UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXII Ciclo

Studio e applicazione di algoritmi stereoscopici per la ricostruzione tridimensionale in ambiente automotive

Coordinatore:

Chiar.mo Prof. Carlo Morandi

Tutor:

Chiar.mo Prof. Alberto Broggi

Dottorando: *Mirko Felisa*

GENNAIO 2010

Alla mia famiglia
e ai miei amici

Sommario

Elenco delle figure

Elenco delle tabelle

Introduzione

Ogni giorno le strade e autostrade del mondo sono teatro di incidenti, che causano un numero di morti e feriti molto elevato, con costi sociali ed economici notevoli. L'Unione Europea [1] stima che nel solo anno 2008 sul suo territorio siano avvenuti 1,3 milioni di incidenti stradali con 39.000 morti e 1,6 milioni di feriti di cui 300.000 gravi. I costo diretto e indiretto annuo degli incidenti stradali è pari a 180.000 miliardi di euro: il 2% del prodotto interno lordo dell'intera europa. Uno studio [2] del 2006 quantifica rispettivamente in 1.018.200€, 143.100€ e 23.100€ l'impatto economico di una morte, un ferimento grave ed uno lieve.

Per ridurre questo costo, la Comunità Europea si è posta l'obiettivo di dimezzare, a partire dal 2001, il numero di morti annuo in incidenti stradali entro il 2010. In figura 1, sono raffigurati i dati dei decessi avvenuti dal 1990 al 2008: come si può osservare l'andamento è sicuramente positivo, ma non è ancora sufficiente per raggiungere l'obiettivo prefissato. Il ritardo accumulato, dovuto ad un tasso di riduzione annuo del 4.4% invece del 7.7% necessario, probabilmente non consentirà di raggiungere l'obiettivo prima del 2017.

Negli ultimi anni sono stati sviluppati diversi sistemi tecnologici per aumentare la sicurezza dei nostri veicoli. Questi sistemi possono essere suddivisi in due categorie: quelli che riducono il danno di un incidente (come le cinture di sicurezza e l'airbag) e quelli che cercano di evitarlo (come l'ESP e l'ABS).

La Comunità Europea si è dimostrata molto sensibile su questo argomento, infatti molti di questi sistemi sono obbligatori per legge come l'ABS e altri lo saranno nel breve periodo. Dal 2013, per esempio, su tutti i veicoli pesanti saranno obbligato-

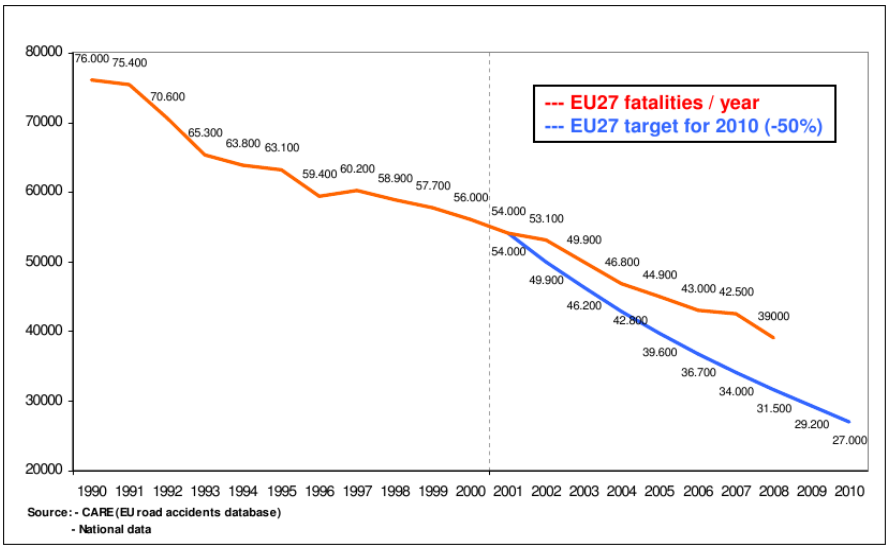


Figura 1: L'andamento dei decessi dovuti ad incidenti nel territorio europeo.

ri [3] due nuove tipologie di sensori: i primi denominati *Advance Emergency Braking Systems* (AEBS) sono in grado di avvertire il guidatore quando è troppo vicino al veicolo di fronte, ed in determinate situazioni, effettuare una frenata di emergenza per prevenire o quanto meno ridurre le conseguenze di una collisione e il secondo è un sistema di *Lane Departure Warning* in grado di avvisare il conducente in caso di uscita di carreggiata.

Il futuro degli ADAS (*Advanced driver assistance systems*) è rappresentato dalla costruzione di sistemi con la capacità di percepire l'ambiente circostante e l'intelligenza necessaria per decidere come e quando intervenire per aiutare il guidatore in caso di pericolo. Per questo genere di applicazioni, il sensore visivo appare particolarmente adatto, permettendo la segmentazione di scene complesse, l'individuazione e la classificazione di oggetti, grazie alla ricchezza delle informazioni fornite da ogni singolo fotogramma. Inoltre le telecamere, rispetto ai sensori attivi come laserscanner o radar, sono poco costose, robuste, non hanno parti in movimento e non emettono segnali che interferiscono con l'ambiente circostante.

Attività di ricerca volte a prevenire o almeno attenuare i danni dovuti ad incidenti stradali sono sicuramente meritevoli di attenzione. Questa tesi si pone l'obiettivo di sviluppare nuovi algoritmi di visione stereoscopica per la ricostruzione tridimensionale dell'ambiente circostante indicati per essere usati in ambiente *automotive*. Questi algoritmi vogliono rappresentare la parte di basso livello su cui costruire successivamente diversi sistemi ADAS. Sviluppare sistemi di assistenza alla guida significa dover elaborare i dati acquisiti in tempo reale e la generazione di immagini di disparità dense è un processo costoso in termini computazioni, quindi occorre sviluppare algoritmi *ad hoc* per il campo di utilizzo in grado di sfruttare tutte le informazioni disponibili. I vincoli sui tempi di esecuzione, infatti, orientano la scelta su un algoritmo che operi confronti locali, mentre la disponibilità delle informazioni sull'andamento del terreno consente di limitare il campo di ricerca, aumentando la correttezza dei risultati forniti. L'affidabilità dei valori di disparità restituiti può essere ulteriormente migliorata dall'applicazione di un insieme di filtri, che hanno lo scopo di riconoscere ed eliminare gli abbinamenti che con buona probabilità sono da considerare errati.

Infine sarà presentato un sistema che sfrutta tali algoritmi per la localizzazione di pedoni. L'individuazione automatica delle figure umane è un problema molto complesso a causa della difficoltà di individuare delle caratteristiche invarianti; allo stesso tempo lo sviluppo di un sistema, da installare a bordo dei veicoli, in grado di evitare la collisione con i pedoni, permetterebbe di salvare molte vite umane: basti pensare che i pedoni rappresentano 19,8% dei morti in incidenti stradali e tale percentuale sale al 36,35% all'interno dei centri urbani [1].

Questa tesi è organizzata nel seguente modo. Nel capitolo 1 saranno presentati alcuni algoritmi ottimizzati, orientati all'ambiente *automotive*, per la generazione di immagini di disparità ad elevati framerate. Nel capitolo 2, sarà presentata la trasformata di *Inverse Perspective Mapping* con le sue proprietà e sarà mostrato il risultato notevole di poter fornire una ricostruzione tridimensionale della scena vista da più telecamere. Infine nell'appendice sarà presentata un'applicazione di individuazione pedoni basata su quattro telecamere operanti nel visibile e nel lontano infrarosso.

Capitolo 1

Algoritmi di visione stereoscopica

1.1 Introduzione

La generazione di immagini di profondità a partire da una coppia di immagini stereoscopiche è un compito molto complesso, che spesso è reso più difficile da ulteriori vincoli imposti dall'ambito di utilizzo. Nel campo *automotive*, per esempio, una ricostruzione 3D dell'ambiente circostante in tempo reale è essenziale per sistemi di navigazione autonoma o di assistenza alla guida (ADAS). Questi sistemi richiedono tempi di elaborazione molto ristretti, che spesso vengono raggiunti a spese della precisione dei risultati ottenuti. Tuttavia le conoscenze *a priori* sull'ambito di utilizzo dell'algoritmo possono essere usate per aumentare le performance dello stesso sia in termini di precisione, sia di carico computazionale.

In letteratura sono presenti diversi algoritmi per la generazione di immagini di disparità che utilizzano approcci globali al problema che permettono di raggiungere un grado di precisione, calcolato su immagini di riferimento, molto elevato, con un tempo di elaborazione nell'ordine di decine di secondi su hardware molto potenti. Risulta evidente che questa categoria di algoritmi non è adatta ad essere utilizzata in applicazioni *automotive* dove i sistemi devono interagire con un ambiente estremamente dinamico. Per rispettare il vincolo di elaborazione in tempo reale si è scelto di utilizzare un approccio locale basato sulla correlazione. Questa categoria include

diversi algoritmi; questo studio si è concentrato su una tecnica di elaborazione incrementale basata sulla metrica di correlazione SAD (somma dei valori assoluti delle differenze). L'idea di fondo di questo approccio e i relativi vantaggi in termini di tempi di elaborazione sono già stati investigati prima in [4] e poi in [5]; in questo studio sono presentati tre nuovi algoritmi che, sfruttando le proprietà di un sistema stereo installato su un veicolo, generano immagini di disparità dense ad elevati framerate su *commodity hardware* sfruttando il parallelismo e le istruzioni SIMD[1] delle moderne architetture multicore.

Figura 1.1: Il veicolo di test: in alto, evidenziato in rosso, il sistema di visione stereo montato sul parabrezza, sotto una vista ravvicinata della telecamera destra.

[1] Single Instruction Multiple Data

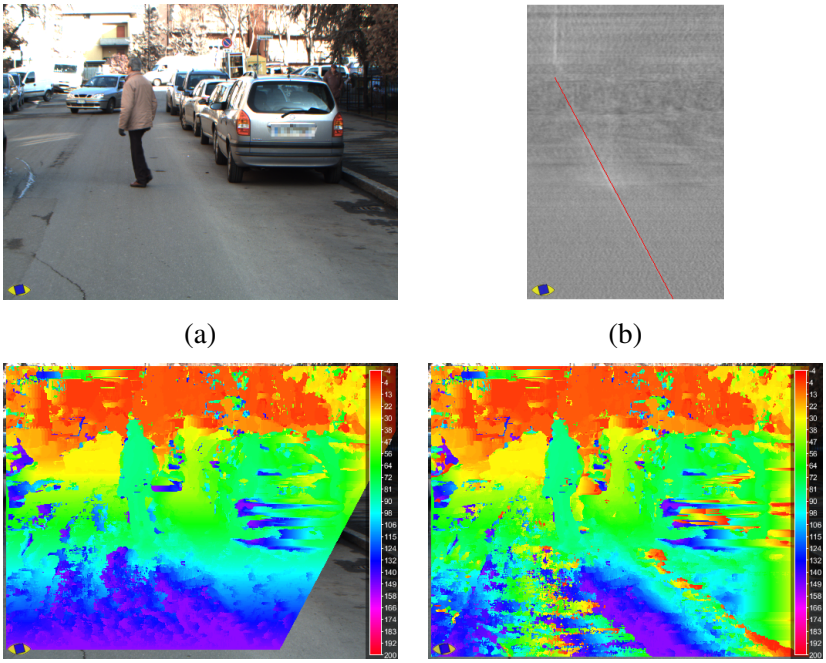
Usando un sistema di visione stereo come quello raffigurato in figura 1.1, sia in ambiente urbano che non strutturato, si nota che il terreno di fronte al veicolo occupa una porzione molto grande dell'immagine; questa situazione può essere modellata usando un approccio basato sulla V-disparity Image [6]. Analizzando questa immagine, come descritto in [7], è possibile estrarre una stima della disparità del terreno per ogni riga dell'immagine. Si può, inoltre, osservare che ogni punto di un ostacolo ha una disparità pari a quella del terreno sottostante e quindi avrà una disparità maggiore o uguale a quella del terreno corrispondente alla medesima riga dell'immagine. Questa considerazione, insieme al fatto che nelle applicazioni *automotive* l'angolo di *roll* del veicolo può essere annullato rettificando le immagini, permette di limitare l'intervallo di ricerca delle finestre omologhe a disparità maggiori di quella del terreno nella medesima riga dell'immagine. Questo intervallo di ricerca può essere leggermente allargato includendo alcune disparità sotto il terreno per aumentare la robustezza del sistema ad eventuali imprecisioni nel determinare il profilo del terreno.

In modo analogo è possibile imporre un limite al bordo superiore dell'intervallo di ricerca. A seconda del tipo di applicazione, è possibile tenerlo fisso per ogni punto dell'immagine, oppure rendere costante la lunghezza dell'intervallo stesso. In questo ultimo caso è possibile che la parte superiore degli ostacoli, oltre una determinata altezza dal suolo, non venga individuata.

I benefici di una riduzione dell'intervallo di ricerca sono, in primo luogo, una riduzione del tempo di elaborazione, in quanto diminuisce il numero di correlazioni richieste, ma anche un aumento della qualità dell'immagine risultante, in quanto le correlazioni errate dovute a valori di disparità inaccettabili vengono automaticamente evitate. In figura 1.2 sono confrontate due mappe di disparità generate a partire dalla stessa coppia di immagini sorgenti: la prima è stata ottenuta utilizzando la posizione del terreno, nella seconda si è utilizzato un intervallo di ricerca costante. Come si può osservare, quest'ultima presenta, soprattutto nella parte inferiore dell'immagine, un numero di valori di disparità errati superiore alla prima.

Sebbene la combinazione di un approccio incrementale con una limitazione degli intervalli di ricerca è invitante, essa richiede lo sviluppo di algoritmi complessi

che devono trattare un elevato numero di casi di bordo per innescare correttamente l'algoritmo incrementale.



(a) (b)

(c) (d)

Figura 1.2: La generazione dell'immagine di disparità (DSI): (a) immagine destra (512×384 pixel), (b) V-Disparity image con il profilo del terreno evidenziato in rosso, (c) immagine di disparità con intervallo $[d_{gnd}(i), 150] \forall i \in [0, 384[$ usando finestre di correlazione 8×8 pixel e 1.12 m di *baseline*; (d) immagine di disparità con intervallo $[0, 150] \forall i \in [0, 384[$. La parte di DSI mancante in (c) corrisponde ai punti dell'immagine destra il cui intervallo di ricerca cade oltre il bordo destro dell'immagine sinistra.

1.2 Algoritmi

Data una coppia di immagini stereo **L** e **R** (sinistra e destra rispettivamente) di dimensione $width \times height$, l'immagine di disparità DSI può essere generata iterando su ogni riga e ogni colonna di **R** ed effettuando una corrrelazione di una finestra centrata attorno ad ogni punto con un insieme di finestre posizionate lungo la retta epipolare corrispondente dell'immagine **L**; il valore di disparità sarà calcolato come

$$\underset{d \in [d_{min}, d_{max}]}{\arg\min} \; SAD(x, y, d) \tag{1.1}$$

L'uso della rettificazione [8, 9] delle immagini sorgenti permette di ottenere linee epipolari orizzontali, mentre per limitare l'area di ricerca viene usata una matrice **K** di dimensione $height \times 2$ che, per ogni riga, contiene il valore di disparità massima minima e massima ammmissibile.

Avendo scelto una metrica di correlazione di tipo SAD su finestre rettangolari di dimensione $(2m+1) \times (2n+1)$ il punteggio di correlazione tra due finestre centrate nel generico punto di coordinate (x, y) può essere calcolato come

$$SAD(x, y, d) \stackrel{\text{def}}{=} \sum_{i=-m}^{m} \sum_{j=-n}^{n} |\mathbf{R}[x+j, y+i] - \mathbf{L}[x+j+d, y+i]| \tag{1.2}$$

Nell'algoritmo 1 sono mostrati i passi necessari per la generazione dell'immagine di disparità. Quel che appare subito evidente è che, pur ottenendo risultati corretti, se si procede in questo modo la maggior parte dei calcoli viene ripetuta più volte inutilmente, con un decadimento delle prestazioni che si fa sempre più sensibile man mano che si aumentano le dimensioni delle finestre di confronto. Supponendo l'area di ricerca linearmente proporzionale alla larghezza dell'immagine, la complessità del suddetto algoritmo risulta essere $O(width^2 \cdot height \cdot win\_w \cdot win\_h)$. Utilizzando un approccio incrementale è possibile rendere il calcolo della DSI indipendente[2] dalla grandezza della finestra di correlazione.

[2] Il calcolo della disparità nel generico pixel risulta indipendente dalla dimensione della finestra; rimane una dipendenza per i pixel sul bordo dell'immagine, che comunque sono un numero trascurabile rispetto a quelli all'interno.

L'approccio più diretto all'applicazione dei concetti di limitazione dell'intervallo di ricerca e del calcolo incrementale della DSI, è presentato nel paragrafo 1.2.1 e deriva da [4] e [5]; il suo principale limite è rappresentato dalla sua inefficenza a gestire immagini sorgenti sparse (il calcolo della disparità in zone dell'immagine con poca *texture* è molto rumoroso, quindi può essere utile non procedere all'elaborazione in tali punti) in quanto ogni punto dipende dai suoi vicini.

Algorithm 1 Algoritmo base per il calcolo della DSI

```
for i ← 0 to height do
  for j ← 0 to width do
    best_score ← ∞
    best_disp ← NaN
    for d ← d_min[i] to d_max[i] do
      score ← 0
      for h ← -win_h/2 to win_h/2 do
        for k ← -win_w/2 to win_w/2 do
          score ← score + |R(i+h, j+k) - L(i+h, j+k+d)|
        end for
      end for
      if (score < best_score) then
        best_score← score
        best_disp← d
      end if
    end for
    if (best_score < ∞) then
      DSI(i, j) ← best_disp
    else
      DSI(i, j) ← DISPARITY_UNKNOWN
    end if
  end for
end for
```

Questo vincolo forza il calcolo di tutti i punti della DSI, anche se essi, successivamente, saranno scartati. Per superare questa limitazione è stato sviluppato l'algoritmo descritto nel paragrafo 1.2.2: il vincolo di inter-dipendenza tra i pixel è stato rilassato in modo tale che ogni pixel non dipendi più dai punti precedenti nella sua stessa riga; questo permette di saltare i valori di disparità non necessari al costo di non avere il tempo di elaborazione indipendente dalla larghezza della finestra di correlazione; inoltre diventa possibile usare un intervallo di ricerca variabile all'interno della stessa riga. L'utilizzo del primo o del secondo algoritmo dipende principalmente dal tipo di applicazione in cui lo si intende usare. Quest'ultimo approccio ha suggerito la riformulazione dell'algoritmo incrementale in modo tale che utilizzi un singolo confronto tra pixel per ogni disparità testata indipendentemente dalla dimensione della finestra, come descritto nel paragrafo 1.2.3.

1.2.1 Algoritmo incrementale

L'algoritmo incrementale riutilizza i valori di SAD utilizzati per calcolare la disparità in un dato punto per ottenere il valore di correlazione dei pixel adiacenti, sfruttando la sovrapposizione tra le finestre di correlazione. L'unico vincolo aggiuntivo rispetto l'algoritmo 1 è rappresentato dalla disparità minima del terreno che deve essere debolmente crescente andando dall'alto verso il basso:

$$\mathbf{K}[y,0] \geq \mathbf{K}[y-1,0] \,\forall\, y \in [\,1, \mathit{height}\,[\tag{1.3}$$

Questo vincolo, sempre soddisfatto nei sistemi *automotive*, è stato introdotto per ridurre il numero di casi di bordo da gestire e aumentare le prestazioni dell'algoritmo stesso.

È mostrato in [4, 5, 10] che l'equazione 1.2 può essere espressa in forma incrementale come

$$SAD(x,y,d) = SAD(x,y-1,d) + \Delta_y(x,y,d) \tag{1.4}$$

con

$$\Delta_y(x,y,d) \stackrel{\text{def}}{=} \sum_{j=-n}^{n} |\mathbf{R}[x+j,y+m] - \mathbf{L}[x+j+d,y+m]| - \sum_{j=-n}^{n} |\mathbf{R}[x+j,y-m-1] - \mathbf{L}[x+j+d,y-m-1]| \tag{1.5}$$

la quale può essere ulteriormente ridotta a

$$\Delta_y(x,y,d) = \Delta_y(x-1,y,d) - A(x,y,d) + B(x,y,d) \tag{1.6}$$

avendo

$$A(x,y,d) \stackrel{\text{def}}{=} |\mathbf{R}[x-n-1,y+m] - \mathbf{L}[x-n-1+d,y+m]| - |\mathbf{R}[x-n-1,y-m-1] - \mathbf{L}[x-n-1+d,y-m-1]| \tag{1.7}$$

e

$$B(x,y,d) \stackrel{\text{def}}{=} |\mathbf{R}[x+n,y+m] - \mathbf{L}[x+n+d,y+m]| - |\mathbf{R}[x+n,y-m-1] - \mathbf{L}[x+n+d,y-m-1]| \tag{1.8}$$

È possibile evitare il calcolo esplicito dei termini $A(x,y,d)$ e $B(x,y,d)$ in quanto ridondanti, osservando che $A(x,y,d) \equiv B(x-2n-1,y,d)$: sia $\mathbf{A_x}$ una matrice ausiliaria di dimensione $2n+1 \times max\_k$, con

$$max\_k \stackrel{\text{def}}{=} \max_{y \in [0,height[} (\mathbf{K}[y,1] - \mathbf{K}[y,0] + 1) \tag{1.9}$$

usata per memorizzare, alla posizione $(\bar{x},d)$, con $\bar{x} = x \mod (2n+1)$, il valore di $B(x,y,d)$; essa viene aggiornata mentre la finestra di correlazione si muove lungo una data riga di $\mathbf{R}$. Il valore di Δ_y può, quindi, essere espresso come

$$\Delta_y(x,y,d) = \mathbf{\Delta_y}[d] - \mathbf{A_x}[\bar{x},d] + B(x,y,d) \tag{1.10}$$

con $\mathbf{\Delta_y}$ un vettore di dimensione *max_k* contenente il valore $\Delta_y(x-1,y,d)$ alla posizione d. Combinando l'equazioni 1.4 e. 1.10 è infine possibile esprimere $SAD(x,y,d)$ come

$$SAD(x,y,d) = SAD(x,y-1,d) + \mathbf{\Delta_y}[d] - \mathbf{A_x}[\bar{x},d] + B(x,y,d) \tag{1.11}$$

Usando questa formula per generare la DSI è necessario gestire quattro differenti categorie di punti:

- il primo punto della prima riga, calcolato usando l'equazione 1.2;
- i punti seguenti della prima riga per i quali è valida la seguente relazione

$$\begin{aligned} SAD(x,y,d) = SAD(x-1,y,d) + \\ \sum_{i=-m}^{m} |\mathbf{R}[x+n,y+i] - \mathbf{L}[x+n+d,y+i]| - \\ \sum_{i=-m}^{m} |\mathbf{R}[x-n-1,y+i] - \mathbf{L}[x-n-1+d,y+i]| \end{aligned} \tag{1.12}$$

  Usando lo stesso principio sfruttato nell'equazione 1.10, non è necessario calcolare entrambi i termini incrementali;

- il primo punto della generica riga, calcolato usando le equazioni 1.4 e 1.5;
- il generico punto all'interno dell'immagine calcolato usando l'equazione 1.11.

I termini $\mathbf{\Delta_y}$ e $\mathbf{A_x}$ sono inizializzati quando si analizza la prima riga e il primo pixel della generica riga rispettivamente, e iterativamente aggiornati nel calcolo di ogni punto dell'immagine di disparità.

È necessario osservare che l'intervallo di ricerca si riduce quando la finestra base, posizionata sull'immagine destra, si avvicina ai bordi dell'immagine:

$$\begin{cases} d_{min}(x,y) = \max(n-x, \mathbf{K}[y,0]) \\ d_{max}(x,y) = \min((width-1-n-x, \mathbf{K}[y,1]) \end{cases} \tag{1.13}$$

Inoltre, non avendo imposto nessun vincolo sui valori di $\mathbf{K}[y,1]$, può accadere che

$$\mathbf{K}[y,1] < \mathbf{K}[y+1,1] \tag{1.14}$$

in questi casi non è più possibile utilizzare l'equazione 1.11 per ottenere i punteggi di SAD, a meno che non si proceda al calcolo dei termini $\Delta_y(x, y+1, d)$, $d \in [\, d_{max}(x,y),\, d_{max}(x,y+1)\,]$ alla riga y senza procedere alla generazione dei punteggi di correlazione $SAD(x,y,d)$, $d \in [\, d_{max}(x,y),\, d_{max}(x,y+1)\,]$ corrispondenti . Si noti che non è garantita l'esistenza dei $\Delta_y(x, y-1, d)$ in questo intervallo, quindi il metodo più diretto per procedere alla loro computazione è usare l'equazione 1.12. Questa soluzione non è sempre ottima, infatti se $\mathbf{K}[y-1,1] \geq \mathbf{K}[y+1,1] > \mathbf{K}[y,1]$ si potrebbe procedere in modo completamente incrementale; si è scelto di non muoversi in questa direzione perché un bordo destro siffatto non è presenti in casi pratici.

Per rendere il sistema più robusto ad errori nella stima del profilo del terreno, e per individuare ostacoli negativi, è auspicabile includere nell'intervallo di ricerca, per una data riga, alcune disparità inferiori a quella del terreno; permettendo questo, è necessario gestire il fatto che quando $\mathbf{K}[y,0] = \bar{d} < 0$ i primi $|\bar{d}|$ punti della riga non possono essere calcolati con l'equazione 1.11, in quanto i termini $\mathbf{A_x}[\bar{x}, d_{min}(x,y)]\, \forall x \in [\,1,\, |\bar{d}|\,[$ non sono disponibili. Questo accade perchè l'equazione 1.13 implica che

$$\max(n-x, \bar{d}) = n-x$$

essendo

$$n-x > \bar{d} \quad \forall x \in [\,1,\, |\bar{d}|\,[\text{ con } n > 0, \bar{d} < 0$$

con la conseguenza che

$$\max(n-x, \bar{d}) > \max(n-x-1, \bar{d}) \quad \forall x \in [\,1,\, |\bar{d}|\,[$$

ovvero

$$d_{min}(x,y) > d_{min}(x-1,y)$$

o più esattamente

$$d_{min}(x,y) = d_{min}(x-1,y) + 1$$

La soluzione è ancora una volta abbandonare l'approccio completamente incrementale, usando l'equazione 1.5.

