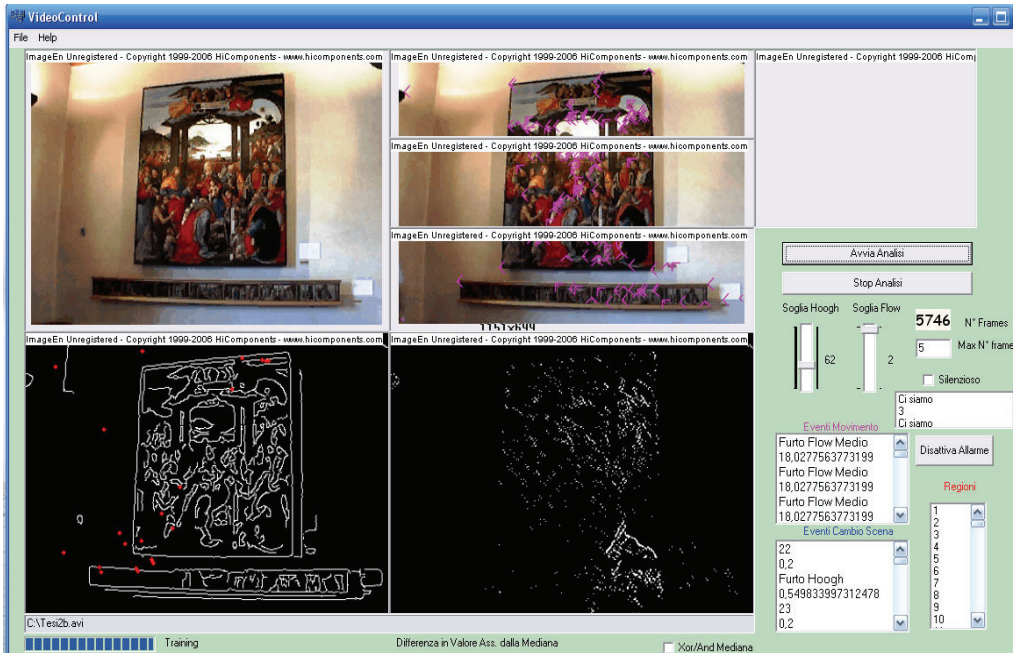


Figura 33

Nell'immagine 33 la mediana è stata calcolata e viene mostrata. Il sistema sta rilevando il prolungato movimento del mio collega davanti al quadro come azione potenzialmente pericolosa.

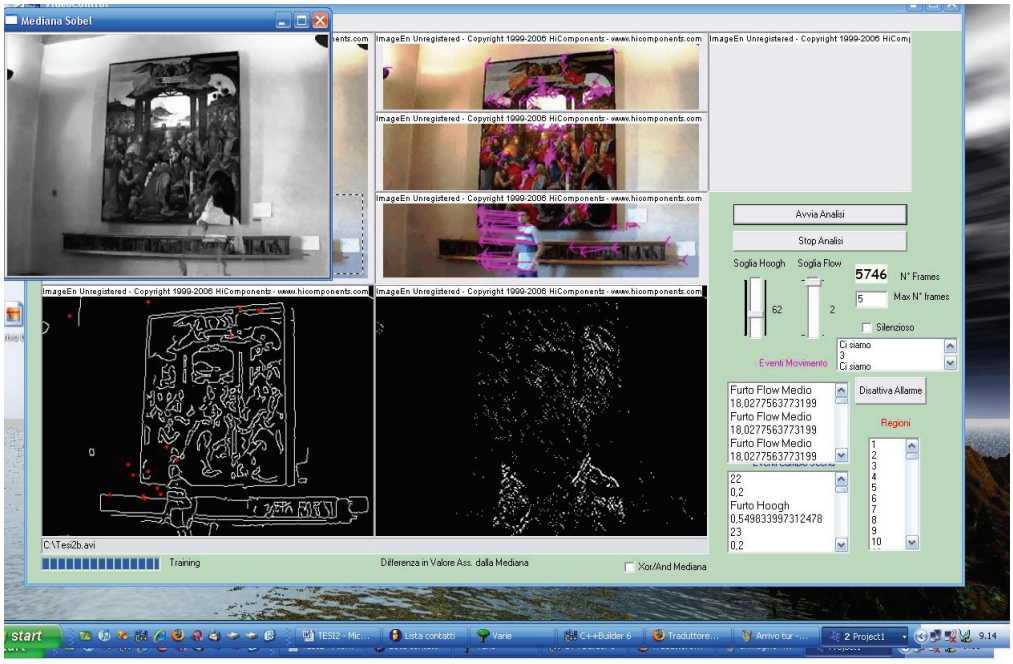


Figura 34

Nell'ultima figura si può avere una visione generale del sistema sviluppato mentre lavora su un filmato del museo degli Innocenti con tutte le sue finestre in funzione.