1.2.2 Algoritmo semi-incrementale

La disparità, usando un approccio locale basato sulla correlazione, può essere calcolato in modo attendibile solo nelle parti dell'immagine con presenza di *texture*; nelle zone uniformi essa assume valori errati, per questo motivo, in molte applicazioni, tali valori vengono scartati (figura 1.3), al fine di ottenere un risultato più affidabile. Con l'approccio incrementale descritto in 1.2.1, non è possibile saltare il calcolo di alcuni pixel di disparità in quanto necessari al computo dei successivi. È possibile rimuovere questo vincolo di interdipendenza tra i pixel, usando solo le equazioni 1.4 e 1.5 per calcolare i valori di correlazione.

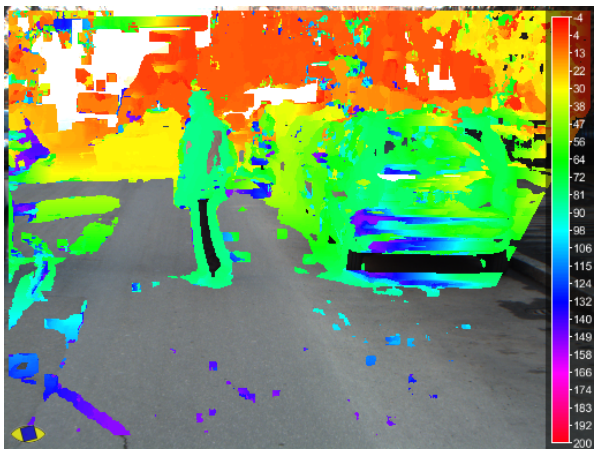


Figura 1.3: L'immagine di disparità con filtro sulla *texture*: le finestre con una $\sigma^2 < 30$ vengono scartate.

Questa strategia rende il costo computazionale dell'algoritmo dipendente dalla larghezza della finestra di correlazione ma, per finestre di normale utilizzo e per immagini di disparità non dense, questo algoritmo risulta più performante del precendente, specialmente sulle moderne CPU (che supportano l'insieme di istruzioni Intel®SSE 4.1) che permettono una implementazione SIMD molto efficiente. In quest'ultimo caso, grazie al parallelismo fornito dall'*hardware* è possibile rendere il tem-

po impiegato dall'algoritmo costante al variare della larghezza della finestra fino ad una dimensione di 16 pixel.

La strategia adottata nel derivare l'equazione 1.10 può essere adottata una seconda volta, usando una matrice $\mathbf{C_y}$ di dimensione $width \times 2m+1 \times max\_k$ per memorizzare i risultati intermedi così che

$$SAD(x,y,d) = SAD(x,y-1,d) - \mathbf{C_y}[x,\bar{y},d] + D(x,y,d) \tag{1.15}$$

con $\bar{y} = y \mod (2m+1)$ e

$$D(x,y,d) \stackrel{\text{def}}{=} \sum_{j=-n}^{n} |\mathbf{R}[x+j,y+m] - \mathbf{L}[x+j+d,y+m]| \tag{1.16}$$

Il processo di costruzione della DSI, in questo caso, implica solo due categorie di punti:

- i punti della prima riga, calcolati secondo l'equazione 1.2;
- i generici punti all'interno dell'immagine, calcolati sfruttando le equazioni 1.15 e 1.16.

Questo algoritmo richiede di soddisfare sia l'equazione 1.13 che i vincoli derivanti dalla 1.14. È possibile evitare il calcolo della SAD in un determinato punto solo se nessun altro pixel, sulla stessa colonna, m righe sopra e sotto ad esso, è richiesto. L'algoritmo per prima cosa si costruisce una maschera in cui assegna ogni punto dell'immagine di disparità ad una di queste tre categorie:

- punti che necessitano sia del valore di SAD che della minimizzazione dei punteggi di correlazione;
- punti che necessitano solo del valore di SAD per sostenere lo schema incrementale;
- punti che possono essere completamente saltati.

Dopo questa prima fase di classificazione, il processo di costruzione della DSI può partire come spiegato sopra.

1.2.3 Algoritmo completamente incrementale

L'algoritmo semi-incrementale presentato nel paragrafo 1.2.2 può essere trasformato in completamente incrementale, osservando che

$$D(x,y,d) = \mathbf{D_x}[d] - E(x,y,d) + F(x,y,d) \tag{1.17}$$

con $\mathbf{D_x}$ un vettore di dimensione *max_k* che, alla posizione d, memorizza il valore $D(x-1,y,d)$, mentre

$$E(x,y,d) \stackrel{\text{def}}{=} |\mathbf{R}[x-n-1,y+m] - \mathbf{L}[x-n-1+d,y+m]| \tag{1.18}$$

e

$$F(x,y,d) \stackrel{\text{def}}{=} |\mathbf{R}[x+n,y+m] - \mathbf{L}[x+n+d,y+m]| \tag{1.19}$$

Questo significa che

$$D(x,y,d) = \mathbf{D_x}[d] - \mathbf{E_x}[\bar{x},d] + F(x,y,d) \tag{1.20}$$

con $\bar{x} = x \mod (2n+1)$, ed $\mathbf{E_x}$ viene aggiornato con i valori di $F(x,y,d)$ man mano che sono calcolati.
Combinando l'equazione 1.20 con la 1.15, infine, si ottiene

$$\begin{aligned} SAD(x,y,d) = \\ SAD(x,y-1,d) - \mathbf{C_y}[x,\bar{y},d] + \\ \mathbf{D_x}[d] - \mathbf{E_x}[\bar{x},d] + F(x,y,d) \end{aligned} \tag{1.21}$$

che comporta che ogni SAD può essere ottenuta con un solo confronto tra pixel delle immagini sorgenti, corrispondente al termine $F(x,y,d)$, indipendentemente dalla dimensione della finestra. Si noti che il termine $\mathbf{C_y}$ viene iterativamente aggiornato con il risultato di $\mathbf{D_x}[d] - \mathbf{E_x}[\bar{x},d] + F(x,y,d)$.

Come accade per l'algoritmo presentato nel paragrafo 1.2.1, ci sono quattro categorie di punti da considerare:

- il primo punto della prima riga, calcolato come

$$SAD(x,y,d) = \sum_{i=-m}^{m} \mathbf{C_y}[x, i+m, d] \tag{1.22}$$

  con

$$\mathbf{C_y}[x, i+m, d] = \sum_{j=-n}^{n} |\mathbf{R}[x+j, y+i] - \mathbf{L}[x+j+d, y+i]| \tag{1.23}$$

- i punti seguenti della prima riga, ottenuti ancora con l'equazione 1.22, ma con $\mathbf{C_y}$ che assume i valori

$$\begin{aligned} &\mathbf{C_y}[x, i+m, d] = \\ &\quad \mathbf{C_y}[x-1, i+m, d] - \mathbf{E'_{xy}}[\bar{x}, i+m, d] + \\ &\quad |\mathbf{R}[x+n, y+i] - \mathbf{L}[x+n+d, y+i]| \end{aligned} \tag{1.24}$$

  con $\mathbf{E'_{xy}}$ una matrice di dimensione $2n+1 \times 2m+1 \times max\_k$ contenente i termini incrementali calcolati $2n+1$ colonne prima;

- il primo punto della generica riga, calcolato come

$$\begin{aligned} SAD(x,y,d) = SAD(x,y-1,d) - \\ \mathbf{C_y}[x, \bar{y}, d] + \sum_{j=-n}^{n} \mathbf{E_x}[j+n, d] \end{aligned} \tag{1.25}$$

  con

$$\mathbf{E_x}[j+n, d] = |\mathbf{R}[x+j, y+m] - \mathbf{L}[x+j+d, y+m]| \tag{1.26}$$

- i generici punti all'interno dell'immagine, calcolati usando l'equazione 1.21.

L'uso delle equazioni 1.22, 1.24 e 1.25 permette sia di ottenere il valore corretto di SAD, sia di inizializzare i buffer ausiliari. I casi di bordo presentati nel paragrafo 1.2.1

vengono gestiti in modo differente rispetto al vecchio algoritmo. L'effetto dell'equazione 1.14 è compensato dall'estensione del calcolo dei valori di SAD usando l'equazione 1.21 oltre $\mathbf{K}[y,1]$ fino al $\max_{i\in[0,2m+1]}(\mathbf{K}[y+i,1])$, per riempire $\mathbf{C_y}$ con i valori necessari alle righe successive; si noti, comunque che questi valori di correlazione, oltre l'intervallo di ricerca, non sono considerati nella fase di minimizzazione. Le disparità negative all'inizio della riga sono compensate applicando l'equazione 1.15, che non richiede il termine incrementale $(x-1,y,d)$, che non è disponibile, come spiegato precedentemente.

1.3 Filtraggi

Durante il calcolo dell'immagine di disparità si incontrano alcune regioni che più di altre sono propense a dar luogo ad abbinamenti errati, portando dunque ad una non corretta stima del valore di disparità associato. Gli algoritmi per la generazione della DSI presentati in questa tesi sono di tipo locale, in quanto analizzano soltanto su un intorno (nel nostro caso, una finestra $2m+1\times 2n+1$) del punto in esame, e quindi sono molto sensibili alla quantità di informazione presente nella finestra stessa. Aree prive di *texture* producono risultati errati, in quanto i punteggi associati ai diversi valori di disparità risultano molto simili tra loro: in questi casi, è preferibile scartare il risultato piuttosto che rischiare di fornirne uno errato. Una metrica che permette di identificare queste zone è la varianza della finestra di correlazione: valori troppo bassi indicano pixel con poco contenuto informativo, e dunque da scartare. Nel caso le immagini in ingresso siano state preelaborate con qualche tipo di filtro derivativo è anche possibile impiegare una metrica ancora più semplice: la somma dei valori dei pixel appartenenti alla finestra in esame, che, ancora una volta, deve superare una certa soglia per considerare attendibile il risultato. Entrambe le soluzioni presentate possono essere implementate con uno schema incrementale simile a quello visto per il calcolo dei valori di SAD.

Per verificare la consistenza dei risultati ottenuti è possibile calcolare una seconda mappa di disparità, invertendo le due immagini in ingresso e cambiando di segno all'intervallo di ricerca, e verificare che, per ogni punto, le due disparità siano uguali

e invertite di segno. Questo approccio comporta un costo elevato in termini di tempo di elaborazione; in alternativa è possibile utilizzare, con un costo computazionale limitato, l'algoritmo proposto in [5] denominato *Single Matching Phase* (SMP), che prevede di scartare l'abbinamento col punteggio di SAD maggiore quando si rileva che

$$x_1 + DSI(x_1, y) = x_2 + DSI(x_2, y)$$

ossia nel caso in cui due finestre di riferimento (immagine destra) risultino mappate sulla stessa finestra di destinazione (immagine sinistra); tale fenomeno, privo di significato fisico, è sintomo un falso abbinamento, dovuto in genere alla presenza di condizioni di illuminazione avverse, rumore o occlusioni.

L'analisi della distribuzione dei punteggi di SAD di una finestra per diversi valori di disparità permette di irrobustire ulteriormente i risultati, andando ad individuare situazioni che solitamente portano ad abbinamenti errati. Per prima cosa si individuano il minimo della funzione di correlazione ed altri 3 pseudo-minimi, ottenuti confrontando tra loro i valori a gruppi di 4 elementi per volta. Si procede poi col calcolo di alcune grandezze [5]:

$$\Delta d = \sum_{i=1}^{3} |d(i) - d_{min}|$$

$$\Delta SAD = \sum_{i=1}^{3} |SAD(i) - SAD_{min}|$$

Δd viene detta *sharpness*, e rappresenta una misura della dispersione dei minimi locali attorno al minimo globale: valori alti indicano picchi dispersi, e dunque abbinamenti potenzialmente ambigui. Nel caso si verifichi questa evenienza, si procede misurando la *distinctiveness* associata al punto, ossia il valore $\Delta SAD / SAD_{min}$: più è alto, più il picco risulta marcato, e dunque affidabile. L'ultimo controllo effettuato riguarda il numero minimo di elementi su cui viene effettuata la minimizzazione: risultati ottenuti da funzioni di correlazione riguardanti un intervallo di valori limitato sono infatti da scartare perché affetti da rumore.

La disparità ottenuta minimizzando la funzione di correlazione è per costruzione un valore intero. È possibile ottenere un valore di disparità con *sub-pixel accuracy*

approssimando localmente la funzione di correlazione con una curva del secondo ordine, di cui viene restituito il minimo:

$$d = d_{min} + \frac{SAD(d_{min}-1) - SAD(d_{min}+1)}{2\left(SAD(d_{min}-1) + SAD(d_{min}+1) - 2SAD(d_{min})\right)}$$

1.4 Implementazione

Nell'implementare gli algoritmi precedentemente descritti si è deciso di massimizzare il più possibile la modularità e il riuso del codice scritto, con lo scopo di realizzare un libreria di visione stereo completa adatta a funzionare in diverse configurazioni. A tal fine, si è scelto di utilizzare il linguaggio C++, con un design basato su *policy classes* [11], unito a sezioni in *assembly* per sfruttare al meglio le istruzioni SIMD presenti in vari processori, che permettono, per un determinato pixel, di calcolare contemporaneamente il punteggio associato a più valori di disparità. La disponibilità di sistemi con più *core* permette inoltre di ripartire l'elaborazione in modo automatico su più *thread* paralleli (ciascuno operante su una differente porzione orizzontale dell'immagine), riducendo in maniera consistente i tempi di esecuzione. La libreria sviluppata permette all'utente di scegliere:

- la funzione di correlazione (SAD, SSD);
- la dimensione massima del punteggio di correlazione (16, 32 bit);
- il numero di *thread* di elaborazione;
- il tipo di immagini di ingresso (monocromatiche, colore, con o senza segno);
- l'algoritmo utilizzato (incrementale, semi-incrementale)
- l'implementazione utilizzata (C++, SIMD, GPU)

Il design basato su *policy classes* è risultato decisivo nello sviluppo della libreria, infatti ha evitato di dovere implementare tutte le possibili combinazioni delle caratteristiche precedenti rendendole ortogonali tra di loro, massimizzando il riuso del codice.

Verranno ora descritte le implementazioni SIMD dei vari algoritmi nel caso di immagini in ingresso monocromatiche (8 bit per pixel).

1.4.1 Algoritmo completamente incrementale

L'implementazione SIMD dell'algoritmo non prevede la riscrittura completa in *assembly* del codice descritto nel paragrafo 1.2.3, ma l'ottimizzazione di una piccola parte. I compilatori moderni, infatti, generano codice di ottima qualità, quindi risulta privo di senso riscrivere completamente l'algoritmo nel nuovo linguaggio in quanto il beneficio sarebbe minimo. All'opposto è opportuno riscrivere la porzione di codice chiamata più di frequente ed più adatta alla vettorizzazione in modo da ottenere un significativo vantaggio in termini di tempo di elaborazione. Nell'algoritmo seguente è illustrata la porzione di codice da ottimizzare che permette di ricavare il valore di $SAD(x,y,d)$ per un generico punto all'interno dell'immagine.

Algorithm 2 Implementazione non ottimizzata dell'algoritmo full-incremental

for d =d_{min} **to** d <=d_{max} **do**
  corr_delta = prev_pixel_corr[d] + |pL1[d] - R1| - |pL2[d] - R2|
  corr_sum[d] += corr_delta - corr_value[d]
  corr_value[d] = corr_delta
end for

Il buffer *corr_sum* contiene le correlazioni per le varie disparità della riga precedente, il vettore *corr_value* rappresenta il termine $\mathbf{C_y}$ e *prev_pixel_corr* è un puntatore a *corr_value* del punto precedente.

Gli algoritmi 3 e 4 descrivono la porzione di codice ottimizzata corrispondente, suddivisa in funzioni; ognuna delle quali riporta il codice *assembly* SIMD[3], per processori che supportano le istruzioni SSSE 3.

L'uso di istruzioni SIMD permette di calcolare 8 disparità contemporanemente, come si può osservare nell'algoritmo 4, con valori di correlazione a 16 bit. L'utilizzo

[3]La notazione utilizzata per le parti in assembly è la AT&T: <*opcode*> <*sorgente*> <*destinazione*>.

Algorithm 3 Implementazione SIMD dell'algoritmo full-incremental

```
// carica 64 bit in un registro xmm
function load64(uint8_t* data, xmm_reg)
    movq data, xmm_reg

// carica 128 bit in un registro xmm
function load128(uint8_t* data, xmm_reg)
    movdqa data, xmm_reg

// espande ogni byte del registro in parole da 16 bit
function unpack(xmm_reg)
    punpcklbw xmm_reg, xmm_reg
    psrlw $8, xmm_reg

// calcola corr_delta e lo mette in xmm_reg1
function match(uint16_t* prev, xmm_reg0, xmm_reg1, xmm_reg2, xmm_reg3)
    psubw xmm_reg0, xmm_reg1
    psubw xmm_reg2, xmm_reg3
    pabsw xmm_reg1, xmm_reg1
    pabsw xmm_reg3, xmm_reg3
    psubw xmm_reg3, xmm_reg1
    paddw prev, xmm_reg1

// calcola xmm_reg2 = xmm_reg1 - xmm_reg0 + xmm_reg2
function subadd(xmm_reg0, xmm_reg1, xmm_reg2)
    psubw xmm_reg0, xmm_reg1
    paddw xmm_reg1, xmm_reg2

// salva 128 bit di un registro in memoria
function save128(xmm_reg, uint16_t* data)
    movdqa xmm_reg, data
```

Algorithm 4 Implementazione SIMD dell'algoritmo full-incremental

```
uint8_t pR1[8], pR2[8]
for i = 0 to i < 8 do
    pR1[i] = R1
    pR2[i] = R2
end for
load64(pR1, xmm0)
load64(pR2, xmm2)
unpack(xmm0)
unpack(xmm2)
for d = d_min to d <= d_max inc 8 do
    load64(pL1[d], xmm1)
    load64(pL2[d], xmm3)
    load128(corr_value, xmm4)
    load128(corr_sum, xmm7)
    unpack(xmm1)
    unpack(xmm3)
    match(prev_pixel_corr + d, xmm0, xmm1, xmm2, xmm3)
    save128(xmm1, corr_value)
    subadd(xmm4, xmm7, xmm1)
    save128(xmm1, corr_sum)
end for
```

di una metrica SAD permette di avere valori di correlazione più bassi rispetto a SSD, *sum of square difference*, e questo permette di utilizzare dati di dimensioni minori, 16 bit, e quindi aumentare il parallelismo dell'algoritmo[4].

Il caricamento e il salvataggio dei dati contenuti nei *buffer* ausiliari viene fatto a blocchi di 128 bit, per velocizzare il trasferimento si suppongo tali vettori allineati in memoria a *boundary* di 16 byte[5].

1.4.2 Algoritmo semi-incrementale

L'algoritmo semi-incrementale, privilegiando l'indipendenza tra i pixel della DSI, risulta maggiormente vettorizzabile rispetto al precedente algoritmo; di conseguenza l'aumento di prestazione della versione SIMD rispetto a quella C++ è decisamente maggiore. L'algoritmo 5 descrive la porzione di codice dell'algoritmo che è stata ottimizzata; purtroppo non è possibile scrivere un'unica implementazione SIMD dell'algoritmo, ma è necessario farlo per ogni larghezza della finestra supportata.

Algorithm 5 Implementazione non ottimizzata dell'algoritmo semi-incrementale

```
for d =d_min to d <=d_max do
    corr_delta = 0
    for w = 0 to w <=win do
        corr_delta += |pL[d + w] - pR[w]|
    end for
    corr_sum[d] += corr_delta - corr_value[d]
    corr_value[d] = corr_delta
end for
```

Gli algoritmi 6 e 7 descrivono la porzione di codice SIMD ottimizzata per finestre di larghezza 8 pixel e processori che supportano le istruzioni SSE 4.1. La parte di

[4] I moderni microprocessori permettono di eseguire istruzioni su registri di 128 bit, quindi è possibile calcolare 128 bit / 16 bit = 8 disparità contemporaneamente.

[5] Tale allineamento può essere fatto in fase di allocazione.

Algorithm 6 Implementazione SIMD dell'algoritmo semi-incrementale

```
// carica 64 bit in un registro xmm
function load64(uint8_t* data, xmm_reg)
    movq data, xmm_reg

// carica 128 bit in un registro xmm
function load128(uint8_t* data, xmm_reg)
    movdqa data, xmm_reg

// carica 128 bit non allineati in un registro xmm
function load128u(uint8_t* data, xmm_reg)
    movdqa data, xmm_reg

// copia il registro xmm_reg0 in xmm_reg1
function copy128(xmm_reg0, xmm_reg1)
    movdqa xmm_reg0, xmm_reg1

// calcola corr_delta e lo mette in xmm_reg1
function match(xmm_reg0, xmm_reg1, xmm_reg2)
    mpsadbw $0, xmm_reg0, xmm_reg1
    mpsadbw $5, xmm_reg0, xmm_reg2
    paddw xmm_reg2, xmm_reg1

// calcola xmm_reg2 = xmm_reg1 - xmm_reg0 + xmm_reg2
function subadd(xmm_reg0, xmm_reg1, xmm_reg2)
    psubw xmm_reg0, xmm_reg1
    paddw xmm_reg1, xmm_reg2

// salva 128 bit di un registro in memoria
function save128(xmm_reg, uint16_t* data)
    movdqa xmm_reg, data
```

Algorithm 7 Implementazione SIMD dell'algoritmo semi-incrementale

```
load64(pR, xmm0)
for d =d_min to d <=d_max inc 8 do
  load128u(pL[d], xmm1)
  load128(corr_value, xmm2)
  load128(corr_sum, xmm3)
  match(xmm0, xmm1, xmm2)
  save128(xmm1, corr_value)
  subadd(xmm3, xmm1, xmm1)
  save128(xmm1, corr_sum)
end for
```

codice dipendente dalle dimensioni della finestra di correlazione è rappresentata dal ciclo più interno che calcola il termine *core_delta*. Essa è anche la porzione di codice maggiormente vettorizzabile; infatti, può essere ottimizzata sfruttando l'istruzione SIMD *mpsadbw* che, in un'unica instruzione, esegue 8 SAD di quattro termini di 1 byte.

1.5 Benchmarks

Gli algoritmi sviluppati sono stati testati su tre sistemi, dotati di CPU di diversa generazione:

- Intel® Core™ I7-920 2,66 GHz 64 bit
- Intel® Core™ 2 Duo Mobile-T7200 2,00 GHz 64 bit
- Intel® Pentium4™ 3,00 GHz 32 bit

In questo paragrafo, vengono confrontate le prestazioni dei diversi algoritmi, nelle loro implementazioni basate su instruzioni SIMD e in C++; per sfruttare al massi-

mo il parallelismo dei moderni processori multicore, l'elaborazione è stata suddivisa in più *thread* indipendenti, operanti su fette disgiunte dell'immagine sorgente.
Tutti i test sono stati svolti su una coppia di immagini stereo di dimensione 512×384 con una *baseline* di 1.12 m; in tabella 1.1 è mostrato il numero dei confronti tra finestre al variare dell'intervallo di ricerca, della dimensione della finestra e della densità della DSI.

	Confronti tra finestre		
Finestre	Range costante	Range con terreno	DSI sparsa
4x4	23.166.207	14.434.843	8.153.744
8x8	22.812.867	14.166.947	9.860.957
12x12	22.461.943	13.901.467	10.534.092
16x16	22.113.435	13.638.403	10.914.608

Tabella 1.1: Numero di confronti tra finestre di correlazione nei test effettuati

Il numero dei confronti, come si può osservare, diminuisce utilizzando il terreno nell'intervallo di ricerca e generando una DSI sparsa: questo aiuta a ridurre notevolmente il carico computazionale dell'algoritmo.

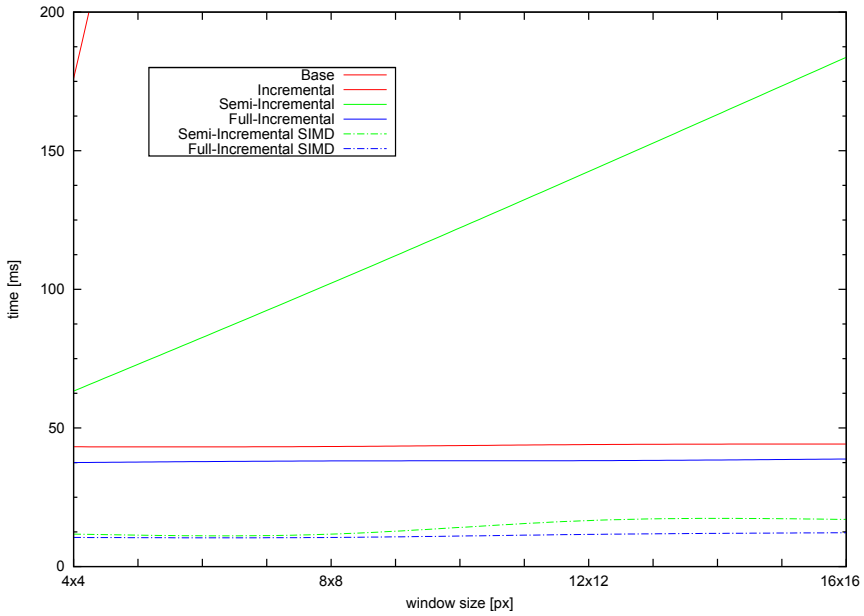
1.5.1 Intervallo di ricerca costante

Il primo test, svolto sulla macchina dotata di Core™I7, consiste nella generazione dell'immagine di disparità densa, raffigurata in figura 1.2.*d*, caratterizzata da un intervallo di ricerca costante $d \in [0,150] \forall y \in [0,384[$.

I risultati sono quelli attesi: tra le implementazioni C++, l'approccio completamente incrementale supera tutti gli altri, avendo una complessità computazionale minore, mentre l'algoritmo semi-incrementale risulta essere linearmente dipendente dalla larghezza della finestra di correlazione. Dall'altra parte, le implementazioni SIMD mostrano che l'algoritmo semi-incrementale è quello più adatto alla vettorizzazione in quanto più parallelizzabile; l'approccio completamente incrementale risulta ancora più veloce, ma la differenza è molto piccola.

Finestre	Tempo di elaborazione [ms]					
	Base	Incremental	Semi-Incremental		Full-Incremental	
	C++	C++	C++	SIMD	C++	SIMD
4x4	176	43,2	63,3	11,7	37,5	10,5
8x8	651	43,3	102,3	11,7	38,1	10,5
12x12	1477	44,0	142,5	16,6	38,2	11,6
16x16	2595	44,2	183,7	17,0	38,8	12,2

Tabella 1.2: Tempo di elaborazione: intervallo costante



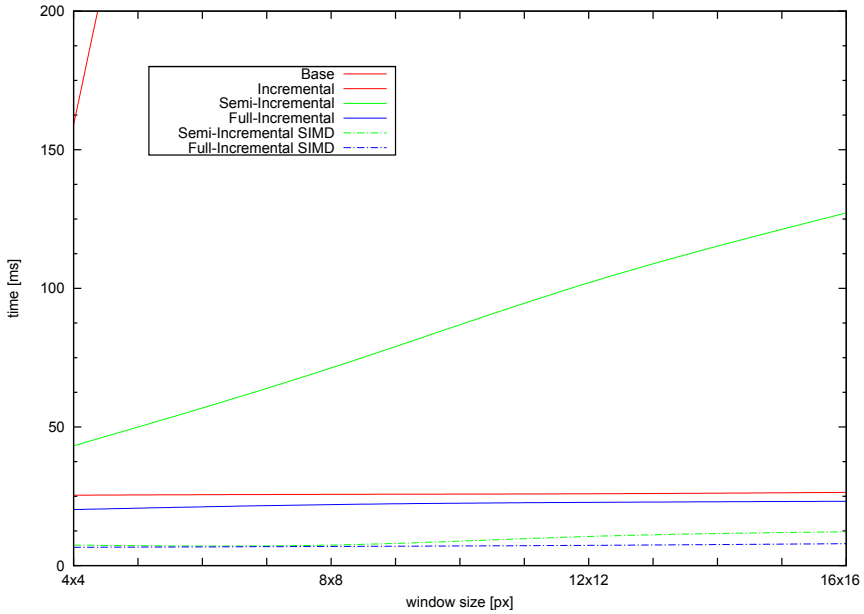
1.5.2 Intervallo di ricerca basato sul terreno

Nel secondo test si è generato la DSI, raffigurata in figura 1.2.*c*, utilizzando una stima della disparità del terreno come bordo sinistro dell'intervallo di ricerca, posto uguale a $d \in [\mathbf{G}[y] - 3, 150] \, \forall \, y \in [0, 384[$.

In questo test, si hanno tempi di elaborazione più bassi del precedente: questo non è sorprendente, in quanto dovuto al minor numero di confronti tra finestre di correlazione, come si può osservare in tabella 1.1. La riduzione dell'intervallo di ricerca impatta in modo eguale su tutti gli algoritmi sviluppati.

Finestre	Tempo di elaborazione [ms]					
	Base	Incremental	Semi-Incremental		Full-Incremental	
	C++	C++	C++	SIMD	C++	SIMD
4x4	159	25,4	43,2	7,4	20,2	6,6
8x8	601	25,7	71,3	7,4	22,0	6,9
12x12	1125	25,9	102,0	10.5	22,8	7,3
16x16	1927	26,4	127,2	12,2	23,2	7,9

Tabella 1.3: Tempo di elaborazione: intervallo di ricerca basato sul terreno



In tabella 1.4, sono mostrati i tempi di elaborazione delle implementazione SIMD dei vari algoritmi sui tre sistemi di elaborazione utilizzati. Essi hanno caratteristiche diverse in termini di frequenza di clock, numero di *core*, dimensione e numero di registri e istruzioni SIMD supportate. Da osservare come nel sistema equipaggiato con un Core2™ il tempo di elaborazione per l'algoritmo semi-incrementale aumenta drasticamente per finestre con larghezza maggiore di 8; non supportando, a differenza del Core™ I7, le instruzioni SSE4, non è infatti possibile effettuare 8 SAD contemporaneamente in quanto l'hardware non fornisce sufficiente parallelismo.

Finestre	Tempo di elaborazione [ms]					
	Core i7 920 2.67 GHz		Core2 T7200 2 GHz		Pentium 4 3 GHz	
	Semi-Inc.	Full-Inc.	Semi-Inc.	Full-Inc.	Semi-Inc.	Full-Inc.
4x4	7,4	6,6	26,8	21,6	70,1	65,6
8x8	7,4	6,9	27,0	23,4	71,0	67,3
12x12	10,5	7,3	41,5	25,9	99,6	70,3
16x16	12,2	7,9	51,3	28,5	107,0	70,6

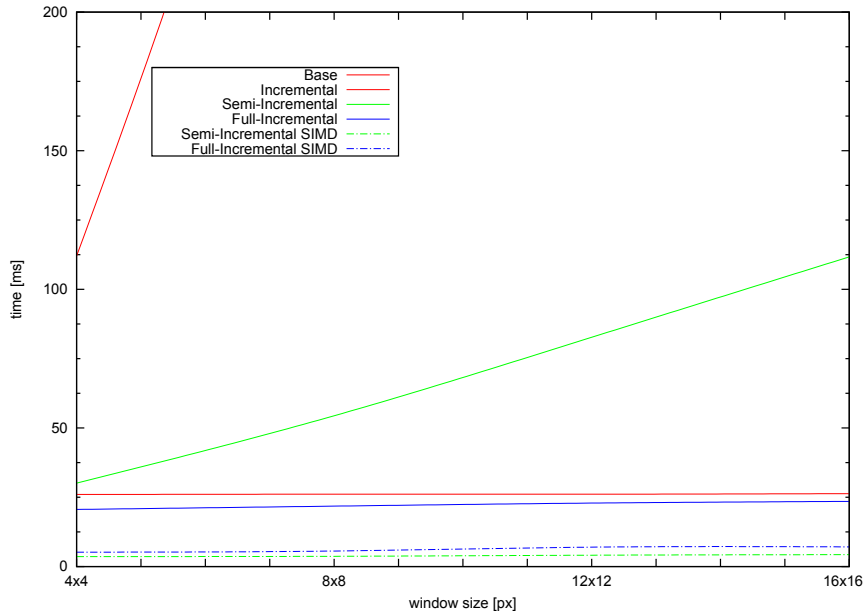
Tabella 1.4: Tempo di elaborazione: le implementazioni SIMD

1.5.3 Immagine di disparità sparsa

Per diminuire ulteriormente il tempo di elaborazione è possibile generare un'immagine di disparità sparsa come quella rappresentata in figura 1.3. In questo test, oltre all'uso del terreno per generare l'intervallo di ricerca, è stato ridotto il numero di punti in cui calcolare la disparità, mantenendo solo quelli la cui finestra di correlazione ha una varianza $\sigma^2 > 30$. È interessante notare che, in questo caso, l'implementazione SIMD dell'algoritmo semi-incrementale risulta la più veloce, grazie alla capacità dell'algoritmo di non computare i punti dell'immmagine di disparità che verranno successivamente scartati. Anche l'algoritmo completamente incrementale subisce una velocizzazione dovuta alla riduzione del numero di minimizzazioni da eseguire.

Finestre	Tempo di elaborazione [ms]					
	Base	Incremental	Semi-Incremental		Full-Incremental	
	C++	C++	C++	SIMD	C++	SIMD
4x4	112	26,0	30,1	3,6	20,6	5,2
8x8	414	26,1	54,4	3,7	21,8	5,6
12x12	946	26,1	82,7	4,1	22,9	7,0
16x16	1668	26,3	111,7	4,3	23,5	7,1

Tabella 1.5: Tempo di elaborazione: immagine di disparità sparsa



1.5.4 Multi-threading

I moderni processori sono dotati di diversi *core* in grado svolgere elaborazione in parallelo; purtroppo i compilatori non sono ancora in grado di parallelizzare automaticamente codice complesso come quello degli algoritmi presentati. Quindi, per

sfruttare al massimo tutta la potenza computazionale disponibile, occorre creare diversi *thread* ognuno dei quali lavora su una fetta orizzontale disgiunta dell'immagine sorgente. Queste fette vengono create in modo da tenere eguale la sommatoria degli intervalli di ricerca dei punti appartenenti alle varie regioni (tale valore è una stima del costo computazionale della singola fetta). Se si usa il terreno per generare l'intervallo di ricerca, mantenendo fisso il suo bordo destro, si avranno, quindi, delle fette di dimensione sempre maggiore dall'alto verso il basso dell'immagine. Per ridurre il *context switch* tra i vari *thread* è opportuno assegnare ognuno di essi ad un determinato *core* attraverso apposite chiamate di sistema fornite dal sistema operativo.

Nella tabella seguente sono mostrati i tempi di elaborazione dell'algoritmo completamente incrementale sulla macchina equipaggiata col processore Core™ I7 al variare del numero di *thread* creati.

Numero Thread	Tempo di elaborazione [ms]
1	16,1
2	9,8
4	6,9
8	9,8

Tabella 1.6: Tempo di elaborazione: multi-threading

Il processore utilizzato è dotato di 4 *core* fisici e 8 virtuali. Come si può osservare dalla tabella 1.6 il numero ottimo di *thread* corrisponde al numero di *core* fisici.

Capitolo 2

Ricostruzione 3D su immagini IPM

Nel capitolo precedente sono stati presentati gli algoritmi per costruire in modo efficiente l'immagine di disparità, ora verrà introdotto il concetto di *Inverse Perspective Mapping*, e sarà analizzata l'applicazione della disparità su immagini IPM.

Il termine *Inverse Perspective Mapping* (IPM) compare per la prima volta nel 1990 introdotto da Mallot [12]. È una trasformazione geometrica che permette di rimuovere l'effetto prospettico delle immagini acquisite. Essa ricampiona l'immagine sorgente mappando ogni pixel in una posizione differente, creando una nuova immagine (IPM) che rappresenta una vista dall'alto della scena osservata. Questa tecnica consente di omogeneizzare il contenuto informativo di ogni singolo pixel dell'immagine: sottocampionando i pixel dell'immagine sorgente vicini alle telecamera e sovracampionando i punti lontani. Come fa notare lo stesso Mallott [12], l'IPM non corrisponde a una vera inversione della proiezione prospettica, che è matematicamente impossibile, ma denota invece un inversione, sotto il vincolo aggiuntivo che i punti riproiettati debbano giacere su un piano orizzontale, in genere corrispondente con quello del terreno. È da notare come tutti i punti che non si trovano sul suddetto piano verranno conseguentemente proiettati in una coordinata mondo errata, acquisendo di fatto erroneamente la coordinata del punto del piano che sottendono.

2.1 Inverse Perspective Mapping

Per introdurre la trasformata di Inverse Perspective Mapping è opportuno partire dalla funzione in forma esplicita della proiezione della *pin-hole* camera:

$$F_{pm}(X,Y,Z) = \begin{pmatrix} k_u \dfrac{r_1(X-X_0)+r_2(Y-Y_0)+r_3(Z-Z_0)}{r_7(X-X_0)+r_8(Y-Y_0)+r_9(Z-Z_0)} + u_0 \\ k_v \dfrac{r_4(X-X_0)+r_5(Y-Y_0)+r_6(Z-Z_0)}{r_7(X-X_0)+r_8(Y-Y_0)+r_9(Z-Z_0)} + v_0 \end{pmatrix} \tag{2.1}$$

dove $(X,Y,Z)^T$ è un generico punto in coordinate mondo, (u_0,v_0) è il *principal point* e (k_u,k_v) la distanza focale in pixel. Tale funzione proietta punti da uno spazio di $\mathbb{R}^3$ in uno spazio $\mathbb{R}^2$ e per questo motivo non è direttamente invertibile. Se tuttavia viene fissata una delle 3 componenti (X,Y,Z) la funzione diventa di $\mathbb{R}^2$ e risulta invertibile. Si definisce l'immagine IPM come l'immagine rappresentante per ogni suo punto (X,Y) la rispettiva coordinata mondo e a cui è associata a tutta l'immagine una coordinata Z costante.

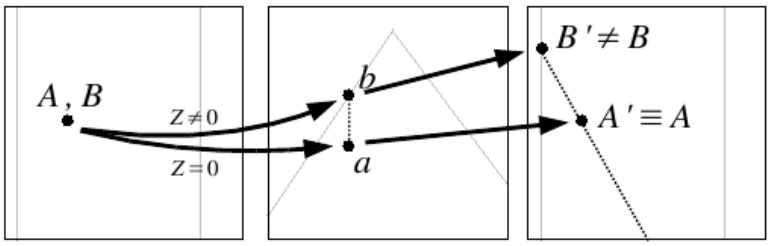


Figura 2.1: La generazione dell'immagine IPM.

L'immagine IPM è il risultato di due trasformazioni prospettiche: una diretta e l'altra inversa, come rappresentato in figura 2.1. Siano $\mathbf{A} = (X,Y,0)^T$ e $\mathbf{B} = (X,Y,Z)^T$, con $Z \neq 0$, due punti del mondo. Essi vengono proiettati su una immagine bidimensionale attraverso la funzione prospettica 2.1 nei punti in coordinate immagine **a** e **b**. I due punti immagine non sono coincidenti in quando derivano da punti mondo differenti. I punti, dopo la proiezione, perdono l'informazione su una coordinata spaziale,

l'altezza dal suolo, che non può essere usata per l'ulteriore trasformata nel dominio IPM. Tale funzione ipotizza che tutti i punti dell'immagine si trovano all'altezza del terreno $Z = 0$: quindi il punto **A** viene riproiettato nella posizione corretta $\mathbf{A}'$, il punto **B**, invece, nella posizione $\mathbf{B}' \neq \mathbf{B}$.

Ora si analizzerà la relazione che intercorre tra due punti riproiettati $\mathbf{A}'$ e $\mathbf{B}'$ e l'altezza dal suolo Z. Nei passaggi seguenti i punti del mondo (X,Y,Z) e quelli dell'IPM (X',Y') saranno espressi nella stessa unità di misura, cioè in metri. Nella realtà le coordinate dell'IPM, trattandosi di un immagine, dovranno poi essere convertite da metri a pixel tramite un opportuno fattore di scala e di *offset*.

Senza perdita di generalità è possibile sottintendere i parametri intrinseci k_u, k_v, u_0 e v_0, perché si elidono nel passaggio tra le due trasformate opposte, e imporre $(X_0,Y_0,Z_0)^T = (0,0,h)^T$, dove h è l'altezza della camera dal suolo. Sotto queste semplificazioni il punto (X,Y,Z) viene proiettato nel punto (u,v) secondo l'equazione:

$$\begin{aligned} u &= \frac{r_1 X + r_2 Y + r_3 (Z-h)}{r_7 X + r_8 Y + r_9 (Z-h)} \\ v &= \frac{r_4 X + r_5 Y + r_6 (Z-h)}{r_7 X + r_8 Y + r_9 (Z-h)} \end{aligned} \tag{2.2}$$

Invertendo questa relazione e imponendo $Z = 0$, si ottiene l'equazione dell'*Inverse Perspective Mapping*:

$$\begin{aligned} X' &= -\frac{r_1 u + r_4 v + r_7}{r_3 u + r_6 v + r_9} h \\ Y' &= -\frac{r_2 u + r_5 v + r_8}{r_3 u + r_6 v + r_9} h \end{aligned} \tag{2.3}$$

Essa rappresenta la relazione tra punti immagine e punti IPM. Per ricavare la relazione tra quest'ultimi e i punti del mondo è necessario unire le equazioni 2.2 e 2.3:

$$X' = -h \frac{X(r_1 r_1 + r_4 r_4 + r_7 r_7) + Y(r_1 r_2 + r_4 r_5 + r_7 r_8) + (Z-h)(r_1 r_3 + r_4 r_6 + r_7 r_9)}{X(r_1 r_3 + r_4 r_6 + r_7 r_9) + Y(r_2 r_3 + r_5 r_6 + r_8 r_9) + (Z-h)(r_3 r_3 + r_6 r_6 + r_9 r_9)}$$

$$Y' = -h \frac{X(r_1 r_2 + r_4 r_5 + r_7 r_8) + Y(r_2 r_2 + r_5 r_5 + r_8 r_8) + (Z-h)(r_2 r_3 + r_5 r_6 + r_8 r_9)}{X(r_1 r_3 + r_4 r_6 + r_7 r_9) + Y(r_2 r_3 + r_5 r_6 + r_8 r_9) + (Z-h)(r_3 r_3 + r_6 r_6 + r_9 r_9)}$$

Osservando che i termini r_i sono i coefficienti di una matrice di rotazione e ricordando che le righe e le colonne di tale matrice sono vettori ortonormali, si ha:

$$\begin{aligned} X' &= -h\frac{X}{Z-h} \\ Y' &= -h\frac{Y}{Z-h} \end{aligned} \tag{2.4}$$

Rimuovendo le semplificazione precedentemente imposte sulla posizione della telecamera, si ottiene l'equazione più generica:

$$\begin{aligned} X' &= -Z_0\frac{X-X_0}{Z-Z_0} + X_0 \\ Y' &= -Z_0\frac{Y-Y_0}{Z-Z_0} + Y_0 \end{aligned} \tag{2.5}$$

dove $(X_0, Y_0, Z_0)^T$ sono le coordinate della camera, $(X, Y, Z)^T$ la posizione del punto in coordinate mondo, e $(X', Y', 0)^T$ le coordinate del punto corrispondente nell'immagine IPM. Questo rappresenta un risultato notevole, infatti l'equazione 2.5 non dipende dai parametri intriseci ed estrinseci della telecamera, ad eccezione della posizione nel mondo di quest'ultima. Si può quindi affermare che, l'IPM è indipendente dalla rotazione con cui la telecamera osserva la scena, dal tipo di ottica usata e dal metodo utilizzato per generarla.

Si nota, inoltre, che lo stesso punto mondo (X, Y, Z) viene rimappato in coordinate IPM diverse (X', Y') al variare della posizione della telecamera nel mondo. Se il punto mondo ha $Z = 0$, cioè appartiene al suolo, le equazioni si semplificano e risulta che $X' = X$ e $Y' = Y$ indipendentemente dalla posizione della telecamera.

Dall'equazione 2.5, è possibile mettere in relazione le coordinate (X', Y') del punto IPM eliminando la dipendenza dalla variabile Z:

$$\frac{X'-X_0}{X-X_0} = \frac{Y'-Y_0}{Y-Y_0} \tag{2.6}$$

Fissando ora X e Y, il che equivale a prendere un ostacolo con base puntiforme che si erge dal suolo, l'equazione 2.6 diventa l'equazione di una retta con coefficiente

angolare generale m':

$$m' = \frac{X - X_0}{Y - Y_0}$$

Si può quindi affermare che tutti i punti mondo che si trovano su una stessa retta verticale, perpendicolare al terreno, con coordinate (X,Y), sono rimappati nell'IPM sulla retta che collega il *pin-hole* alla posizione della base dell'ostacolo. La retta dipende dalle coordinate (X,Y) del punto del mondo che si vuole proiettare e dalla posizione della telecamera (X_0,Y_0,Z_0). La coordinata Z_0 non influenza però la retta su cui gli ostacoli verticali sono proiettati, ma solo la loro posizione su di essa.

Nelle figure 2.2.*a* e e 2.2.*b* si vede il corpo di un pedone: esso viene rimappato nelle immagini IPM 2.2.*c* e 2.2.*d* su due rette con angolazione diversa, come descritto.

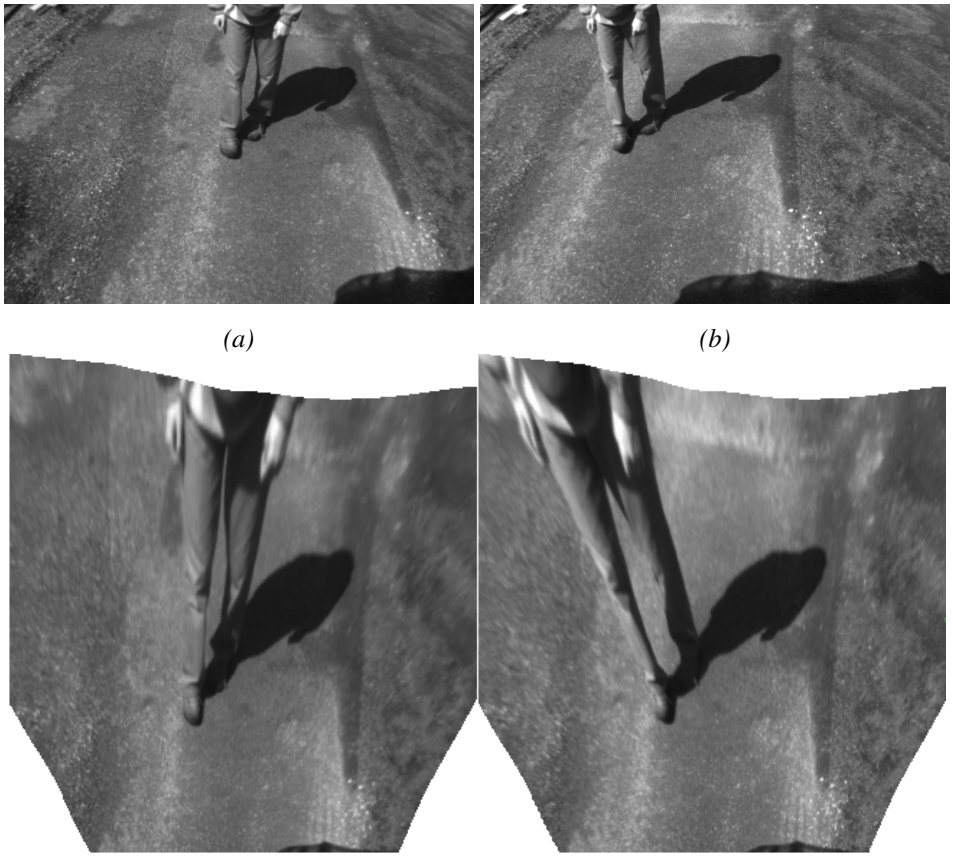
2.2 La disparità e la ricostruzione 3D su immagini IPM

Si prenda in esame un sistema composto da due telecamere stereo poste alla stessa altezza h e distanziate fra di loro di una *baseline* b. La telecamera sinistra si troverà in $(0,b/2,h)$; quella destra in $(0,-b/2,h)$. Un generico punto mondo di coordinate (X,Y,Z), secondo l'equazione 2.5, viene rimappato sulle immagini IPM in

$$\begin{aligned} X'_d &= X'_s = -h\frac{X}{Z-h} \\ Y'_d &= -h\frac{Y+\frac{b}{2}}{Z-h} - \frac{b}{2} \\ Y'_s &= -h\frac{Y-\frac{b}{2}}{Z-h} + \frac{b}{2} \end{aligned} \tag{2.7}$$

Esprimendo la coordinata Y'_d in funzione della coordinata Y'_s si ha:

$$Y'_d = Y'_s - b\left(\frac{h}{Z-h}+1\right) = Y'_s + b\frac{Z}{h-Z}$$



(a) *(b)*

(c) *(d)*

Figura 2.2: Immagini sorgenti: sinistra *(a)* e destra *(b)*; immagini IPM: sinistra *(c)* e destra *(d)*

Si può, quindi, affermare che un generico punto mondo viene proiettato sulla stessa coordinata X' sulle due immagini IPM e con disparità

$$\Delta = Y'_d - Y'_s = b\frac{Z}{h-Z} \tag{2.8}$$

Dalla definizione di disparità su immagini IPM, derivano molte proprietà che possono essere estratte analizzando la equazione 2.8: e osservando la figura 2.3:

- dato che nell'immagine IPM sono rimappati solo punti del mondo con un'altezza dal suolo minore rispetto a quella della telecamera, cioè con $Z < h$, la disparità è sempre positiva;
- la disparità di un punto non dipende dalle coordinate (X,Y) del punto nel mondo, ma solo dalla sua coordinata Z. Questo implica che tutti i punti posti ad una stessa altezza Z nel mondo, presentino la stessa disparità Δ nell'immagine IPM. Il suolo, avendo un'altezza $Z = 0$, ha una disparità $\Delta = 0$. All'aumentare dell'altezza aumenta anche la disparità, fino ad arrivare all'altezza $Z = h$ dove l'equazione presenta una singolarità. Gli oggetti verticali presentano una crescita di disparità lungo la retta definita dall'equazione 2.6;
- la disparità dipende proporzionalmente dalla distanza fra le due telecamere.

Sia s il fattore di scala tra coordinate mondo e immagine. Affinché un punto ad una determinata altezza Z abbia almeno la disparità di un pixel, ovvero $\Delta = s$, la *baseline* deve essere

$$\frac{b}{s} > \frac{h-Z}{Z} \tag{2.9}$$

Il termine sinistro della disequazione è la *baseline* in pixel e può essere usato per determinare la distanza tra le telecamere o la risoluzione dell'IPM in funzione dell'altezza della telecamera e del minimo ostacolo osservabile.