CONCLUSIONI

Nel corso di questi capitoli è stata presentata l'architettura di un sistema informatico intelligente di videosorveglianza. Il sistema ha il compito di riconoscere in tempo reale se particolari situazioni possono essere classificate come normali o d'allarme a seconda della "configurazione di normalità" settata in precedenza dopo una breve fase di training sull'ambiente da osservare. Nonostante il sistema sia abbastanza generalizzabile alle più disparate situazioni, per la nostra sperimentazione si è deciso di applicarlo al museo degli Innocenti di Firenze per l'osservazione delle stupende opere d'arte in esso contenute. Quindi dopo una configurazione iniziale (fase di training) il sistema riesce in seguito, a riconoscere in modo del tutto autonomo se le opere d'arte osservate, sono in situazioni che non rientrano nelle condizioni stabili. Il sistema per esempio riesce con molta facilità a distinguere se un particolare quadro è stato spostato oppure se ha subito delle alterazioni come un taglio. Dai dati sperimentali il prodotto risulta molto efficiente, riuscendo a distinguere molto facilmente situazioni di normalità che invece altri sistemi meno "intelligenti" avrebbero classificato come situazioni "anormali" facendo scattare inutili falsi allarmi. E soprattutto il sistema risulta molto generalizzabile, dopo una fase iniziale di configurazione per adattarsi. Sviluppi futuri che possono essere effettuati ,riguardano principalmente il livello decisionale che può essere ottimizzato, affiancando la grammatica realizzata a regole, con un paradigma di *self-learning* più complesso come le Reti Neurali, oppure cercando di migliorare fortemente le regole che compongono la grammatica già presente nel sistema. Altri sviluppi futuri possono migliorare la velocità del sistema specialmente in tempo reale, si potrebbe per esempio implementare una soluzione multiprocessore che renderebbe meno stringente, l'ottimizzazione del codice e degli algoritmi da utilizzare.

BIBLIOGRAFIA

1. Flavio Bernardotti, “Progettazione hardware e software di sistemi di sicurezza intelligenti”, 2006.

2. HiComponents, “ImageEn 2.2.4 Imaging Components “, *Guida*, 2006.

3. Intel® Corporation, “Open Source Computer Vision Library*”, Reference Manual*, 2001

4. Intel® Corporation, “Intel® Open Source Computer Vision Library”, *OpenCV 0.9.5*, 2001.

5. Stéphane Marchand-Mailet, Yazid M. Sharaida, “Binary Digital Image Processing”, *Academic Press*, 2000.

6. Jèrome Landré, “Programming with Intel IPP (Integrated Performance Primitives) and Intel OpenCV (Open Computer Vision) Under gnu Linux”, 2003.

7. K. Akita, "Image sequence analysis of real world human motion", *Pattern Recognition*, 17(1), 1984, pp.73-83.

8. Saverio Zuccotti, Tesi: “Sistema Elettronico Intelligente di Videosorveglianza Indoor per Tracciamento di Persone”, 2003.

9. Probiq, “Borland C++ Builder 5.0 Manual”, 2002.

10. Borland Software Corporation, “Borland C++ Builder 6 per Windows”, *Guida alla programmazione.* 2002.

11. D. M. Gavrila, J. Giebel and S. Munder, “Vision-based Pedestrian Detection: the PROTECTOR+ System”, *Proc. of the IEEE Intelligent Vehicles Symposium*, 2004.

12. D. M. Gavrila, “The Visual Analysis of Human Movement: A Survey”, *Computer Vision and Image Understanding, Academic Press*, vol. 73, nr. 1, pp. 82-98, 1999.