La disparità di un punto su immagini IPM consente di ottenere le coordinate mondo del punto stesso. In particolare la coordinata Z si ottiene direttamente dall'equazione 2.8:

$$Z = h\left(1 - \frac{b}{\Delta + b}\right) = h\frac{\Delta}{\Delta + b} \tag{2.10}$$

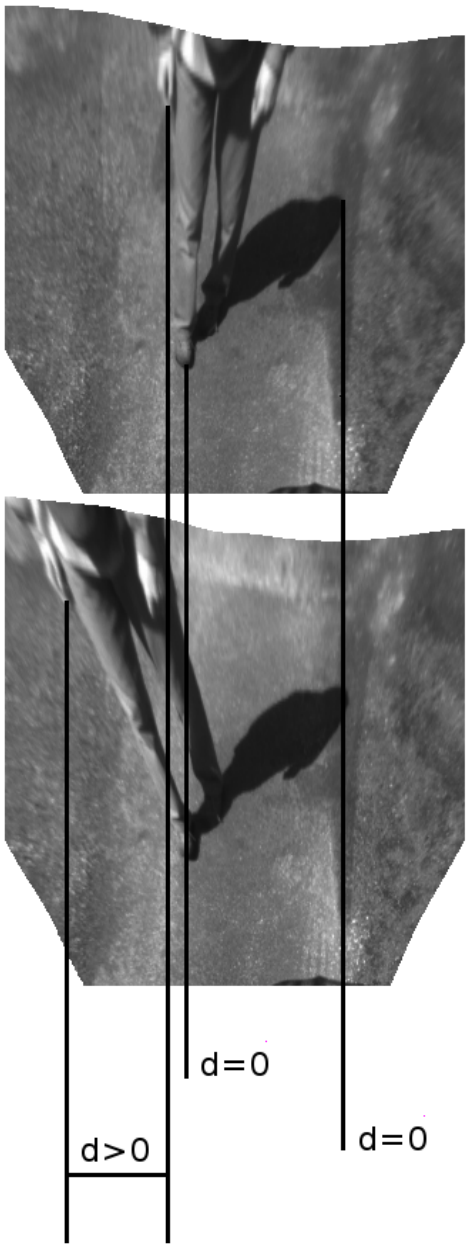


Figura 2.3: La disparità su immagini IPM

dove h e b sono rispettivamente l'altezza dal suolo e la *baseline* delle telecamere e Δ la disparità in metri. Sostituendo all'equazione 2.5 il valore di Z appena trovato si ottiene:

$$\begin{aligned} X'_d &= -h\frac{X}{\left(h\frac{\Delta}{\Delta+b}-h\right)} \\ Y'_d &= -h\frac{Y+\frac{b}{2}}{\left(h\frac{\Delta}{\Delta+b}-h\right)}-\frac{b}{2} \end{aligned}$$

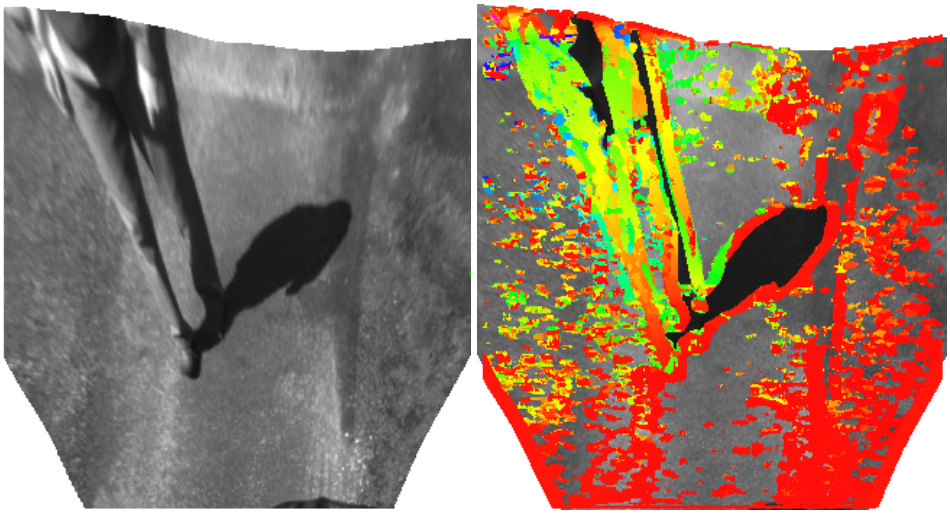
da cui è possibile ricavare i valori delle coordinate mondo (X,Y) in funzione della disparità:

$$\begin{aligned} X &= X'_d\left(\frac{b}{\Delta+b}\right) \\ Y &= \left(Y'_d-\frac{\Delta}{2}\right)\left(\frac{b}{\Delta+b}\right) \end{aligned} \tag{2.11}$$

Nel capitolo precedente si è discusso dello sviluppo di algoritmi efficienti per la generazione di immagini di disparità. Anche nel caso di immagini IPM è possibile utilizzare tali algoritmi per generare la DSI, infatti grazie all'equazione 2.8 i punti omologhi risultano sulla stessa riga. Ovviamente gli intervalli di ricerca e il significato stesso della disparità è diverso rispetto ad immagini tradizionali, ma permette comunque di effettuare una ricostruzione 3D della scena osservata, secondo le equazioni 2.10 e 2.12.

La ricerca dei punti omologhi viene effettuata sull'immagine IPM sinistra sulla stessa riga del punto dell'immagine destra. Il punto omologo non può trovarsi in un generico punto della riga, ma solo in una determinata area. Cerchiamo, ora, di imporre dei limiti all'intervallo di ricerca. Dall'equazione 2.8 discende che la disparità è sempre positiva e il valore minimo che può assumere è uguale a zero se si riferisce ad un punto appartenente al suolo; quindi per ogni punto la disparità minima possibile sarà uguale a zero.

In modo analogo, per ogni punto esiste anche una possibile disparità massima, dipendente dalla sua posizione nell'immagine IPM. Dato un punto $P'_d = (X'_d, Y'_d)$ nell'immagine IPM destra si vuole calcolare il valore massimo di disparità che questo punto



(a) *(b)*

Figura 2.4: Immagine IPM destra *(a)* e immagine di disparità *(b)*

può assumere. Un ostacolo verticale con la base nel punto di coordinate $(X,Y,0)$ viene rimappato nell'immagine IPM destra, secondo l'equazione 2.6, su una retta passante per il punto (X,Y) e per il *pin-hole* della telecamera:

$$\frac{X}{X'_d} = \frac{Y+\frac{b}{2}}{Y'_d+\frac{b}{2}} \tag{2.12}$$

Fissato un punto sull'immagine IPM e visto che la disparità è legata all'altezza dal suolo, essa assume valori più elevati per punti mondo con coordinata X prossima alla telecamera. Sia $X = X_{min}$ la distanza minima dal sistema che può assumere un oggetto da individuare, sostituendo tale valore nell'equazione precedente si ottiene:

$$\frac{X_{min}}{X'_d} = \frac{Y+\frac{b}{2}}{Y'_d+\frac{b}{2}} \tag{2.13}$$

In modo analogo, si può ricavare un'equazione corrispondente per l'immagine IPM sinistra

$$\frac{X_{min}}{X'_s} = \frac{Y-\frac{b}{2}}{Y'_s-\frac{b}{2}} \tag{2.14}$$

Unendo le equazioni 2.13 e 2.14 si ottiene il valore di disparità massima:

$$\Delta_{max} = b\frac{X'_d - X_{min}}{X_{min}} \tag{2.15}$$

La disparità massima che un punto IPM può assumere non dipende dalla sua coordinata Y', ma solo dalla coordinata X'. Quindi ogni riga orizzontale dell'immagine IPM ha la stessa disparità massima. La disparità massima dipende proporzionalmente dalla *baseline*, cioè solo dalla distanza fra le due telecamere. Utilizzando questa equazione è possibile calcolare per ogni punto dell'immagine IPM l'intervallo di ricerca dell'immagine di disparità. Per fare ciò è necessario convertire la Δ_{max} da metri a pixel, tramite un fattore di scala, tenendo conto delle dimensioni dell'immagine IPM.

2.3 Cylindric Inverse Perspective Mapping

Nel paragrafo precedente si è dimostrato che un oggetto verticale viene rimappato sull'immagine IPM lungo la retta che unisce la base dell'ostacolo con il *pin-hole* della telecamera. Visto che le due telecamere non sono coincidenti, un oggetto viene deformato in due modi differenti sulle immagini IPM sinistra e destra, come si può osservare in figura 2.3. Questo fenomeno rende difficoltosa l'individuazione dei punti omologhi e quindi il calcolo della disparità. L'immagine di disparità infatti viene costruita effettuando correlazioni tra finestre appartenenti all'immagine destra e sinistra. Poiché le finestre omologhe hanno un diverso contenuto informativo, non sempre è possibile individuare l'abbinamento corretto e, quindi, l'immagine di disparità risulta rumorosa. Per ovviare a questo problema, si è cercato di individuare una trasformazione di coordinate in grado di rimappare un oggetto verticale su rette verticali e che mantenga inalterata la coordinata X. La trasformazione in oggetto, detta semipolare o in coordinate cilindriche, è così definita:

$$\begin{aligned} \vartheta &= -\arctan\left(\frac{Y' - Y_0}{X'}\right) \\ X'' &= X' \end{aligned} \tag{2.16}$$

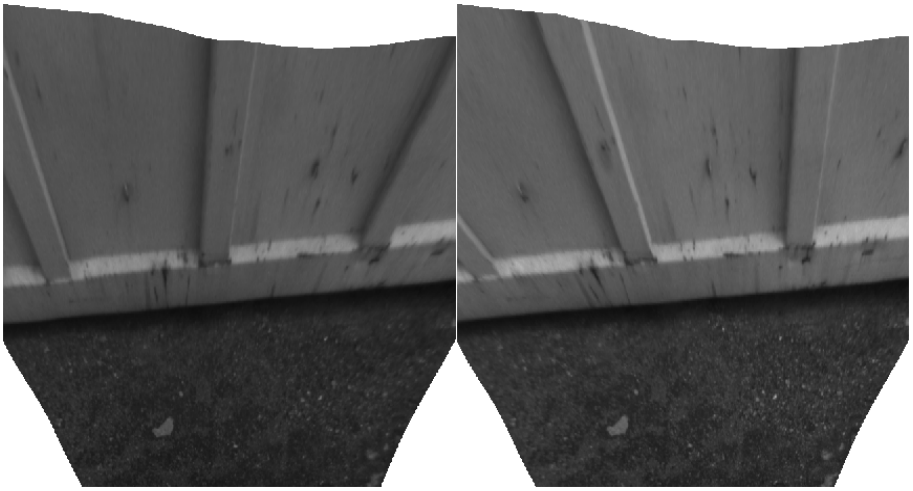
dove ϑ rappresenta l'angolo compreso tra il generico punto mondo e l'asse X, mentre X'' la distanza dalla telecamera lungo l'asse X. Utilizzando questa equazione è possibile generare una nuova immagine detta Cylindric Inverse Perspective Mapping (CIPM), che sulle ascisse ha la variabile ϑ, che può assumere valori compresi nell'intervallo $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ e sulle ordinate la variabile X''. Questa immagine ha la proprietà di mappare in modo simile oggetti nel mondo a X costante tra diverse immagini IPM prese da punti di vista differenti e permettere in questo modo di fare un mapping robusto tra le immagini. Le immagini 2.5, mostrano come un ostacolo verticale viene rimappato allo stesso modo in entrambe le immagini CIPM, figura 2.5.*b*. I punti omologhi in queste immagini si trovano come supposto sulla stessa riga dell'immagine come accade nelle immagini IPM.

Le immagini CIPM possono essere generate a partire dalle immagini IPM, invertendo l'equazione 2.16:

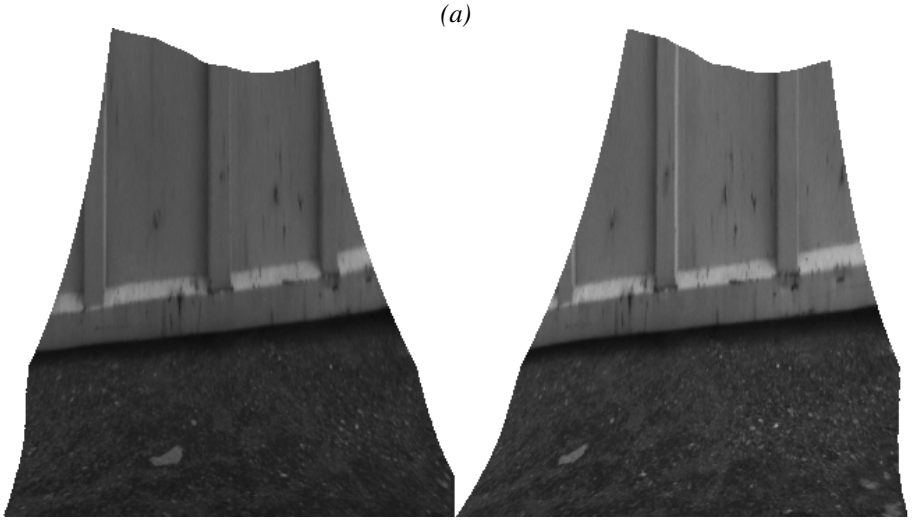
$$\begin{aligned} Y' &= Y_0 - X' \tan(\vartheta) \\ X' &= X'' \end{aligned} \tag{2.17}$$

Per ridurre il tempo di elaborazione ed eventuali errori di quantizzazione è opportuno generare tali immagini in un solo passaggio a partire dalle immagini sorgenti componendo le varie trasformazioni: utilizzando una sola *lookup table* è possibile eliminare la distorsione, la prospettiva e applicare la trasformazione semipolare.

Per alcuni valori iniziali e finali di ϑ, le immagini CIPM possono presentare zone senza nessuna informazione in quanto esterni all'immagine IPM. Questo permette di ridurre l'intervallo di variazione di ϑ a $[\vartheta_{min}, \vartheta_{max}]$. Le immagini CIPM generate dall'immagine IPM destra e dall'immagine IPM sinistra avranno angoli massimi e minimi diversi perché differente è la posizione del *pin-hole* delle due telecamere. I valori di questi angoli possono essere ricavati dalla conoscenza dei punti limite



(a)



(b)

Figura 2.5: Immagini IPM: sinistra e destra *(a)*; immagini CIPM: sinistra e destra *(b)*.

sinistro Y_{max} e destro Y_{min} delle immagini IPM mediante le seguenti equazioni:

$$\vartheta_{d,min} = -\arctan\left(\frac{Y_{max}+\frac{b}{2}}{X_{min}}\right) \qquad \vartheta_{d,max} = -\arctan\left(\frac{Y_{min}+\frac{b}{2}}{X_{min}}\right)$$

$$\vartheta_{s,min} = -\arctan\left(\frac{Y_{max}-\frac{b}{2}}{X_{min}}\right) \qquad \vartheta_{s,max} = -\arctan\left(\frac{Y_{min}-\frac{b}{2}}{X_{min}}\right)$$

Per costruzione l'angolo $\vartheta_{s,min}$ risulta sempre maggiore di $\vartheta_{d,min}$, si definisce quindi $\phi_{min} = \vartheta_{s,min} - \vartheta_{d,min}$: tale grandezza verrà usata in seguito nella trattazione.
Dovendo generare immagini della stessa larghezza W, si definisce un passo di quantizzazione S che definisce come quantizzare l'immagine sorgente:

$$S = \frac{\max\left[(\vartheta_{d,max} - \vartheta_{d,min}), (\vartheta_{s,max} - \vartheta_{s,min})\right]}{W-1}$$

È possibile, dato il valore quantizzato ϑ' in pixel, risalire al valore reale ϑ:

$$\begin{aligned} \vartheta_d &= \vartheta_{d,min} + S\vartheta'_d \\ \vartheta_s &= \vartheta_{s,min} + S\vartheta'_s \end{aligned} \tag{2.18}$$

2.4 La disparità e la ricostruzione 3D su immagini CIPM

Date due immagini CIPM e una coppia di punti omologhi, si definisce disparità cilindrica la differenza fra l'angolo ϑ nell'immagine sinistra e in quella destra.

$$\Delta_c = \vartheta_s - \vartheta_d \tag{2.19}$$

Sostituendo nella precedente l'equazione 2.18, si ottiene

$$\begin{aligned} \Delta_c &= (\vartheta_{s,min} + S\vartheta'_s) - (\vartheta_{d,min} + S\vartheta'_d) \\ \Delta_c &= S(\vartheta'_s - \vartheta'_d) + \phi_{min} \end{aligned} \tag{2.20}$$

Quindi la disparità cilindrica a meno di un fattore di scale S e di un *offset* risulta uguale alla differenza fra due punti omologhi, che può essere calcolata utilizzando

gli algoritmi standard per il calcolo della disparità. La disparità cilindrica, per definizione, corrisponde all'angolo compreso tra le rette passanti per la proiezione del punto sul terreno e il *pin-hole* delle telecamere.

L'equazione 2.16 può essere riscritta sostituendo a Y' e X' i loro valori derivati dall'equazione 2.5 ottenendo:

$$\vartheta = -\arctan\left(\frac{Y - Y_0}{X}\right) \tag{2.21}$$

Questo risultato ci permette di mettere in relazione la posizione di un punto nel mondo (X,Y,Z) con la sua disparità cilindrica:

$$\Delta_c = \vartheta_s - \vartheta_d = \arctan\left(\frac{Y + \frac{b}{2}}{X}\right) - \arctan\left(\frac{Y - \frac{b}{2}}{X}\right) \tag{2.22}$$

Applicando la formula trigonometrica della differenza di tangenti si ottiene

$$\tan\Delta_c = \frac{\tan\vartheta_s - \tan\vartheta_d}{1 + \tan\vartheta_d \tan\vartheta_s} = \frac{bX}{X^2 + Y^2 - (\frac{b}{2})^2} \tag{2.23}$$

Dalle equazioni appena ricavate si può osservare che:

- la disparità cilindrica è sempre positiva, in quanto per costruzione $\vartheta_s > \vartheta_d$;
- la disparità cilindrica dipende solo dalle coordinate (X,Y) del punto, come mostrato dall'equazione 2.22; i punti appartenenti ad oggetti verticali hanno, quindi, disparità cilindrica costante;
- la disparità cilindrica di un punto appartenente al terreno non è più uguale a zero, figura 2.6, come avviene per la disparità sull'immagine IPM;
- la disparità cilindrica dipende dalla posizione Y_0 delle due telecamere ma non dalla loro altezza Z_0.

In modo analogo alla disparità sull'immagine IPM, la conoscenza della disparità dei punti di un'immagine CIPM permette la ricostruzione tridimensionale della scena. Per ricavare le equazioni che permettono, nota Δ_c, di ricavare le coordinate mondo di

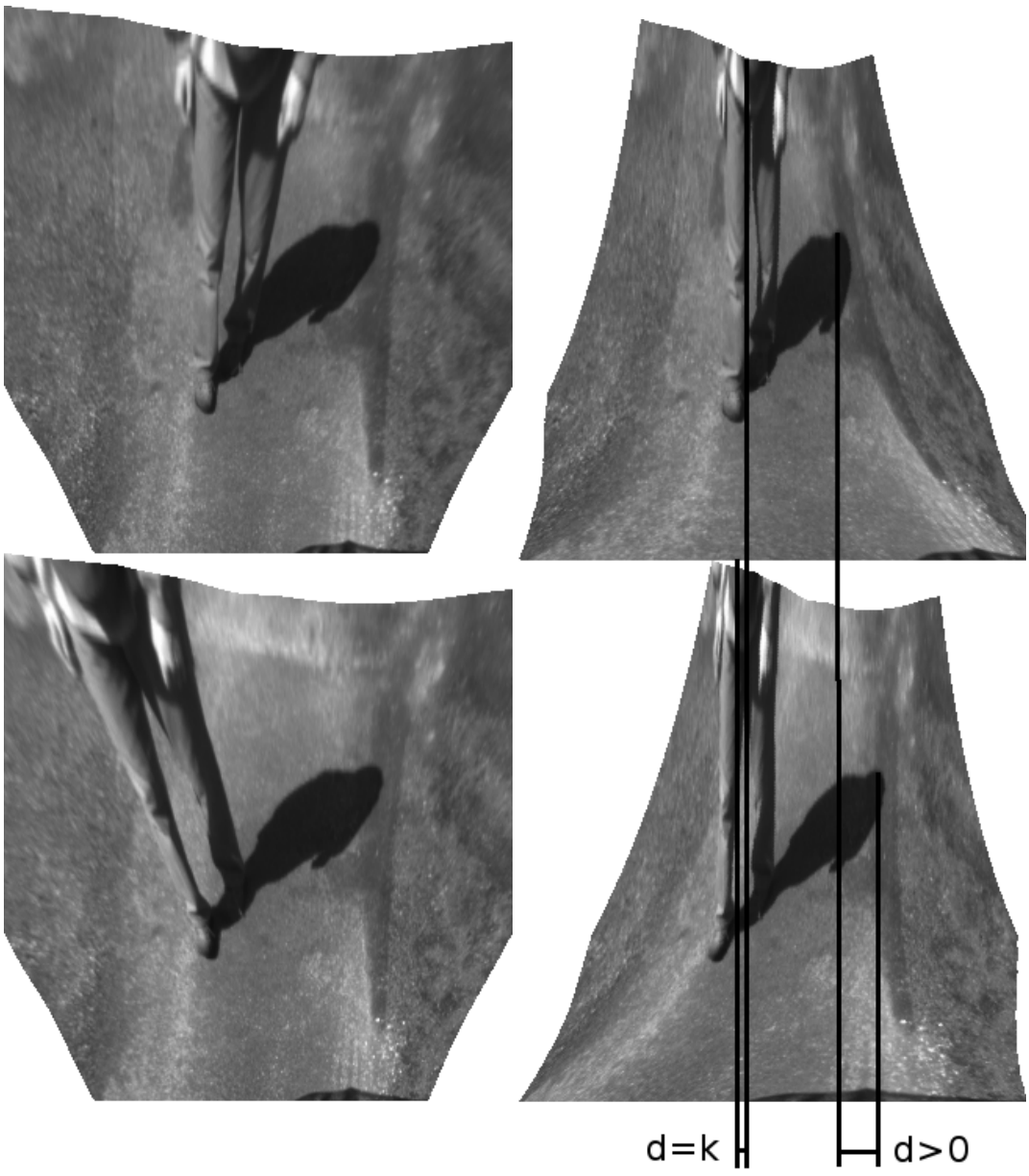


Figura 2.6: La disparità su immagini CIPM.

un punto, è possibile utilizzare le equazioni che mettono in relazione queste ultime e la disparità su immagini IPM. Riscrivendo l'equazione 2.17 per le immagini destra e sinistra, si ottiene:

$$\begin{aligned} Y'_d &= -\tan(\vartheta_d)X'_d - \frac{b}{2} \\ Y'_s &= -\tan(\vartheta_s)X'_s + \frac{b}{2} \end{aligned} \tag{2.24}$$

Usando l'equazione 2.8, è, quindi, possibile esprimere la disparità calcolata sull'immagine IPM in funzione della disparità cilindrica:

$$\Delta = Y'_d - Y'_s = (\tan(\Delta_c + \vartheta_d) - \tan(\vartheta_d))X' - b \tag{2.25}$$

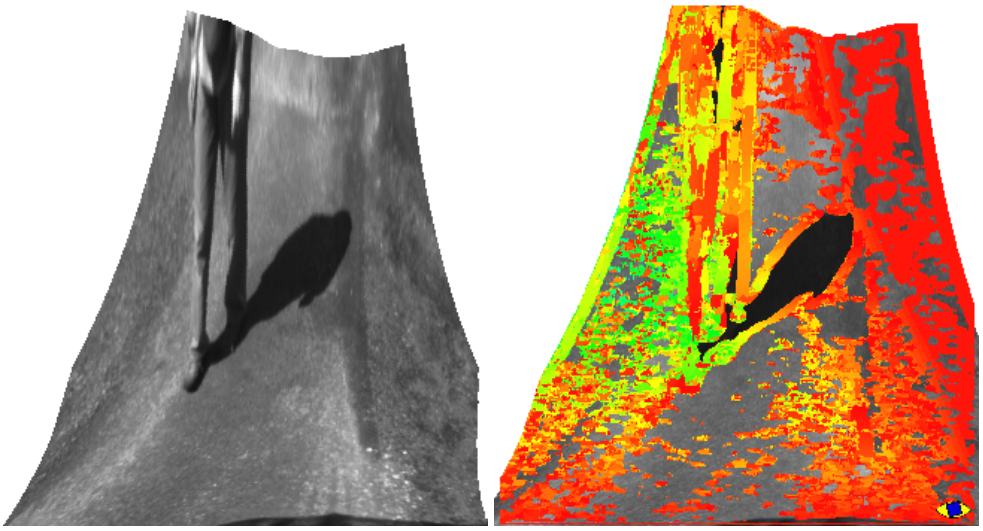
Per risalire alle coordinate mondo si possono applicare le equazioni 2.10 e 2.12, sostituendo a Δ il valore della 2.25:

$$\begin{aligned} Z = h\frac{\Delta}{\Delta + b} &= h\left(1 - \frac{b}{(\tan(\Delta_c + \vartheta_d) - \tan(\vartheta_d))X''}\right) \\ X = \frac{X'b}{\Delta + b} &= \frac{b}{\tan(\Delta_c + \vartheta_d) - \tan(\vartheta_d)} \end{aligned} \tag{2.26}$$

L'equazione della Y può essere ricavata dall'equazione 2.21 sostituendo a X il valore appena trovato:

$$Y = -\tan(\vartheta_d)\frac{b}{\tan(\Delta_c + \vartheta_d) - \tan(\vartheta_d)} - \frac{b}{2} \tag{2.27}$$

L'immagine di disparità, mostrata in figura 2.7, può essere generata utilizzando gli algoritmi discussi nel capitolo precedente. A differenza della disparità sulle immagini IPM, il confronto fra due finestre di correlazione di oggetti verticali risulta più semplice, grazie alla proprietà della trasformata CIPM di rimappare oggetti con (X, Y) costante in modo simile nelle due immagini destra e sinistra; allo stesso tempo, risulta difficoltoso il confronto dei pixel appartenenti al terreno che vengono deformati in modo differente.



(a) *(b)*

Figura 2.7: Immagine CIPM destra *(a)* e immagine di disparità *(b)*

La disparità cilindrica, come si è dimostrato, è sempre positiva; la sua corrispondente quantizzata, invece può assumere valori negativi, a causa del diverso ϑ_{min} nelle due immagini CIPM. Infatti l'immagine CIPM è definita nell'intervallo $[\vartheta_{min}, \vartheta_{max}]$, quindi a causa del diverso angolo iniziale, alla stessa coordinata ϑ' corrisponde sulle due immagini un valore di ϑ diverso.

Per definire l'intervallo di ricerca dell'immagine di disparità è necessario calcolare i valori massimi e minimi che la disparità cilindrica può assumere. Avendo definito la disparità cilindrica di un punto come l'angolo α compreso tra le rette passanti per il *pin-hole* delle telecamere e la sua proiezione sul terreno, si può affermare, per costruzione, che il valore massimo si raggiunge nel punto con una coordinata X minore possibile e $Y = 0$. Visto che la disparità cilindrica è costante al variare di Z, la disparità massima è valida per ogni riga dell'immagine ed è possibile calcolarla, utilizzando

l'equazione 2.23:

$$\Delta_{c,max} = \arctan\left(\frac{bX_{min}}{X_{min}^2 - \left(\frac{b}{2}\right)^2}\right) \tag{2.28}$$

A differenza delle immagini IPM, la disparità cilindrica minima non è uguale a zero per ogni riga dell'immagine, ma dipende dalla riga in esame. Fissata la coordinata X'', essa assume il suo minimo per punti appartenenti al terreno e per valori di ϑ corrispondenti ai limiti dell'IPM. Il valore minimo della disparità cilindrica sulla riga X'' si ha quindi per:

$$\vartheta_1 = -\arctan\left(\frac{Y_{max} + \frac{b}{2}}{X''}\right) \qquad \vartheta_2 = -\arctan\left(\frac{Y_{min} + \frac{b}{2}}{X''}\right)$$

$$\vartheta_{min} = \begin{cases} \vartheta_1 & \text{se } |\vartheta_1| \geq |\vartheta_2| \\ \vartheta_2 & \text{se } |\vartheta_1| < |\vartheta_2| \end{cases}$$

ed utilizzando le equazioni 2.23 e 2.25 si ottiene:

$$\Delta_{c,min} = \arctan\left(\frac{b}{\tan^2(\vartheta_d)X'' + b\tan(\vartheta_d) + X''}\right) \tag{2.29}$$

Utilizzando queste equazioni è possibile calcolare l'intervallo di ricerca dell'immagine di disparità, per ogni punto dell'immagine CIPM ed infine applicare gli algoritmi descritti nel precedente capitolo.

Conclusioni

In questo lavoro di tesi sono stati presentati tre nuovi algoritmi per la generazione di immagini di disparità orientati all'utilizzo nel campo *automotive*. I vincoli sui tempi di esecuzione hanno indirizzato lo sviluppo verso algoritmi che operino confronti locali, mentre la disponibilità delle informazioni sull'andamento del terreno ha consentito di limitare l'area di ricerca dei punti omologhi, aumentando la correttezza dei risultati forniti. L'utilizzo di un approccio incrementale ha permesso di ottenere il risultato notevole di eseguire un solo confronto per pixel nel calcolo del valore di correlazione indipendentemente dalle dimensioni delle finestre, riducendo sensibilmente il tempo di esecuzione.

L'idea di utilizzare un approccio di tipo incrementale ha portato alla realizzazione di un algoritmo ibrido, in cui si sacrifica in parte l'indipendenza dalle dimensioni delle finestre di correlazione allo scopo di non calcolare i valori di correlazione per tutti i pixel, in particolare quelli relativi a zone prive di contenuto informativo, consentendo di ottenere un'immagine di disparità sparsa, riducendo in modo consistente il tempo di elaborazione.

Per ottenere un elaborazione in tempo reale, si è fornito per ogni algoritmo presentato un'implementazione in grado di sfruttare il parallelismo dei processori moderni utilizzando le istruzioni SIMD, inoltre se il sistema è dotato di più unità di elaborazione il processo di generazione dell'immagine di disparità verrà automaticamente suddiviso in modo eguale su di esse.

Gli algoritmi per la generazione dell'immagine di disparità proposti si sono dimostrati molto flessibili, infatti è stato possibile utilizzarli in contesti con intervalli di

ricerca e finestre molto differenti tra loro. In questa tesi, per esempio, è stata analizzata la loro applicazioni su immagini IPM e CIPM ed è stato mostrato come è possibile ottenere una ricostruzione tridimensionale della scena osservata.

Un campo di ricerca interessante è anche quello di realizzare implementazioni che sfruttino, in maniera trasparente al programmatore, risorse di calcolo di tipo differente rispetto alle tradizionali CPU, come le schede grafiche utilizzando librerie per il calcolo di tipo GPGPU[1] come CUDA; inoltre un altro possibile sviluppo futuro è rappresentato dal *porting* degli algoritmi presentati su dispositivi embedded come i *dsp*.

[1] *General-Purpose computing on Graphics Processing Units*

Appendice A

A stereo based pedestrian detection system

A.1 Abstract

This appendix describes an application exploiting the disparity map generation algorithms, presentated in the chapter 1: a tetra-vision system for the detection of pedestrians by means of the simultaneous use of two far infrared and visible cameras stereo pairs. The main idea is to exploit the advantages of both far infrared and visible cameras to develop a system that combines the advantages of using far infrared or daylight technologies.

Different approaches are used to process the two stereo flows in an independent fashion to produce a list of areas of attention that potentially contain pedestrians. Then, four different following approaches are used to refine and filter this list and to validate the presence of a pedestrian. Preliminary results show that the combined use of two vision systems as well as the use of different and independent validation steps enable the system to effectively detect pedestrians in different conditions of illumination and background.

A.2 Introduction

Pedestrian detection is an important field of research for commercial or governmental organizations. The automatic identification of people or obstacles can improve safety for regular vehicles. In particular, the U. S. Army is actively developing obstacle detection for mule operations, path following and intent based anti-tamper surveillance systems for its robotic vehicles safety [13, 14, 15].

Nevertheless, the potential field of use of such technology is larger than U. S. Army necessities: surveillance systems, driver assistance systems, and intelligent systems for autonomous or semi-autonomous driving represent few of the many areas that need to detect the presence of persons in order to take appropriate actions like avoiding it or enabling safety countermeasures.

Unfortunately, the detection of pedestrians is a really challenging task and the difficulty of the problem increases when the movement of the sensors, uncontrolled outdoor environments, and variations in pedestrian appearance and pose have to be taken in account. In the last years, many different approaches to solve this problem have been tested. Some use LADAR or laser scanners to retrieve a 3D map of the terrain and detect pedestrians [16, 17, 18], others use ultrasonic sensors to determine the reflection of pedestrians [19]. Radar sensors can be used in a similar way: pedestrian detection is based on the analysis of the reflections on the targets to discriminate whether they are pedestrians or not [20, 21].

Vision is a natural choice for a pedestrian detection sensor because it is based on how people perceive humans (through visual cues). In particular, for the U. S. Army, vision based detection is important since cameras are non-evasive sensors. A number of systems that approach this task have been developed, based both on monocular [22, 23, 24] or stereo [25, 26, 27, 28, 29, 30] vision.

Recently also infrared technologies (both far and near infrared based) came into in the pedestrian detection arena [31, 32, 33, 34, 35], thanks to the decreasing cost of infrared technology. In many scenarios, far infrared (FIR) cameras are more suited than daylight ones for detecting pedestrians; especially, when the background is colder than the human beings or in night or low-illumination conditions. Moreover, FIR

images generally show less noisy details, easing the initial steps of a detection process. Anyway, far infrared cameras fail the detection of pedestrians in hot or sunny weather, namely when pedestrians are not warmer than the background.

The following presents a pedestrian detection system based on the simultaneous use of a visible and a far infrared stereo systems, in order to exploit the benefits of both approaches.

This appendix is organized as follows. Section A.3 summarizes the whole algorithm; section A.4 introduces the initial phase of the detection of areas of attention. In section A.5 a refinement step based on the use of different symmetry operators is explained; the result is fed to three different validators that are explained in section A.6. Section A.7 shows the results of each validation process while section A.8 summarizes and concludes the chapter.

A.3 The approach

Visible cameras have been widely used for surveillance and detection systems; more recently, thanks to the decreasing costs, also far and near infrared devices became a viable option for such tasks. Generally, the choice between infrared and daylight technologies depends on the specific use and on costs vs benefits analysis.

An object in a far infrared (FIR) domain shows a pattern that depends on its thermal features, namely on the amount of heat it emits. Moreover, this pattern is barely affected by illumination changes. Far infrared images are also generally less noisy, thanks to the relative absence of shadows and textures due to colors. Therefore, FIR images ease the detection of objects warmer (or colder) than the background (like pedestrians, moving vehicles...), since they are sufficiently contrasted with respect to the background. Moreover, FIR cameras are suitable for night or low illumination scenarios, since they are barely affected by illumination. Unfortunately, strong sun heating or high temperature conditions can increase the background thermal impact in the far infrared images, and can also introduce additional textures due to the different thermal behavior of different materials. Moreover, for pedestrian detection, clothes affect the thermal footprint of a human shape, making detection more challenging.

Conversely, images acquired using daylight cameras show a number of small details and shadows. Visible cameras are also highly affected by changes in the illumination conditions.

Often, in the first stages of a detection process, the shape is used as a preliminary detector pattern. In such a case, the presence of a big number of details is cumbersome and makes the detection more difficult. Nevertheless, it is of paramount importance for distinguishing between pedestrians and objects.

In the following, a system based on the simultaneous use of two infrared and visible cameras stereo pairs is described. The main idea is to exploit both the advantages of far infrared and visible cameras trying at the same time to cope with the deficiencies of each system. Different approaches have been developed for pedestrian detection in the two different image domains: warm areas detection, vertical edges detection, and an approach based on the simultaneous computation of disparity space images in the two domains.

These first stages of detection output a list of bounding boxes that enclose potential pedestrians. A symmetry-based approach is furtherly used to refine this rough result. Then, a number of validators are used to evaluate the presence of a human shape inside each bounding box. The validation is performed searching for human shape characteristics: head detection, shape detection, neural network, and an active contour-based approach. Figure A.1 sketches the overall algorithm flow.

A.4 Detection of Areas of Attention

Different approaches are used for computing a list of areas of the image that potentially contain a pedestrian.

A.4.1 Run-time calibration

The knowledge of correct calibration parameters is of paramount importance for machine vision, since it is mandatory to resolve the relationship between image matrix and real world coordinates.

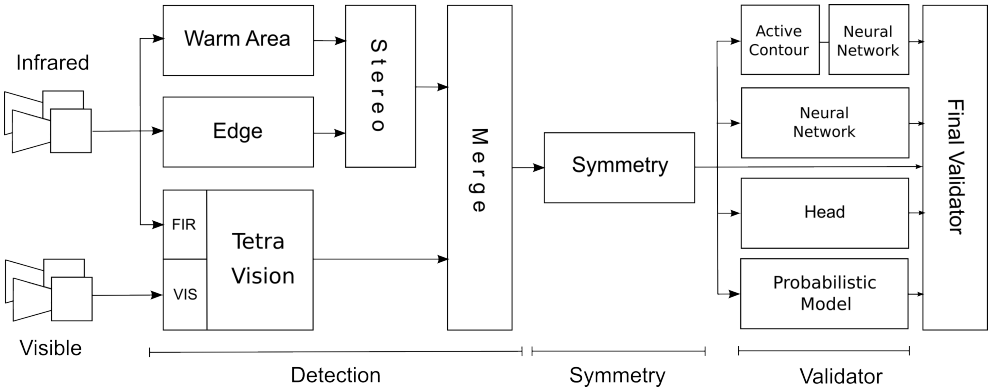


Figura A.1: Overall algorithm flow.

A lot of vision-based systems for intelligent vehicles rely on the assumption of moving on a flat road or –more realistically– on a road with a smoothly varying slope. Anyway, this assumption, can lead to errors in computing scene parameters or even to misdetections, not only when the road is not really flat, but also due to the fact that a camera installed on a moving vehicle continuously varies its yaw, pitch and roll.

A V-disparity approach is used for computing the actual road slope, exploiting the presence of two stereo vision systems [36, 37]. The knowledge of the road slope allows to compute the actual pitch angle that is widely used in the following detection steps.

A Sobel filter is used to enhance images features, especially edges. Then, a correlation is computed for different offset values (*disparities*), for each pair (left and right) of rows of the Sobel images. Thanks to the specific setup of the stereo pair, the same rows in the stereo images are epipolar lines. This process has been tested both for visible and infrared images; the result is a new image, the *V-disparity image*, encoding correlation values (see figure A.2). The brighter the pixel, the higher the match quality. It can be noticed that in visible images, the ground components produce a slanted line that can give information about the road cross section. Conversely, due to the bare presence of ground edges in infrared images, the FIR V-disparity image does not permit to recover road data.

(a) *(b)* *(c)*

Figura A.2: V-disparity computation in visible and infrared domains: *(a)* original images, *(b)* Sobel images, and *(c)* V-disparity image.

Therefore, only the visible domain is used to compute the ground slope; anyway, this information can be used in the following process also for the infrared domain thanks to the knowledge of relative calibration of all four cameras.

A.4.2 FIR-only processing

Two different processings are dedicated to analyze FIR images for detecting a list of areas in the image that potentially contain pedestrians: the detection of warm areas and an edge-based filtering.

Warm areas detection

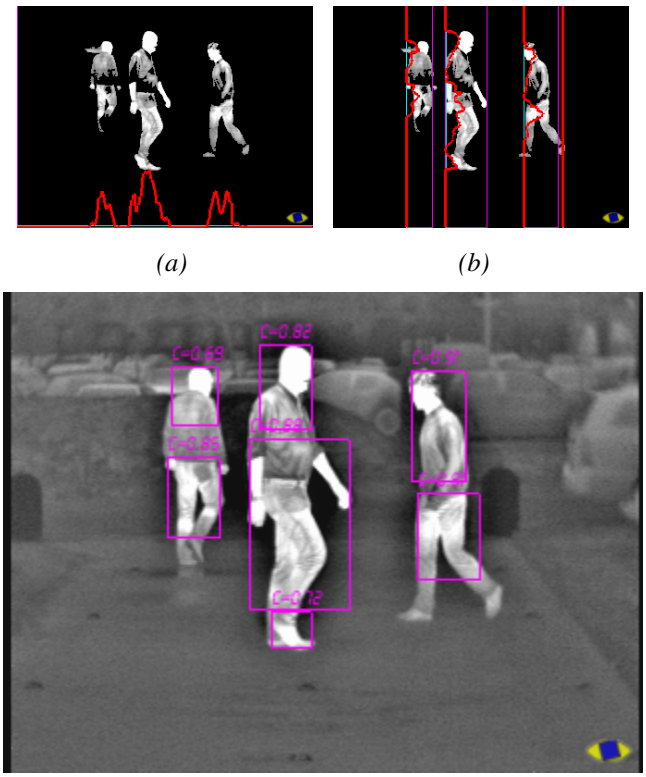
This approach is based on the fact that pedestrians are usually warmer than the background, and therefore they are rendered as bright regions in FIR images. Therefore,

(a) *(b)*

Figura A.3: Preprocessing phase: *(a)* original input image, *(b)* focus of attention.

as a first step of the algorithm, high intensity areas on the input image are searched for. This is obtained using two different threshold values: initially, a high threshold is applied on all pixels in order to get rid of cold or barely warm areas, selecting pixels corresponding to very warm objects only. Then, pixels featuring a grey level higher than a lower threshold are selected if they are contiguous to other already selected pixels in a region-growing fashion. The resulting image contains only warm contiguous areas that present hot spots (figure A.3).

In order to select vertical stripes containing hot regions, a column-wise histogram is computed on the resulting image (see figure A.4.*a*). The histogram is filtered with an adaptive threshold whose value is a fraction of the average value of the whole histogram. Obviously, multiple hot objects may be vertically aligned in the image, so that their contributions add up in the histogram. Nevertheless, warm areas belonging to the same horizontal stripe can be distinguished by computing a new row-wise histogram of the grey-levels for each stripe (figure A.4.*b*). This procedure yields rectangular bounding boxes framing areas where pedestrians may be located. In order to refine these bounding boxes, the column-wise and row-wise histogram procedure is iteratively applied to each rectangular box until its size is no longer reduced. In this phase, small bounding boxes are removed, as they represent nuisance elements. Results are shown in figure A.4.*c*.



(a) *(b)*

(c)

Figura A.4: Warm elements detection: *(a)* a column-wise histogram detects columns with warm object, *(b)* following row-histograms for each column detect initial areas of attention, *(c)* final results are shown on the original image.

Edge detection

Another intrinsic characteristic of FIR images is the relative absence of image textures. This is due to the higher homogeneity of thermal patterns with respect to color features. Therefore, FIR images present less edges than their daylight counterparts; in addition, these edges are much more likely due to the object borders: this reduces the problem of noise in a edge detection based approach.

Usually, a human shape features more vertical edges than the background or than other objects. Therefore, this approach is based on the detection of areas that contain a high amount of vertical edges.

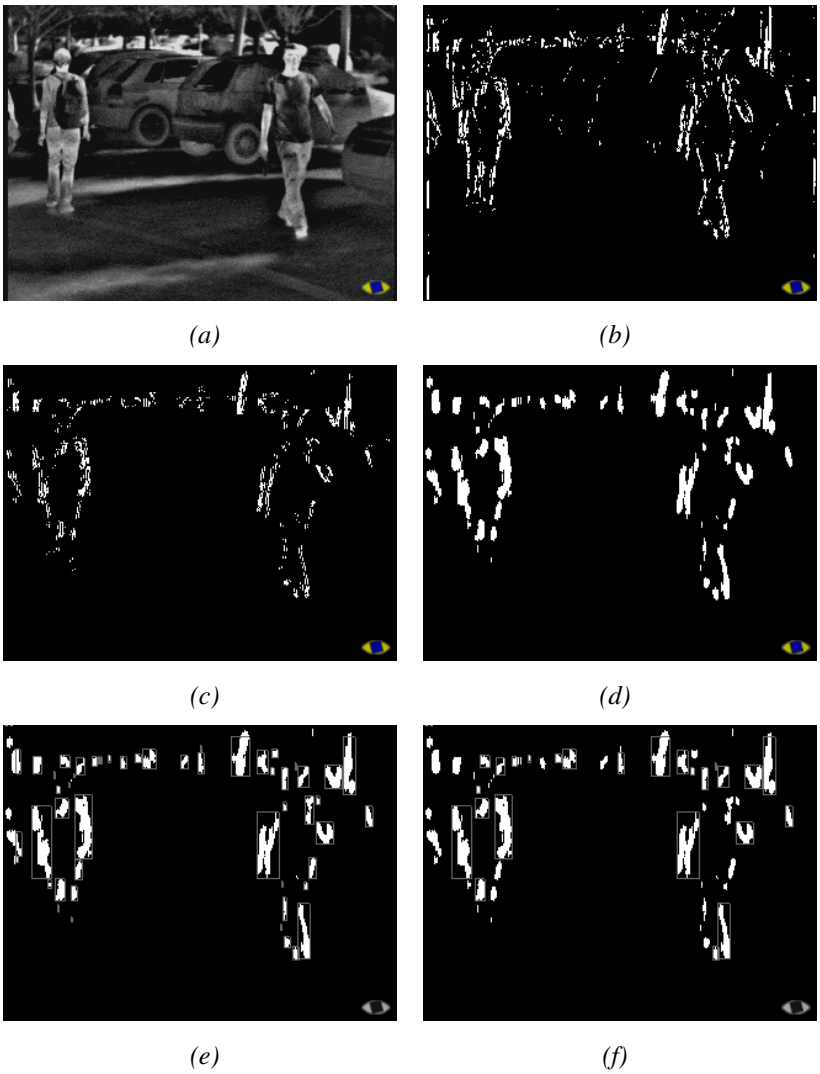
Initially, acquired images are filtered using a Sobel operator and an adaptive threshold to detect nearly-vertical edges (figures A.5.*a* and A.5.*b*). Unfortunately, also some objects other than pedestrians contain a high amount of vertical edges: buildings, cars, poles, etc. But vertical edges of such objects are generally longer and more uniform than the ones that belong to human shapes. Therefore, a filtering phase devoted to the removal of regular vertical edges that are longer than a given threshold is performed. Also isolated pixels are considered as noise and removed as well (figure A.5.*c*). In order to further enhance vertical edges and to group edges of a pedestrian in the same cluster, a morphological expansion is performed using a 3×7 operator. In the final result (figure A.5.*d*), only a few clusters are present.

A labelling approach is used to compute connected clusters of pixels; the resulting image is analyzed, and a list of bounding boxes containing connected clusters of pixels is built (figure A.5.*e*). Generally, in complex scenarios, a large number of small clusters are detected affecting both the effectiveness and the efficiency of the subsequent processing steps. Thus, a filtering phase is used to remove bounding boxes that are too small and contain cold areas only; therefore boxes sufficiently large or containing bright pixels survive this step (figure A.5.*f*).

Stereo Match

Bounding boxes detected during the previous two steps are then matched against the other image in order to compute their size and position in the real world. Thanks to the knowledge of vision system calibration computed in section A.4.1, a search area in the other image for each bounding box can be estimated.
The following Pearson's correlation function is used to evaluate the result of the match:

$$r = \frac{\sum a_{xy} b_{xy} - \frac{\sum a_{xy} \sum b_{xy}}{N}}{\sqrt{\left(\sum {a_{xy}}^2 - \frac{(\sum a_{xy})^2}{N}\right)\left(\sum {b_{xy}}^2 - \frac{(\sum b_{xy})^2}{N}\right)}} \tag{A.1}$$



(a) *(b)*

(c) *(d)*

(e) *(f)*

Figura A.5: Edge detection: *(a)* original image, *(b)* vertical edge detected using the Sobel operator and a threshold, *(c)* edge after the removal of strong and regular edges, *(d)* expanded edges, *(e)* bounding boxes containing connected clusters of pixels, and *(f)* final resulting list of bounding boxes.

Figura A.6: Stereo match: homologous bounding boxes in the left image.

where N is the number of pixels in the considered bounding box, a_{xy} and b_{xy} are grey-level values of corresponding pixels of the two images. The bounding box in the other image featuring the maximum value for the correlation is selected as the best match. The algorithm discards the matches whose correlation values are lower than a given threshold (figure A.6). A triangulation technique is used to estimate the distance between each object and the vision system.

A.4.3 Independent tetra-vision obstacle detection

In the following a multi-domain approach to detect region of interest is explained. Dealing with two stereo systems that work in different domains is a bit tricky, since the two stereo systems feature different points of view, different angular apertures, and –generally– different frame rates and resolutions. In addition, the whole system can not rely on a search for homologous points, since it is very difficult to match features across different spectral domains.

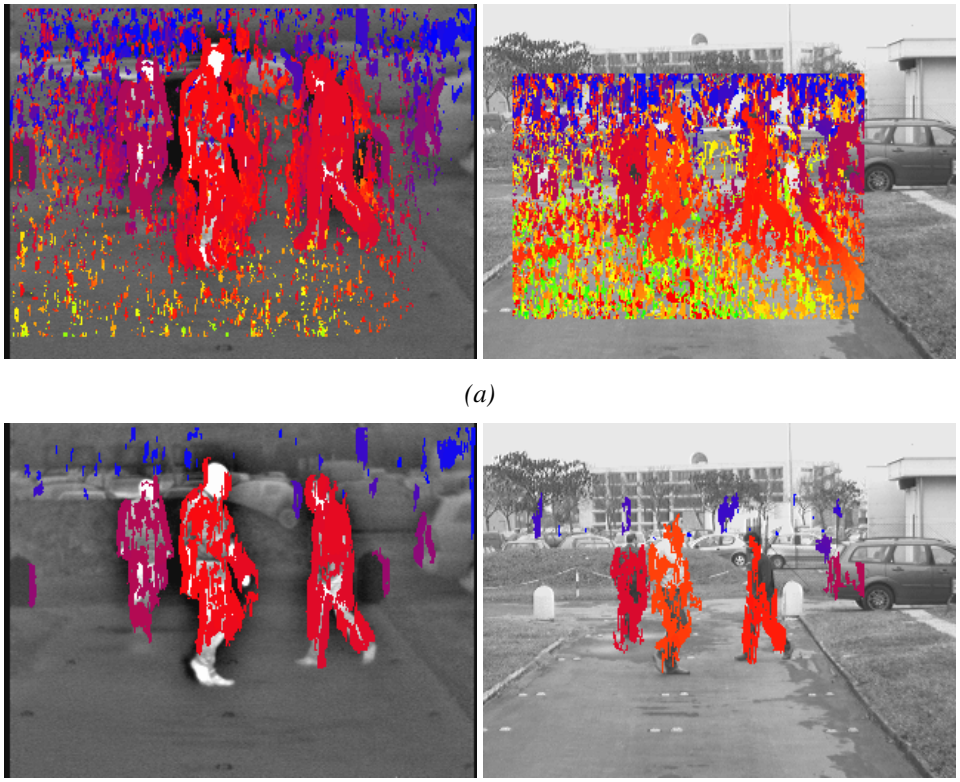
Many image registration strategies have been proposed [38], but unfortunately they require annotation of homologous points or an a-priori knowledge of the environment. Conversely, the proposed system independently processes the two stereo flows and then fuses the results coming from the two different domains.

Initially, a *disparity space image* (DSI) computation is used to detect the presence of obstacles: the right images of each flow are subdivided into 3×8 pixels regions and their corresponding regions are searched for into the left images. The reason for a 3×8 region size is due to the fact that a pedestrian shape is generally characterized by strong vertical features. The search for a homologous region is limited to the same row of the left image, since the optical axis of the two stereo systems are parallel, thus two corresponding rows in the two images are epipolar lines. Moreover, calibration information permits to further bound the search area, reducing both the required computational load and the probability of wrong matches. For each region in the image, the best matching area in the other image is considered, and the disparity between their coordinates is computed. The match is performed using the SAD (sum of absolute differences) correlation:

$$S = \sum_{i=1}^{8} \sum_{j=1}^{3} (|L_{i,j} - R_{i,j}|) \tag{A.2}$$

where (i, j) are the coordinates of each pixel into the 3×8 regions. The result is a new image, the disparity space image, in which each pixel encodes the disparity value. Figures A.7.*a* show an example of DSIs computed both for the visible and infrared domains; the colors encode the disparity value: bright colors correspond to high disparities, while dark colors correspond to small disparity values. It can be noticed that only obstacles feature large clusters of pixels that have similar disparity values; conversely, background features present colors that evolve from bright to dark when moving from the bottom part of the image towards the top.

An aggregation step on the DSI is therefore used to detect the presence of obstacles: large connected regions featuring a similar disparity are marked as obstacles. DSI pixels that have the same value as the ground disparity are removed. Then the DSI images are clustered in order to detect the connected areas having similar disparity. Not all cluster of pixels are considered as obstacles: in fact, size, distance, and height constraints are used to further reduce their number. Figures A.7.*b* show resulting clusters with different colors.



(a)

(b)

Figura A.7: DSI: *(a)* the disparity images for visible and infrared domains –the darker the color, the smaller the disparity– and *(b)* detected obstacles regions labelled using different colors.

A.4.4 Merge and rough filtering step

Each area of attention detected by the segmentation process previously described is marked using a bounding box (see figure A.8.*a*). Unfortunately, since different approaches are used to process the same scene, different bounding boxes often belong to the same obstacle; therefore a merging process is mandatory, following the rule that two bounding boxes are fused to compose a larger bounding box when they are

sufficiently close in real world coordinates. Figure A.8.*b* shows an example of the merging process in the infrared domain; thanks to the use of world coordinates for the merging, bounding boxes belonging to obstacles that partially overlap are not merged.

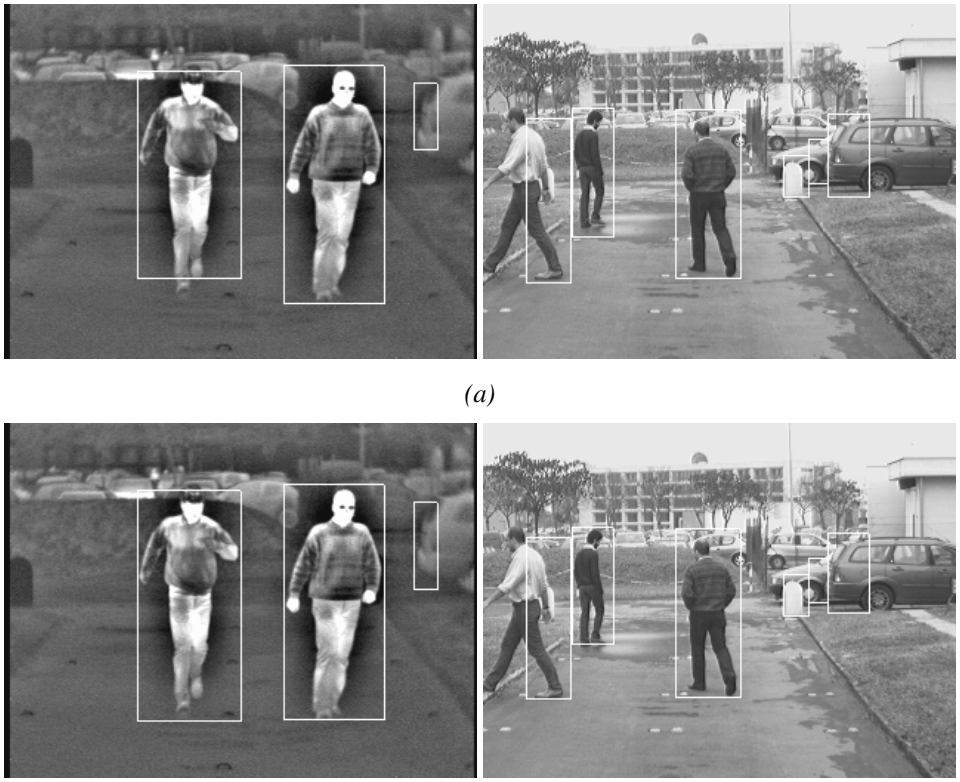
(a) *(b)*

Figura A.8: Bounding boxes generation and merging: *(a)* detected regions enclosed by bounding boxes, and *(b)* resulting bounding boxes after fusion.

After the merging step another filtering phase takes place. Bounding boxes that are too small, too big, or featuring an aspect ratio incompatible with a human shape are discarded. Finally, the size of resulting bounding boxes is refined. In fact, thanks to the knowledge of the road slope computed as described in section A.4.1, it is possible to compute the point of contact between each object framed by a bounding box and the ground. Thus, the bounding boxes' bottoms are stretched to the ground [35] (see figures A.9). This allows to include legs when they are misdetected and, at the same time, to remove portions of the background below the human shape when it has been incorrectly included in a bounding box.

Bounding boxes registration

At the end of the previous stages two lists of bounding boxes have been produced: one for the obstacles detected in the visible domain, and another one for the obstacles



(a)

(b)

Figura A.9: Two different cases of bounding boxes refinement for the two stereo flows: bounding boxes *(a)* before and *(b)* after the bottom refinement.

detected in the infrared images. Due to the different extrinsic and intrinsic calibration parameters of the two stereo systems, a registration phase is required. In fact, even bounding boxes that correspond to the same obstacle in the two lists are differently sized and positioned. Therefore, in order to fuse the results obtained working on the two spectral domains, it is necessary to compute the position and size of bounding boxes of each list in the other domain. Moreover, the different field of view of the two stereo systems has to be taken into account; in this case, visible cameras have a larger

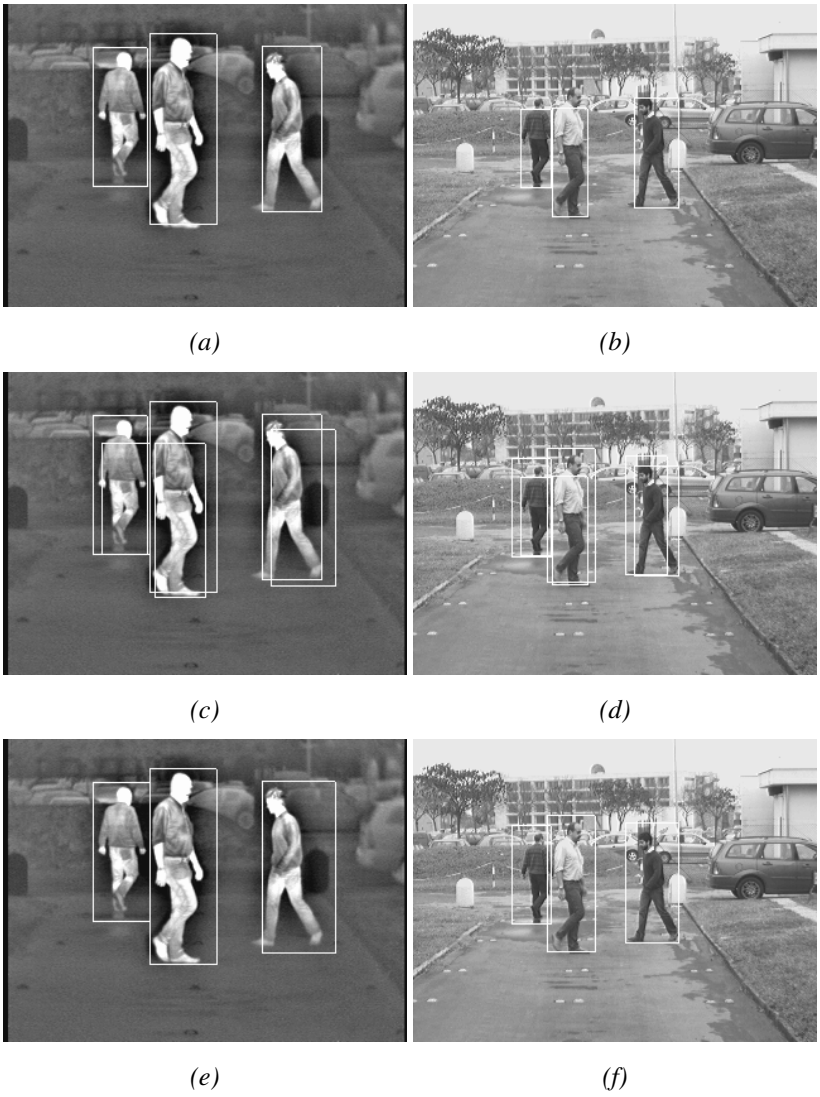


Figura A.10: Bounding boxes registration and fusion: bounding boxes computed in the *(a)* visible and *(b)* infrared domains and their union in the *(c)* infrared and *(d)* visible domains; boxes fusion in *(e)* infrared and *(f)* visible images.

field of view, so not all detected objects in such domain have a correspondent in the infrared images. Thanks to the knowledge of the calibration data and road slope, the bounding boxes computed in the visible domain are resized, repositioned according to the infrared vision system setup, and finally added to the bounding boxes detected in the infrared domain. Bounding boxes that are completely or partially out of the field of view of the infrared stereo system are removed or cropped. The same process is performed to compute size and position of bounding boxes detected in the infrared images towards the visible domain.

Cross-domain fusion of results

Since most of the obstacles in front of the vision system are correctly detected both in the visible and in the infrared domains, the two separate processings can produce two different bounding boxes for the same obstacle. Moreover, during the previous step, the two lists of bounding boxes have been merged; therefore, in each list two bounding boxes can refer to the same object so that another merge step is required. The merging is operated using a bounding box that encloses the two bounding boxes to be fused together. To determine if two bounding boxes refer to the same object, the percentage of overlapping in the images and their distance in world coordinates are considered (see figure A.10). At the end of this stage the two lists of bounding boxes contain the same number of bounding boxes, and each bounding box in a list have an homologous one in the other list.

A.5 Symmetry-based refinement

The previously discussed low-level obstacle detectors generate a bounding boxes list. Unfortunately, the same bounding box sometimes may enclose different pedestrians especially if they are at a similar distance from the cameras (figure A.12.*a*). In addition, both objects and pedestrians can be detected.

Therefore a refinement and filtering process has been developed to get one bounding box for each object and to get rid of non-pedestrian obstacles. More specifically, a symmetry-based processing is used to split, refine and preliminary validate

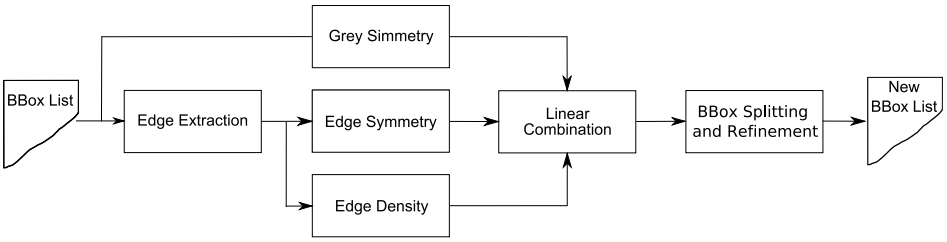


Figura A.11: Flow of symmetry processing.

each bounding box. In fact, pedestrians are generally symmetrical, then a symmetry computation can be used both as a validator and as refinement step.

Experimental results demonstrated that infrared images are more suitable than daylight images for a symmetry computation, since they contain a lower number of small details. Therefore, this phase is performed on the right infrared images only; the whole process is depicted in figure A.11. This step produces a updated bounding boxes list for both infrared and visible domains.

Two different vertical symmetry measures are performed: one on the grey-level values and one on the vertical edge values. The density of edges is also considered to remove homogeneous edgeless areas. For each bounding box all the possible positions of a symmetry axis are considered. For each axis, different symmetry window widths are evaluated considering the size of a human shape at the bounding box distance. Two symmetry maps are computed: one for grey levels (figure A.12.*b*) and one for edges (figure A.12.*c*). In these maps, the horizontal axis encodes the symmetry axis horizontal position within the bounding box; the vertical axis represents the symmetry window width. Figure A.12.*d* shows the density map that is computed in a similar way to the one used for the symmetry map. The triangular shape of the maps is due to the limitation in scanning large windows for peripheral vertical axis.

Bright points encode the presence of a high symmetric content: the value of each symmetry map pixel depends on the number of symmetric points in the related symmetry window. In the grey level symmetry computation, two points are regarded as symmetric if they are equidistant from symmetry axis and have about the same

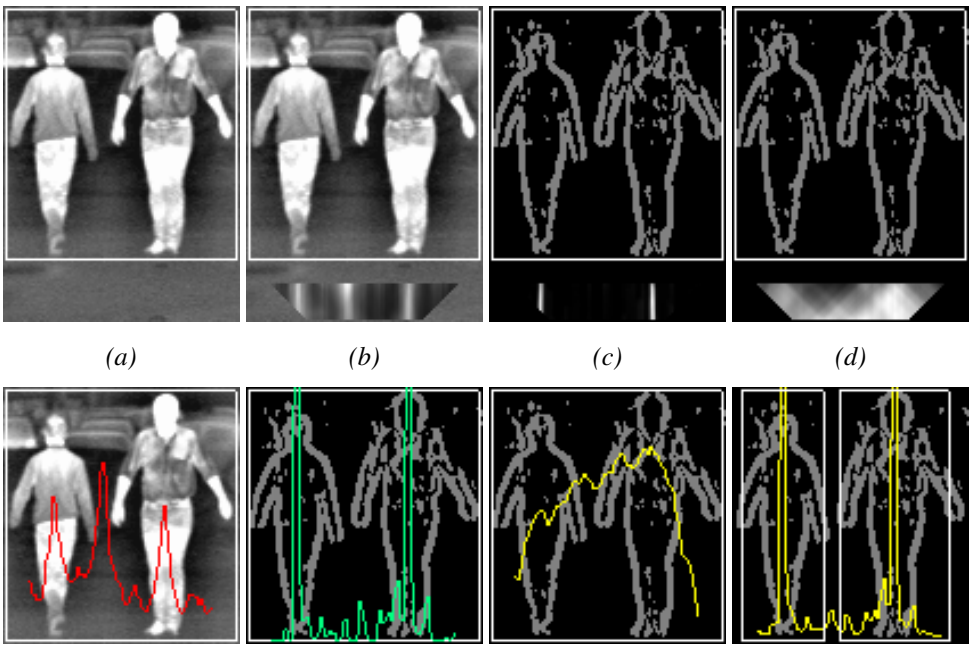


Figura A.12: Symmetry computation steps: *(a)* original bounding box, *(b)* grey and *(c)* edge symmetry maps, *(d)* edge density map, *(e)* grey and *(f)* edge symmetry histograms, *(g)* edge density histograms, *(h)* overall histogram and new splitted bounding boxes.

grey value. Conversely, in the edge symmetry computation, they must also feature an opposite edge sign in order to match edges belonging to the same object (figure A.13).

Let $A = (x_a, y_a)$ and $B = (x_b, y_b)$ be two symmetric edge points with $x_a < x_b$ and let $\rho_a, \rho_b \in \pm 1$ be respectively their edge sign. A and B are regarded as *warm symmetric edge points* if $\rho_a = 1$ and $\rho_b = -1$, namely if A is a transition from dark to bright and B is a bright to dark one. The edge symmetry computation mainly exploits the warm symmetric edge points. This particular approach is aimed to regard as symmetric the objects that are brighter than background. Also the match between

bright to dark and dark to bright transitions is considered for computing the symmetry maps; in fact, when, given an axis position and a symmetry window, the edge symmetry is higher than a given threshold, also this contribution is computed and added to the symmetry map. The underlying ratio of this approach is to further enhance the symmetry weight related to warm objects.

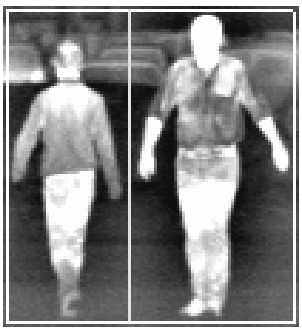


Figura A.13: A wrong axis is not detected thanks to the use of edge sign.

Grey level symmetry (figure A.12.*e*), edge symmetry (figure A.12.*f*) and edge density (figure A.12.*g*) histograms are computed scanning the related maps and detecting the highest values for each column. Symmetry histograms and edge density are then combined to compute the overall histogram (figure A.12.*h*), as shown in the following formula:

$$TotSym = (EdgeSym + k \times GreySym) \times EdgeDens$$

where k is a constant value used to weight grey level symmetry contribution.
A new bounding boxes list is then generated by thresholding the resulting histogram. Therefore, for each bounding box, the over-threshold peaks generate new bounding boxes that contain the shape of a potential pedestrian. This step allows both to refine bounding boxes width and to split bounding boxes that contain more than one object.

The new bounding boxes width is, in fact, computed maximizing the edge density. Unfortunately, the so-computed bounding boxes tend to be narrower than the object they contain (figure A.14.*a*). Therefore, the bounding boxes are furtherly expanded

using a vertical histogram of edges, in order to better match the objects width (figure A.14.*b*).

(a) *(b)*

Figura A.14: The bounding boxes before and after the expansion. The yellow areas show the refined bounding box; the white border depicts the bounding box before the symmetry based step.

Also the symmetry content of each bounding box is added to the bounding box list. In fact, the list of refined and filtered bounding boxes is used by a following validator stages, and by the final decision step, as depicted in figure A.1.

A.6 Human shape detection

After the symmetry step, the list of bounding boxes that contains potential pedestrians is further investigated to validate the presence of a human shape. Three different validation processes are currently used to evaluate the boxes content. One is based on the search of the most evident feature of a human shape, namely the presence of a head. The two others are based on the evaluation of the overall shape of the object: an active contour based shape detection and the use of a probabilistic model. A final rules validator is used to compute the probability that a box contains a human shape and to discard boxes whose probability is too low.

A.6.1 Active contour models

Active contour models are widely used in pattern recognition for extracting an object shape. First introduced by [39], this topic has been extensively explored also in the last years. Basically, a snake is a curve described by the parametric equation $\mathbf{v}(s) = (x(s), y(s))$, where s is the normalized length, assuming values in the range $[0,1]$. This continuous curve becomes, in a discrete domain, a set of points, that are pushed by some energies that depend on the specific problem being addressed. Every point composing the snake reaches a local energy minimum; this means that the active contour does not find a global optimum position; rather, since it is based on local minimization, the final position strongly depends on the initial condition, i.e. the initial snake position.

Because initial stages of the pedestrian detection system provide a bounding box for each detected object, the snake initial position can be chosen as the bounding box contour; then, a contracting behavior should be impressed, to force the snake to move inside the bounding box. Other energies must also be introduced to make the snake stop when the object contour is reached.

Forces, and associated energies, can be divided into two different categories: internal and external. Internal energy only depends on the topology of the snake, and controls the continuity of the curve derivatives; it is evaluated by the equation:

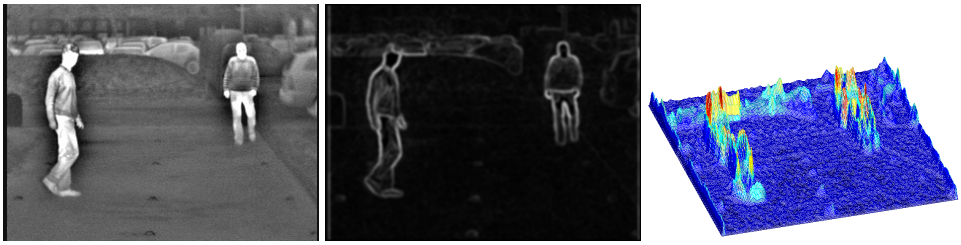
$$E_{int} = \alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2 \tag{A.3}$$

where $\mathbf{v}_s(s)$ and $\mathbf{v}_{ss}(s)$ are, respectively, the first and second derivatives of $\mathbf{v}(s)$ with respect to s. The first contribution appearing in the sum represents the tension of the snake, that is responsible for the elastic behavior; the second one gives the snake resistance to bending; $\alpha(s)$ and $\beta(s)$ are weights.

Therefore, internal energy controls the snake mechanical properties, but is independent of the image; external energy, on the contrary, causes the snake to be attracted to the desired features, and should therefore be a function of the image.

Analytically, the snake will try to minimize the whole energy balance, given by the equation:

$$E_{snake} = \int_0^1 \left(E_{int}\left(\mathbf{v}(s)\right) + E_{ext}\left(\mathbf{v}(s)\right)\right) \mathrm{d}s \tag{A.4}$$



(a) *(b)* *(c)*

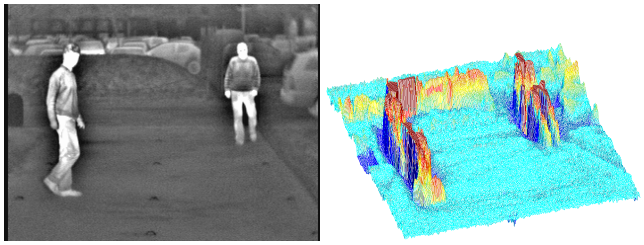
Figura A.15: Energy field due to edges: *(a)* original image, *(b)* edge image obtained using Sobel operator and gaussian smooth, *(c)* edge energy functional with inverted sign, to obtain a more effective graphical representation.

Because energies are the only way to control a snake, a proper choice of both internal and external energies should be made. In particular, the external energy depending on the image must decrease in the regions where the snake should be attracted. In the following, the energies adopted to obtain an object shape are described.

As previously said, the initial snake position was chosen to be along the bounding box contour. In this system both visible and far infrared images are available, but the latter seem much more convenient when dealing with pedestrians, due to the thermal difference between a human being and the background [40].

To extract a pedestrian shape, the Sobel filter output is a useful starting point; moreover, the edge image is needed also by previous steps of the recognition algorithm, so it is already available. A Gaussian smoothing filter is then applied to enlarge the edges, and therefore the area capable of influencing the snake behavior. The resulting image is then associated with an energy field that pushes the snake towards the edges: for this reason, the brighter a pixel in that image, the lowest the associated energy; in this way, *snaxels* (the points into which the snake is discretized) are attracted by the strongest edges, see figure A.15.

Bright regions of the original FIR image are also considered. In fact, smoothed edges does not accurately define the object contour (mainly because they are smoothed), so snake contraction has to be arrested by bright regions in the FIR image,



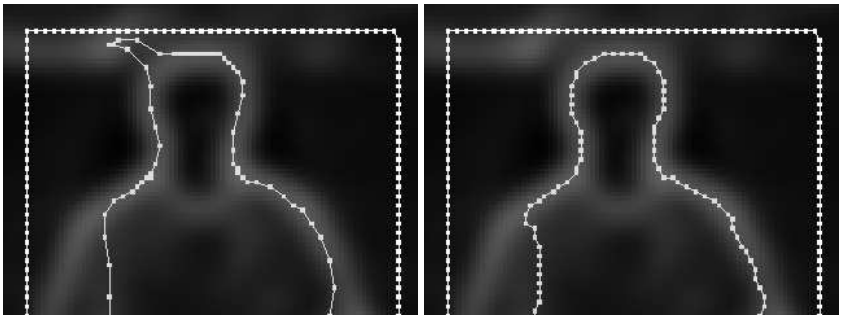
(a) *(b)*

Figura A.16: Energy field due to the image: *(a)* original image, *(b)* intensity energy functional with inverted sign, to obtain a more effective graphical representation.

that can belong to a portion of a human body (see figure A.16). This method lets the snake correctly adapt to a body shape in a lot of situations, and it should also be noticed that this mechanism works only if there are hot regions inside the bounding box; a useful side effect, then, is an excessive snake contraction when there are not warm blobs inside a bounding box. The minimum energy location is found by iteratively moving each snaxel, following an *energy minimization algorithm*. Many of them were proposed in the literature. For this application, the greedy snake algorithm [41], applied on 5×5 neighborhood, was adopted. During the initial iterations, the snake tends to contract, due to the elastic energy; this tendency stops when some other energy counterweights it: for instance, the presence of edges, or a light image region. While adapting to the object shape, the snake length decreases, as well as the mean distance between two adjacent snaxels. Since this mean distance is a value that affects the internal energy, to keep almost constant the elastic property also during strong contraction, the snake is periodically resampled using a fixed step; in this way some unwanted snaxels accumulation can be avoided (figure A.17).

Double snake

The active contour technique turned out to be effective, but it showed some weaknesses when adapting to concave shapes, like those created by a pedestrian when



(a) *(b)*

Figura A.17: Final configuration of the snake without resampling *(a)* and with resampling *(b)*. White dots represent snaxels of the snake in its initial position; yellow dots compose the snake in its final configuration. In *(b)* the correct snaxel disposition and the absence of the artifact towards the head are evident.

his legs are open. In this case, the active contour needs to sensibly extend his length while wrapping around the concave shape, but this process is usually not complete because of the elastic energy. Moreover, the initialization, that is, the initial configuration of the snake, strongly influences the shape extracted at the end of the process. To increase the capability of adapting to concave shapes, and to partially solve the dependence on the initialization, [42] proposed a technique based on two snakes: a snake external to the shape to recover, like the one previously discussed, and a new one, placed inside the pedestrian shape, that tends to adapt from inside, driven by a force that makes the snake expand, instead of contracting. Moreover, the two snakes do not evolve independently, but rather interact; how they do that is a key point in the development of this technique. The simplest interaction is obtained by adding in equation A.4 a contribution that depends on the position of the other snake, so that each one tends to move towards the other.

Note, however, that there is no guarantee that the two snakes will get very close, as there can be strong forces that make the two snakes remain far from each other; for this reason, the tuning of the parameters in the energy calculation should be ca-

refully performed, so that the force between the two contours can balance the other components. This task turns out to be particularly difficult when dealing with images taken in the automotive scenario, that usually present a huge amount of details and noise; it is in fact very difficult to find a set of parameters providing a good attraction between the two snakes, and, at the same time, letting them free of moving towards the desired image features.

Alternatively, the snake evolution can be controlled by a new behavior, that ensures that the two snakes will get very close to each other. Such behavior is based on the idea that, at each iteration, every snaxel should move towards the corresponding snaxel on the other snake. Snaxels are therefore coupled, so that each snaxel in one snake has a corresponding one in the other contour. Then, during the iteration process, snaxels couples are considered: for each of them, one of the point is moved towards the other one, the latter remaining in the same position; the moving point is chosen so that the energy of the couple is minimized. In general, the number of points is different for the two snakes, this meaning that a snaxel of the shorter contour can be included in more than one couple: such points have a greater probability of being moved, but this effect doesn't jeopardize the shape extraction.

In this approach the energy balance is still considered, but here it has a slightly different meaning, because it is used to choose which snaxel in the couple should move. This gives a great power to the force that attracts the two snakes, and the drawback is that they can therefore neglect the other forces, namely the features of the image that should attract them. To mitigate this power, every two iterations with the new algorithm, an iteration with the classical greedy snake algorithm is performed, so that the snakes are better influenced by the image and by the internal energy. This solution turned out to be the most effective one.

Some examples and performance comparisons of contour extraction are presented in figure A.18; in the left column, a simple case: the contour of the same pedestrian is extracted using the single snake technique (*a*), and the double snake (*c*). Then, in (*b*) and (*d*) a more complex scene is considered: together with a pedestrian, some other obstacles are detected in the frame; all the contours are extracted for the classification. In this case it can be analyzed the behavior of the shape extractor when dealing with

obstacles other than pedestrians, that are usually colder than a human being: as a result, in the FIR images they will appear dark, and will therefore lack the features that attract the snakes. In this situation, contours extracted using the double snake algorithm (*d*) tend to become similar to a square, and are clearly different from the shape of a pedestrian; this difference is not so high using the single snake technique, as can be seen in (*b*).

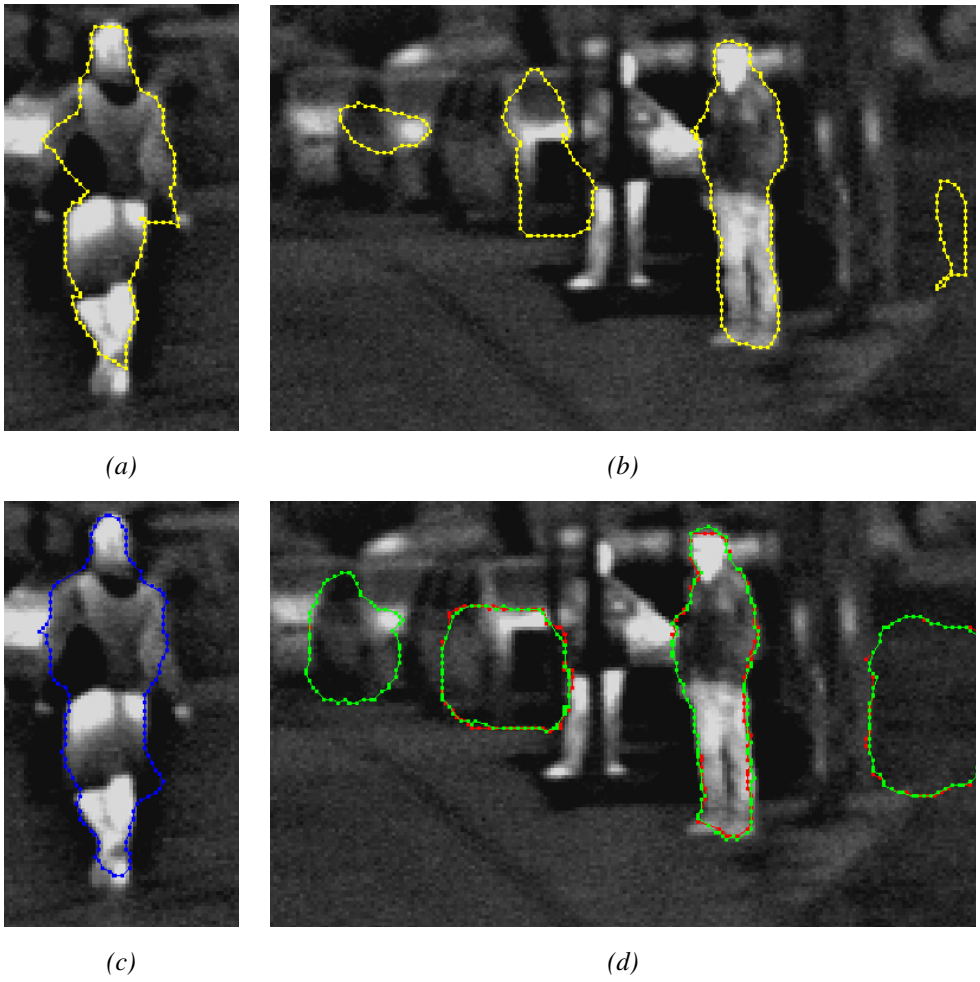
Neural network classification

Once the shape of each obstacle is extracted, it has to be classified, in order to obtain a vote to provide to the final validator. Obstacles shapes extracted using the active contour technique are validated using a neural network.

Prior to be validated, extracted shapes should be further processed: the neural network needs a given number of input data, but each snake has a number of points that depends on its length. For this reason, each snake is resampled with a fixed number of points, and the coordinates are normalized in the range [0; 1]. The neural network has 60 input neurons, two for each of the 30 points of the resampled snake, and only one output neuron, that provides the probability that the contour represents a pedestrian; such probability will be, again, in the range [0; 1].

For the training of the network, a data set of 1200 pedestrian contours and roughly the same number of contours of other objects has been used. They have been chosen in a lot of short sequences of consecutive frames, so that each pedestrian appeared in different positions, but avoiding to use too many snakes of the same pedestrian. During the training phase, the target output has been chosen as 0.95 and 0.05 for pedestrians and non pedestrians, respectively; extreme values, like 0 or 1, have been avoided, because they could have produced some weighting parameters inside the network to assume a too high value, with negative influence on the performance.

This classificator was tested on several sequences. Recall that the output of the neural network is the probability that an obstacle is a pedestrian; it is therefore interesting to analyze which values are assigned to pedestrians and other objects on the test sequences. Output values of the network are shown in figure A.19: (*a*) represents the output values distribution when pedestrians are classified, while (*b*) is the distri-



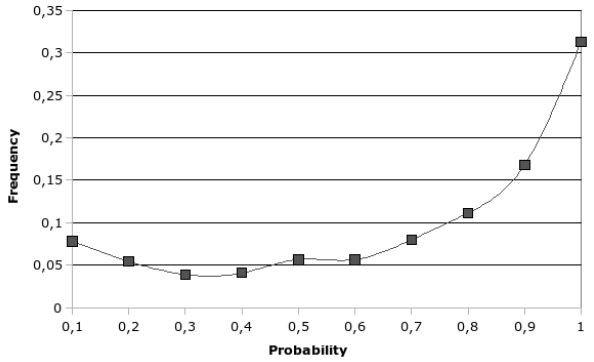
(a) *(b)*

(c) *(d)*

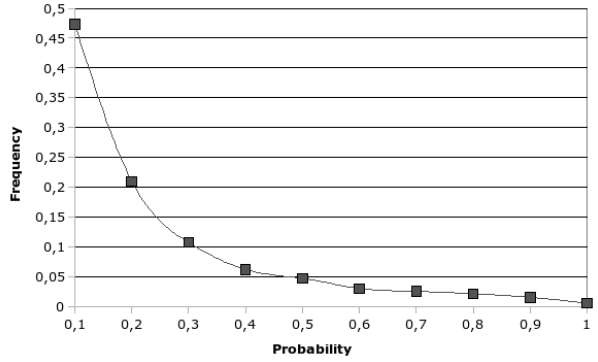
Figura A.18: Examples of shape extraction. In *(a)*, the contour of a pedestrian is extracted using the single snake algorithm, while *(c)* shows the result when the double snake technique is used; it can be seen that the contour is smoother in the latter case. In *(b)* a more complex situation analyzed using the single snake technique, and *(d)* presents the same scene analyzed by the double snake algorithm (the red contour is the inner one, while the green snake is the outer one).

bution when contours of objects that are not pedestrians are analyzed. It can be seen that classification results are accurate, and this classificator was therefore included in the global system depicted in figure A.1.

Moreover, the performance was evaluated also considering this classificator by itself, and not as a part of a greater system. A threshold was therefore calculated to obtain a hard decision; the best value turned out to be 0.4, which provided a correct



(a)

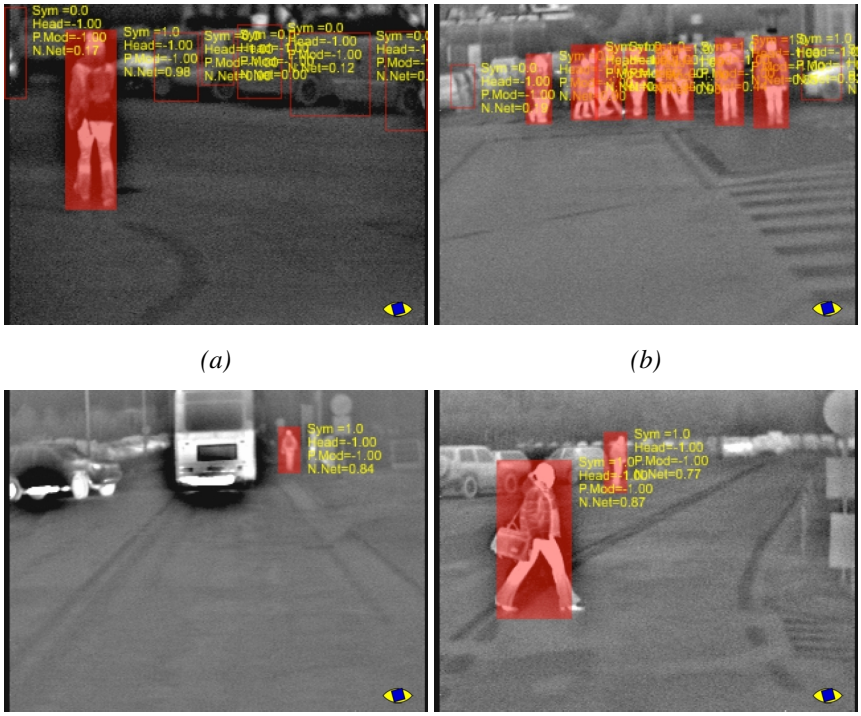


(b)

Figura A.19: Distribution of the neural network output values. On the x-axis is plotted the probability values given by the neural network, while on the y-axis is reported the occurrence of each probability value when the shape of pedestrians *(a)* and other objects *(b)* are analyzed.

classification of 79% of pedestrians and 85% of other objects.

In figure A.20 some classification results of the neural network that analyzes pedestrians shapes are shown. In figure A.20.*a*, a lot of potential pedestrians are found by the obstacle detector of previous system stages, but only one is classified as a pedestrian, with a vote of 0.98, while all the other obstacles received a vote not greater than 0.17. In figure A.20.*b*, a scene with a lot of pedestrians, and two obstacles: the latter received votes not exceeding 0.19, while one of the pedestrians received a vote



(a) *(b)*

(c) *(d)*

Figura A.20: Classification results of the neural network analyzing pedestrians shapes. Bounding boxes that are filled are classified as pedestrians, while a red contour is put aroung obstacles that are classified as non pedestrians. Output values are also printed on the image.

of 0.44, and all the others votes greater than 0.85. In figure A.20.*c*, a distant pedestrian is correctly classified with a vote of 0.84; in figure A.20.*d* two pedestrians are present, at different distances, and are correctly classified, with votes of 0.87 and 0.77.

A.6.2 Neural network

This section describes the neural network based validator, shown in figure A.1. A feed-forward multi-layer neural network is exploited to evaluate the presence of pedestrians in the bounding boxes detected by previous stages. Since neural networks can express highly nonlinear decision surfaces, they are especially appropriate to classify objects that present a high degree of shape variability, like a pedestrian. A trained neural network can implicitly represent the appearance of pedestrians in various poses, postures, sizes, clothing, and occlusion situation.

In the system here described, the neural network is directly trained on infrared images. Generally, neural network-based systems, working on daylight images, do not exploit directly the image; in fact, it is not appropriate for encoding the pedestrian features, since pedestrians present a high degree of variability in color and texture and, moreover, intensity image is sensitive to illumination changes. Conversely, in the infrared domain the image of an object depends on its thermal features and therefore it is nearly invariant to color, texture and illumination changes. The thermal footprint is a useful information for the neural network to evaluate the pedestrian presence and therefore, it is exploited as a direct input for the net (figure A.21). Since neural network needs a fixed sized input ranged from 0 to 1, the bounding boxes are resize and normalized.

The net has been designed as follows: the input layer is composed by 1200 neurons, corresponding to the number of pixels of resized bounding boxes (20×60). The output layer contains an single neuron only and its output corresponds to the probability that the bounding box contains a pedestrian (in the interval [0,1]). The net features a single hidden layer. The number of neurons in the hidden layer has been computed trying different solutions; values in the interval 25 - 140 have been considered.

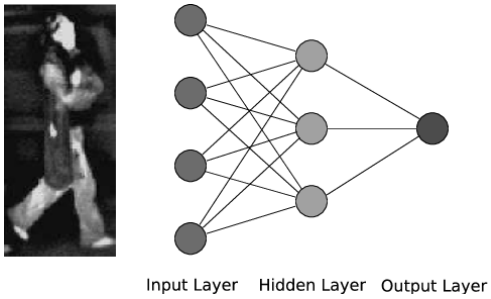


Figura A.21: A three-layer feed-forward neural network: each neuron is connected to all neurons of the following layer. The infrared bounding boxes are exploited as input of the network.

The network has been trained using the back-propagation algorithm. The training set is generated from the results of the previous detection module that were manually labelled. Initially, a training set, composed by 1973 examples, has been created. It contains 902 pedestrians, and 1071 non pedestrians examples ranging from traffic sign poles, vehicles, to trees. Then, the training set has been expanded to 4456 examples (1897 of pedestrian and 2559 of non pedestrian) in order to cover different situations and temperature conditions and to avoid the overfitting. Moreover, an additional test set has been created in order to evaluate the performance of the validator.

The network parameters are initialized by small random numbers between 0.0 and 1.0, and are adapted during the training process. Therefore, the pedestrian features are learnt from the training examples instead of being statically predetermined. The network is trained to produce an output of 0.9 if a pedestrian is present, and 0.1 otherwise. Thus, the detected object is classified thresholding the output value of the trained network: if the output is larger than a given threshold, then the input object is classified as a pedestrian, otherwise as a non pedestrian.

A weakness of the neural network approach is that it can be easily overfitted, namely the net steadily improves its fitting with the training patterns over the epochs, at the cost of diminishing the ability to generalize to patterns never seen during the training. The overfitting, therefore, causes an error rate on validation data larger than

the error rate on the training data. To avoid the overfitting, a careful choice of the training set, the number of neurons in the hidden layer, and the number of training epochs, must be performed.

In order to compute the optimal number of training epochs, the error on validation data set is computed while the network is being trained. The validation error decreases in the early epochs of training but after a while it begins to increase. The training session is stopped if a given number of epochs has passed without finding a better error on validation set and if the ratio between error on validation set and error on training set is greater than a specific value. This point represents a good indicator of the best number of epochs for training and the weights at that stage are likely to provide the best error rate in new data.

The determination of number of neurons in the hidden layer is a critical step as it affects the training time and generalization property of neural networks. Using too few neurons in the hidden layer, the net results inadequate to correctly detect the patterns. Too much neurons, conversely, decreases the generalization property of the net. Overfitting, in fact, occurs when the neural network has so much information

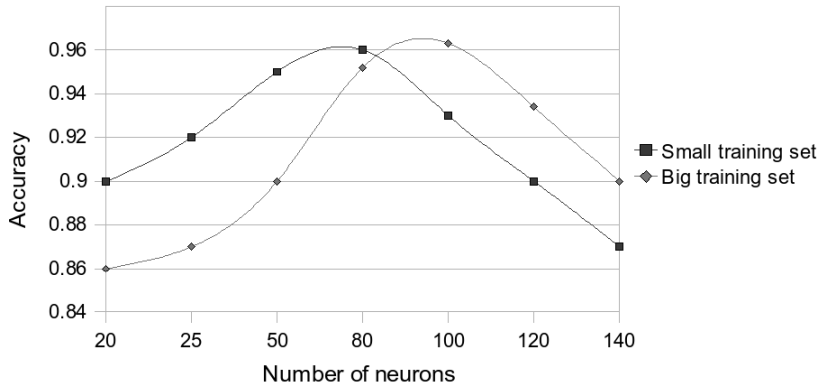


Figura A.22: The accuracy of the net on validation set depending on the number of neurons in hidden layer. The optimal neurons number is a trade off between underfitting and overfitting.

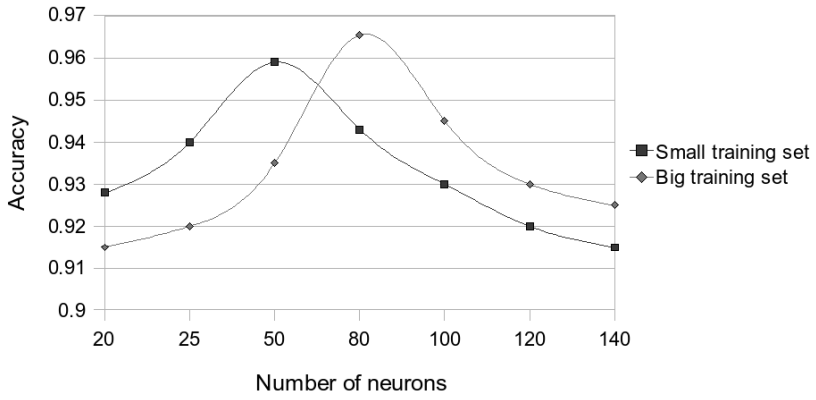


Figura A.23: The accuracy of the net on test set depending on the number of neurons in hidden layer.

processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layer. In figure A.22, the accuracy of the net on validation set depending on the number of neurons in hidden layer is shown. With a larger training set, a bigger number of neurons in the hidden layer is required. This is caused by the bigger complexity of the training set, that contains pedestrians in different conditions; Therefore, a net with more processing capacity is needed.

The trained nets have been tested on the test set that is strictly independent to the training and validation set. It contains examples of pedestrian and non pedestrians in various poses, shapes, sizes, occlusion status and temperature conditions. In figure A.23, the accuracy of the net on test set varying the number of neurons in hidden layer is shown. The performance of the nets, trained on the big training set, is greater than that trained on the small set. This is caused by a more completeness of the training set. The performance of nets is similar to that performed on validation set (figure A.22); but the optimal number of neurons in the hidden layer is lower. The net having 80 neurons in the hidden layer and trained on big training set, is the best one, achieving an accuracy of 96.5% on the test set.

A.6.3 Probabilistic models

A match against probabilistic models of a human shape is also used to validate bounding boxes [43, 32, 44]. To make this validation more reliable and to detect the pedestrian pose, four different probabilistic models that comprise a number of pedestrian poses and orientation are used for the match (figure A.24). Currently, the models are classified considering the position of the legs during the walking process: open, almost open, almost closed, and fully closed legs. Four different training sets of images that contain human shapes in these different poses have been used to obtain the four probabilistic models. The training sets also show people that walk in front or laterally with respect to the camera axis, to take into account different angles of view for human shapes.

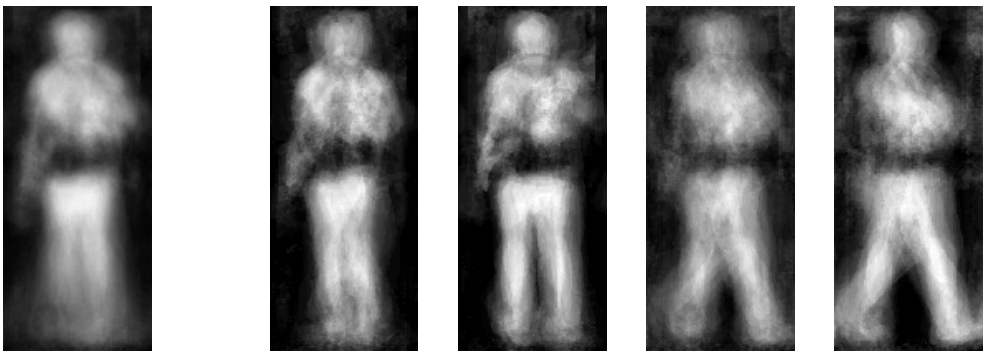
Figura A.24: A few examples of training data set images.

Figure A.25 shows the resulting models used for the recognition step. These models are created off-line, and therefore their computation is not time consuming. Initially a mask model (figure A.25.*a*) is used to remove the background and to enhance the foreground of the bounding boxes. The mask is computed as the average of the four probabilistic models. Since sometimes the bounding boxes are not sufficiently accurate, the search region is enlarged proportionally to each bounding box size and the mask model is scaled according to this search region (figure A.26). To enhance the

foreground and remove the background, the following formula is used:

$$I(x,y) = \begin{cases} 255 & \text{if } p(x,y) \geq 0.75 \text{ AND } I(x,y) > 127 \\ 0 & \text{if } p(x,y) \leq 0.20 \text{ AND } I(x,y) \leq 127 \\ \text{I(x,y)} & \text{otherwise} \end{cases} \tag{A.5}$$

where $I(x,y)$ are the values of search region pixels, and $p(x,y)$ are the values of the mask pixels.



(a) *(b)* *(c)* *(d)* *(e)*

Figura A.25: *(a)* Mask-model obtained considering all kind of poses and models used for the recognition step: *(b)* closed *(c)* almost closed *(d)* almost open *(e)* open legs.

Empirically, it was found that regions where $p(x,y) \geq 0.75$ corresponds to head and torso and $p(x,y)$ is lower than 0.20 in correspondence to the background. Intermediate situations are generally due to legs or arms.

After this phase, the different models are matched against the extracted foreground. Each model is resized to fit the corresponding search region. For each point (x,y) within the search area a correlation probability $cp(x,y)$ is computed using the formula:

$$cp(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} (th_{xy}(i,j) - 127) \times (p(i,j) - 0.5) \tag{A.6}$$

(a) *(b)* *(c)*

Figura A.26: Foreground enhancement and background removal: *(a)* original bounding boxes, *(b)* search areas (blue) and rectangles (red) that correspond to the size of the mask model and *(c)* final result.

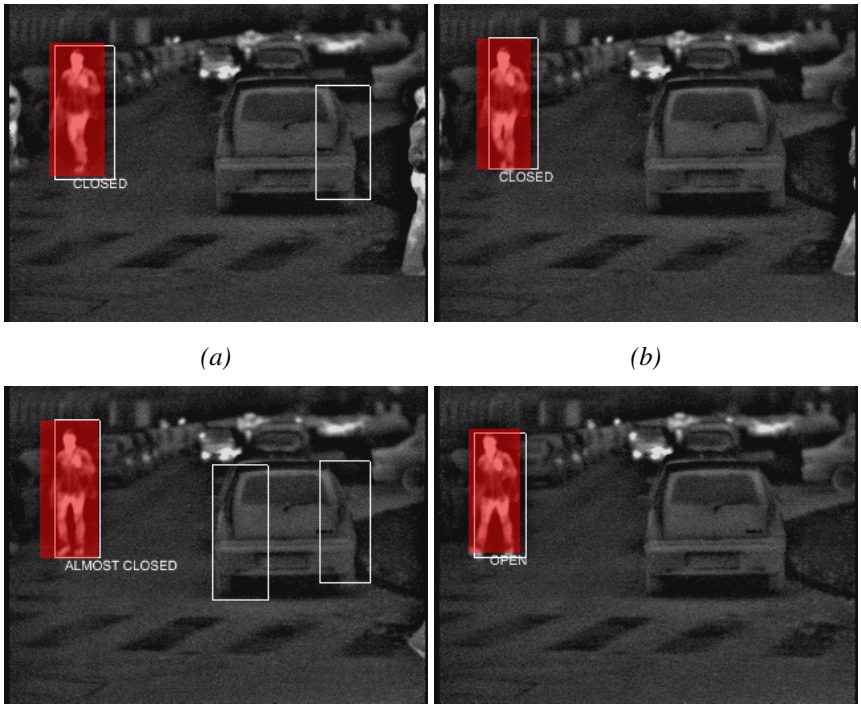
where m and n are the model width and height, $th_{xy}(i,j)$ is the image after the enhancement operation. The function $cp(x,y)$ encodes the probability that the area $m \times n$ centered in (x,y) contains a pedestrian.

Since four different models are used, the classification is solved as a maximization problem. The model that gets the higher probability value is considered as the best match and the probability itself is considered as the final vote and fed to validator. Bounding boxes featuring a probability lower than a given threshold are discarded (figure A.27).

A.6.4 Head detection

For each potential pedestrian the presence of the head is evaluated. This step is performed on the FIR domain only, because the head is the most evident feature of a human shape in the FIR domain, due to the high heat dispersion, which makes pixels brighter than the background. In addition, the position of the head is barely affected by pedestrian pose, being always in the upper part of the bounding box.

The head detection system exploits three different approaches: two of them are based on a model match, the last one on the search of warm areas. The use of a number of different approaches has been selected in order to minimize the risk of false negatives.



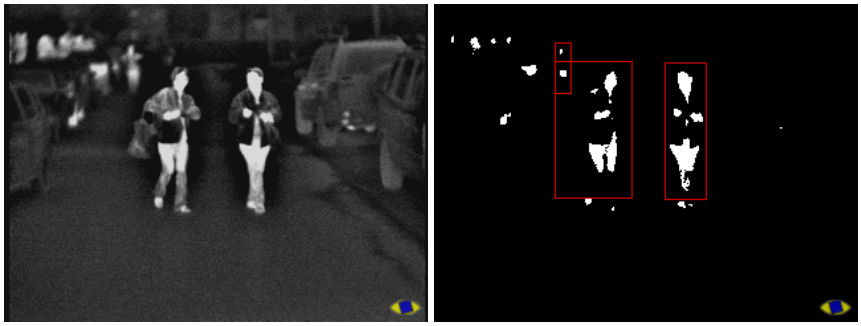
(a) *(b)*

(c) *(d)*

Figura A.27: Probabilistic model results: pedestrians *(a)* and *(b)* are labelled as having closed legs, in *(c)* the pedestrian is labelled as having the legs almost closed, and in *(d)* as open legs.

Pattern matching

The first approach relies on a pattern matching technique. Two different models of a head are used in order to perform different matching operations. The first model encodes thermal characteristics of a head warmer than the background, namely this model is a binary mask showing a white head on a black background (figure A.29.*a*). To ease the match, the areas of attention are binarized using an adaptive threshold (figure A.28). The model is scaled according to the bounding boxes size and assuming that a head measures nearly $1/6$ of a human shape height. The match is performed on



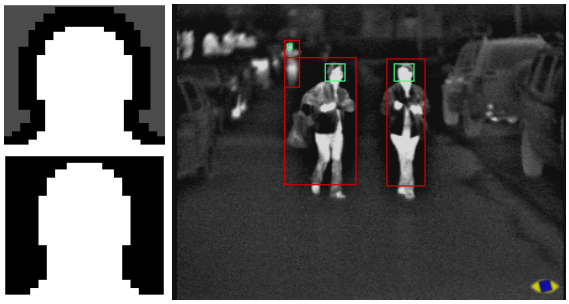
(a) *(b)*

Figura A.28: Head detection by pattern matching: *(a)* original image and *(b)* binarized image.

an area centered around the top of the bounding box using equation (A.1) discussed in section A.4.2. The highest correlation value obtained is considered as the match quality (P_w). Unfortunately, the head is not always warmer than the background. Environmental conditions, hats, helmets, or hair may mask heat radiation. In order to cope with this problem, an additional head model is used. This model encodes the head shape (figure A.29.*a*) and is used to perform another match in the top area of each bounding box. In this case, the areas of attention are not binarized. For each position of the model, the average values of pixels that correspond to the internal (white) or external (black) part of the model are computed. The quality of the match is considered as the absolute value of the difference between these two averages. A higher difference is obtained in correspondence to objects that feature a shape similar to the model. The shape matching quality (P_s) is computed as the highest of such differences. The final match parameter (P_m) is computed as $P_m = 1 - ((1 - P_w) \times (1 - P_s))$ and fed to the rules validator.

Probabilistic model

The probabilistic approach defines a matching method between a head image and a suitable model [32]. The underlying idea is the same as that described in sec-



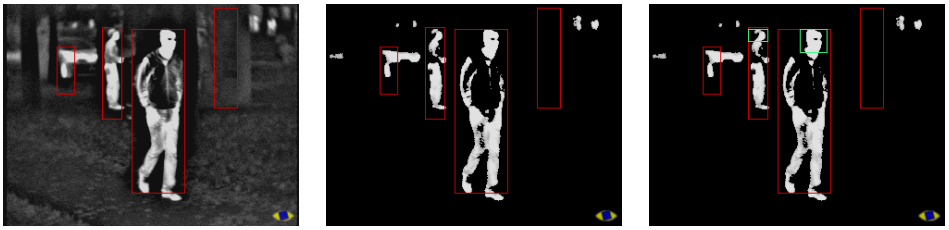
(a) *(b)*

Figura A.29: Head detection by pattern matching: models *(a)* the two models used for head matching, *(b)* final result.

tion A.6.3, but the implementation is slightly different. The region of interest focuses on the head; as previously discussed, the model is generated using a number of head images. The *training set* images show pedestrians in several positions and movements. Then, for each pixel the probability that it belongs to a pedestrian head is computed. The probabilistic head model (figure A.30) is then resized according to the bounding box width to perform the match.

Figura A.30: Head model by probabilistic matching.

For each bounding box the area of attention for the matching is located in the upper part. The height of the searching area is, again, about $1/6$ of the box height. To ease the match, a threshold is applied to remove the background. Moreover, bright zones are enhanced using a morphological filter (figure A.31); the match is performed using the equation (A.6) discussed in section A.6.3 and the result is a probability map.



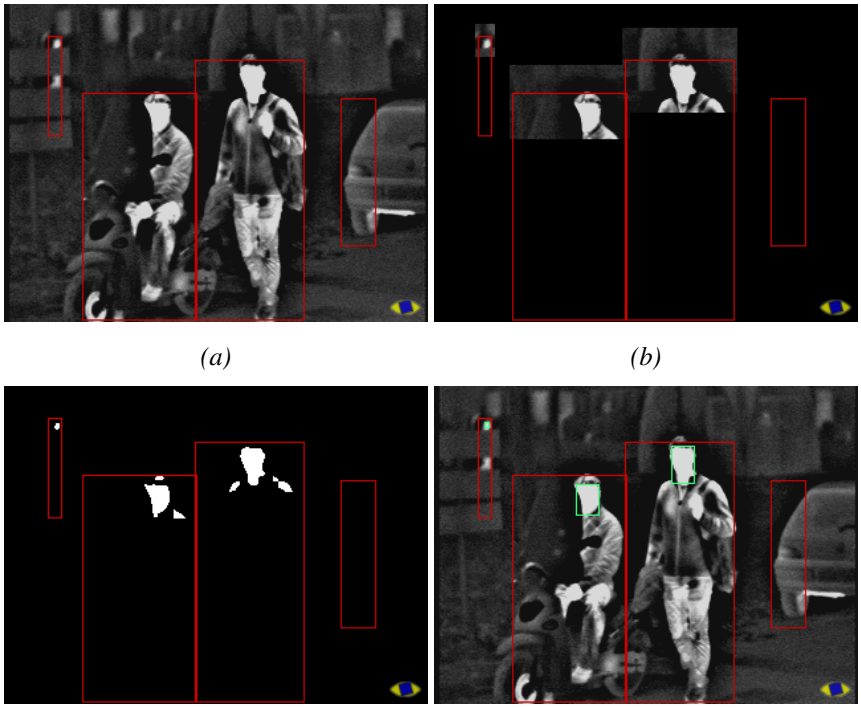
(a) *(b)* *(c)*

Figura A.31: Head matching by probabilistic model: *(a)* detected obstacles, *(b)* preprocessed image and *(c)* final result.

This map contains all the probabilities values greater then a threshold. Then the local maximum is found and considered as the best candidate for the head position in the bounding box; the correspondent correlation probability value is provided on to the rules validator (figure A.31.*c*).

Warm areas search

The last approach is based on warm areas analysis. This search criterion exploits, as a preprocessing phase, the double threshold algorithm described in section A.4.2. Different operations are then performed due to the different aim of the search. The upper side of the bounding box is considered as a search area. moreover, since sometimes bounding boxes are not sufficiently accurate, this search area is slightly enlarged, proportionally to the bounding box size. After the preprocessing phase, warm areas are labelled to detect contiguous areas. These areas are then enclosed in smaller bounding boxes, that become candidates for the head position. The biggest warm area, that satisfies head-like criteria, is chosen as the best match for the head. Criteria are based on head aspect ratio and on the ratio between the head area and the bounding box area. Figure A.32 shows the different process steps.



(a) *(b)*

(c) *(d)*

Figura A.32: Head matching by warm area search: *(a)* detected obstacle, *(b)* searching area, *(c)* preprocessed image and *(d)* final result.

A.7 Validation and results

The results of each validation step are used to filter out false positives. In the current system the vote fed by each validator is linearly combined to compute a final *fitness* value. This simple approach has proven to be not sufficiently effective, since it does not take into account the different behavior of each validator. In fact, each validator features different perception characteristics that are hard to join together. For example, a validator can feature a low false positive rate as well as a medium false negatives percentage; in such a case, when a human shape is detected the probability,

it is a false positive, is low; at the same time, when a bounding box is discarded, the risk it contains a pedestrian is not negligible. Therefore, a single vote for each validator is not sufficient to evaluate the presence of a pedestrian.

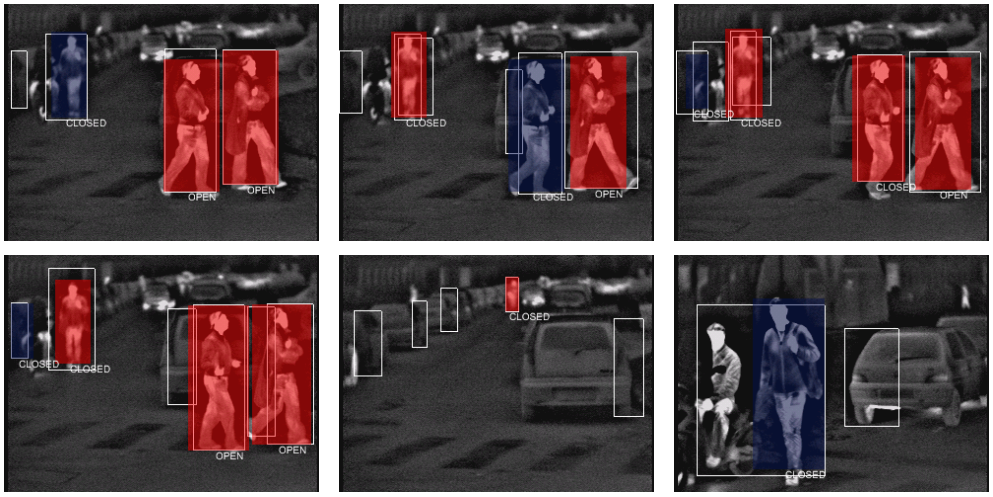


Figura A.33: Results of the probabilistic model validators: detected pedestrians are shown using a superimposed red box when the confidence on the classification is high, or blue when the confidence is low.

Figure A.33 shows a few results of the validator based on the probabilistic model. If the confidence on the detection is above a threshold, the corresponding box is drawn in red, meaning that the region has a high probability of containing a human shape, or in blue when the confidence is low.

Figure A.34 shows the results of the symmetry computation step. This allows both to refine bounding boxes width, to split bounding boxes that contain more than one object and to validate them, filtering out edgeless or asymmetric bounding boxes. Some symmetrical objects, like cars are trees, are validated as well. Moreover, some validation problems are encountered when the far infrared images are not optimal, like those acquired in summer under heavy direct sunlight; in these conditions, many object in the background become warm, and the assumption that a pedestrian features a higher temperature than the background is not satisfied. This causes some problems

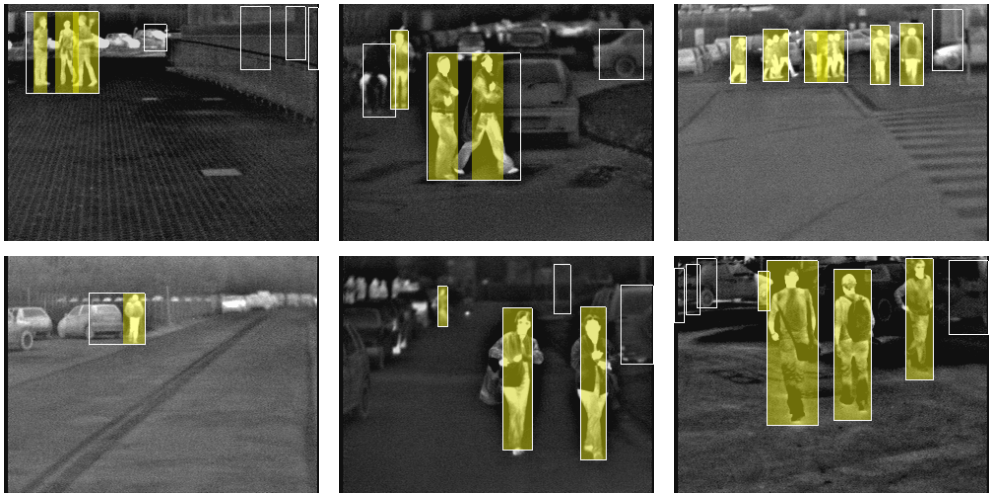


Figura A.34: Symmetry analysis results: validated pedestrians are shown using a superimposed yellow box; the white rectangles are the bounding boxes generated by previous steps before the validation, splitting or resizing.

in the edge and symmetry computation, and therefore in the results.

The result of active contours-based validation is shown in figure A.35. It can be noticed that computed contours fit well with pedestrian shapes. A match with a model has been tested to validate the result before passing it to the rules validator; nevertheless, a more sophisticate filter is currently under development.

Figure A.36 shows some results of the neural network validator. The validated pedestrians are shown using a superimposed solid red box. Conversely, the empty rectangles represent the bounding boxes generated by previous steps and classified as non pedestrians.

Figure A.37 shows the results of the three different approaches for head detection. The use of different approaches allows to reduce the number of false negatives, especially when the head is not much warmer than the background.

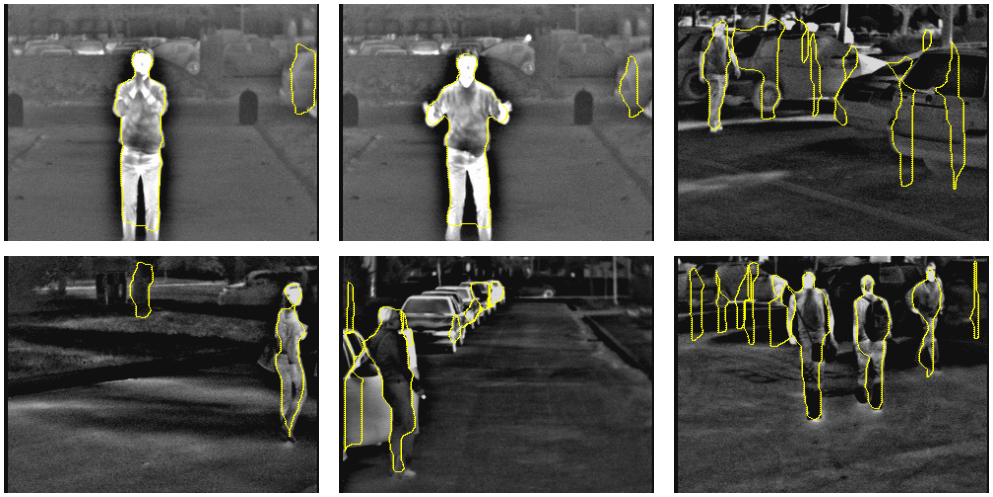


Figura A.35: Some examples of contour extraction using snakes: the active contour model is applied to each detected obstacle. The contour extraction gives good results also when some background details are present.

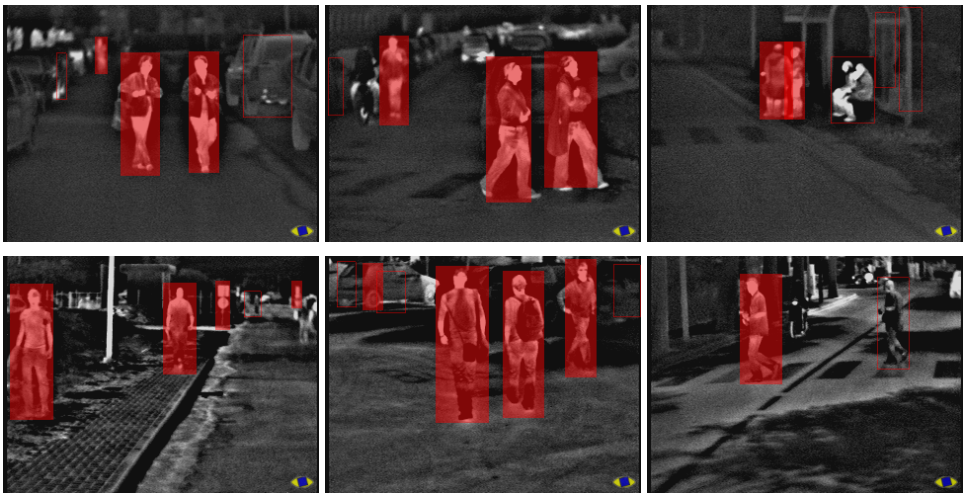


Figura A.36: Neural network results: validated pedestrians are shown using a superimposed red box; the empty rectangles represent the discarded bounding boxes.

Figura A.37: Results of the head detector validator: green bounding boxes show heads detected inside the original areas of attention in red.

A.8 Conclusions

In this appendix, a system aimed at the detection of pedestrians exploiting both a FIR and a visible daylight systems has been presented. It has been tested in urban and rural environments using an experimental vehicle equipped with four cameras, two infrared cameras that work in the 7–14 μm spectrum range and two CMOS-based CCD cameras.

The algorithm is based on the use of different approaches for detecting an initial list of areas of attention: the warm areas and vertical edges detection in the FIR domain and the DSI-based computation in both domains. These different approaches independently produce different lists of areas of attention; therefore, a merging step is used for coalescing these lists in a single one. Thanks to a precise knowledge of the calibration parameters and to the stereo-based computation of the road slope, distance, size and position of each area are computed and areas of attention that feature parameters not compatible with the presence of a pedestrian (or a small group of pedestrians) are discarded.

Thanks to the assumption that a human shape is mostly symmetrical, a following symmetry-based process is used to refine and furtherly filter out the areas of attention. Finally, to validate the presence of pedestrians in the surviving areas, a number of validators are used: a head presence detector, an active contours-based processing, and a match with a probabilistic model of human shape.

Preliminary results demonstrated that this approach is promising. The use of sensors that work in different domains allowed to increase the detection rate with respect to previous similar systems [35] and to detect pedestrians even if they are not warmer than the background or in low illumination conditions. Moreover, the symmetry-based refinement step and the use of some validators allow the system to detect pedestrians where they were not detected by a previous system version [45].

Bibliografia

[1] Care - european road accident database, 2009. Available at http://ec.europa.eu/transport/roadsafety/road_safety_observatory/care_en.htm.

[2] COWI. Cost-benefit assessment and prioritisation of vehicle safety technologies. Technical Report TREN/A1/56-2004, European Commission Directorate General Energy and Transport, January 2006.

[3] European Commission. Proposal for a regulation of the european parliament and of the council concerning type-approval requirements for the general safety of motor vehicles. Available at http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:52008PC0316:EN:NOT.

[4] Olivier Faugeras, Bernard Hotz, Hervé Mathieu, Thierry Viéville, Zhengyou Zhang, Pascal Fua, Eric Théron, Laurent Moll, Gérard Berry, Jean Vuillemin, Patrice Bertin, and Catherine Proy. Real-time correlation-based stereo : algorithm, implementations and applications. Technical Report 2013, INRIA, August 1993.

[5] L. Di Stefano, M. Marchionni, S. Mattoccia, and G. Neri. A fast area based stereo matching algorithm. *Image and Vision Computing*, 22:983–1005, 2004.

[6] Raphaël Labayrade, Didier Aubert, and Jean-Phillipe Tarel. Real Time Obstacle Detection in Stereo Vision on non Flat Road Geometry through "V-Disparity" Representation. In *Procs. IEEE Intelligent Vehicles Symposium 2002*, Paris, France, June 2002.

[7] Alberto Broggi, Claudio Caraffi, Pier Paolo Porta, and Paolo Zani. The Single Frame Stereo Vision System for Reliable Obstacle Detection used during the 2005 Darpa Grand Challenge on TerraMax. In *Procs. IEEE Intl. Conf. on Intelligent Transportation Systems 2006*, pages 745–752, Toronto, Canada, September 2006. arXiv:doi:10.1109/ITSC.2006.1706831.

[8] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.

[9] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[10] L. Di Stefano, M. Marchionni, and S. Mattoccia. A pc-based real-time stereo vision system. *MG&V*, 13(3):197–220, 2004.

[11] Andrei Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Professional, Indianapolis, February 2001. Available from: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0201704315.

[12] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177–185, 1991.

[13] R. Kania, Mike Del Rose, and P. Frederick. Autonomous Robotic Following Using Vision Based Techniques. In *Procs. Ground Vehicle Survivability Symposium*, Monterey, USA, 2005.

[14] Mike Del Rose and P. Frederick. Pedestrian Detection. In *Procs. Intelligent Vehicle Systems Symposium*, Traverse City, USA, 2005.

[15] Mike Del Rose, P. Frederick, and J. Reed. Pedestrian Detection for Anti-Tamper Vehicle Protection. In *Procs. Ground Vehicle Survivability Symposium*, Monterey, USA, 2005.

[16] K. C. Frerstenberg, J. Dietmayer, and V. Willhoeft. Pedestrian Recognition in Urban Traffic using Vehicle Based Multilayer Laserscanner. In *Procs. Automobile Engineers Cooperation International Conf.*, Paris, France, 2001.

[17] K. C. Frerstenberg and U. Lages. Pedestrian Detection and Classification by Laserscanners. In *Procs. IEEE Intelligent Vehicles Symposium 2002*, Paris, France, June 2002.

[18] A. Fod, A. Howard, and M. J. Mataric. Laser-Based People Tracking. In *Procs. IEEE Intl. Conf. on Robotics and Automation*, Washington D. C., USA, 2002.

[19] NIT Phase II: Evaluation of Non-Intrusive Technologies for Traffic Detection. Technical Report SRF No. 3683, Mn DOT Research Report, 2002.

[20] S. Milch and M. Behrens. Pedestrian Detection with Radar and Computer Vision. In *Procs. Conf. on Progress in Automobile Lighting*, Darmstadt, Germany, 2001.

[21] M. Koltz and H. Rohling. 24 GHz Radar Sensors for Automotive Applications. In *Procs. Intl. Conf. on Microwaves and Radar*, Warsaw, Poland, 2000.

[22] A. Shashua, Y. Gdalyahu, and G. Hayun. Pedestrian Detection for Driving Assistance Systems: Single-frame Classification and System level Performance. In *Procs. IEEE Intelligent Vehicles Symposium 2004*, Parma, Italy, June 2004.

[23] G. P. Stein, O. Mano, and A. Shashua. Vision based ACC with a Single Camera: Bounds on Range and Range Rate Accuracy. In *Procs. IEEE Intelligent Vehicles Symposium 2003*, Columbus, USA, June 2003.

[24] L. Zhao. *Dressed Human Modeling, Detection, and Parts Localization*. Ph.D. dissertation, Carnegie Mellon University, 2001.

[25] Constantine Papageorgiou, Theodoros Evgeniou, and Tomaso Poggio. A Trainable Pedestrian Detection System. volume 38, pages 15–33, June 2000.

[26] David Beymer and Kurt Konolige. Real-time Tracking of Multiple People using Continuous Detection. In *Procs. Intl. Conf. on Computer Vision*, Kerkyra, 1999.

[27] H. Shimizu and T. Poggie. Direction Estimation of Pedestrian from Multiple Still Images. In *Procs. IEEE Intelligent Vehicles Symposium 2004*, Parma, Italy, June 2004.

[28] K. Grauman and T. Darrell. Fast Contour Matching Using Approximate Earth Mover's Distance. Technical Report AI Memo, AIM-2003-026, MIT, 2003.

[29] Ross Cutler and Larry S. Davis. Robust Real-time Periodic Motion Detection, Analysis and Applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):781–796, August 2000.

[30] S. Tate and Y. Takefuji. Video-based Human Shape Detection Deformable Templates and Neural Network. In *Procs. of Knowledge Engineering System Conf.*, Crema, Italy, 2002.

[31] J. W. Davis and V. Sharma. Robust Background-Subtraction for Person Detection in Thermal Imagery. In *Procs. Intl. IEEE Wks. on Object Tracking and Classification Beyond the Visible Spectrum*, Washington D. C., USA, 2004.

[32] Harsh Nanda and Larry Davis. Probabilistic Template Based Pedestrian Detection in Infrared Videos. In *Procs. IEEE Intelligent Vehicles Symposium 2002*, Paris, France, June 2002.

[33] B. Bhanu and J. Han. Kinematics-based Human Motion Analysis in Infrared Sequences. In *Procs. IEEE Intl. Workshop on Applications of Computer Vision*, Orlando, USA, 2002.

[34] Fengliang Xu and Kikuo Fujimura. Pedestrian Detection and Tracking with Night Vision. In *Procs. IEEE Intelligent Vehicles Symposium 2002*, Paris, France, June 2002.

[35] Massimo Bertozzi, Alberto Broggi, Michael Del Rose, and Andrea Lasagni. Infrared Stereo Vision-based Human Shape Detection. In *Procs. IEEE Intelligent Vehicles Symposium 2005*, pages 23–28, Las Vegas, USA, June 2005. arXiv:doi:10.1109/IVS.2005.1505072.

[36] Raphael Labayrade and Didier Aubert. A Single Framework for Vehicle Roll, Pitch, Yaw Estimation and Obstacles Detection by Stereovision. In *Procs. IEEE Intelligent Vehicles Symposium 2003*, pages 31–36, Columbus, USA, June 2003.

[37] Alberto Broggi, Claudio Caraffi, Rean Isabella Fedriga, and Paolo Grisleri. Obstacle Detection with Stereo Vision for off-road Vehicle Navigation. In *Procs. Intl. IEEE Wks. on Machine Vision for Intelligent Vehicles*, San Diego, USA, June 2005.

[38] Glenn Hines, Zia ur Rahman, Daniel Jobson, and Glenn Woodell. Multi-image registration for an enhanced vision system, April 2002.

[39] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Intl. Journal of Computer Vision*, 1(4):321–331, 1988.

[40] Massimo Bertozzi, Alberto Broggi, Claudio Caraffi, Mike Del Rose, Mirko Felisa, and Guido Vezzoni. Pedestrian Detection by means of Far-infrared Stereo Vision. *Computer Vision and Image Understanding*, 106(2):194–204, May–June 2007. arXiv:doi:10.1016/j.cviu.2006.07.016.

[41] Donna J. Williams and Mubarak Shah. A Fast Algorithm for Active Contours and Curvature Estimation. *CVGIP: Image Understanding*, 55(1):14–26, 1992.

[42] S. R. Gunn and M. S. Nixon. A Robust Snake Implementation: a Dual Active Contour. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(1):63–68, January 1997. arXiv:doi:10.1109/34.566812.

[43] L. Lee, G. Dalley, and K. Tieu. Learning pedestrian models for silhouette refinement. In *Procs. IEEE International Conf. on Computer Vision*, Nice, France, 2003.

[44] A. Senior. Tracking people with probabilistic appearance models. In *ECCV workshop on Performance Evaluation of Tracking and Surveillance Systems*, 2002.

[45] Massimo Bertozzi, Alberto Broggi, Mirko Felisa, Guido Vezzoni, and Michael Del Rose. Low-level Pedestrian Detection by means of Visible and Far Infrared Tetra-vision. In *Procs. IEEE Intelligent Vehicles Symposium 2006*, pages 231–236, Tokyo, Japan, June 2006. arXiv:doi:10.1109/IVS.2006.1689633.

Ringraziamenti

Prima di tutti vorrei ringraziare il mio supervisore, Alberto Broggi, per il suo l'aiuto e il suo appoggio.

Un grazie, anche, a tutti i ragazzi del laboratorio e della palazzina 1: Pierpa, Michele, Kappa, Maria Chiara, Isabella, Paolo G, Pietro, Paolo Z, Alessandro, Stefano D, Andrea F, Luca G, Giorgia, Luca M, Alessandro, Matteo, Paolo M, Sefano C, Elena, Bomba e Massimo.

Infine ringrazio la mia famiglia e gli amici che in tutti questi anni non mi hanno mai fatto mancare il loro supporto umano.