

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXIV Ciclo

**TECNICHE DI VISIONE ARTIFICIALE PER
ELABORAZIONE DI IMMAGINI PROVENIENTI DA
TELECAMERE FISSA E MOBILE CON APPLICAZIONI
NEI CAMPI AUTOMOTIVE E VIDEOSORVEGLIANZA**

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Alberto Broggi

Dottorando: *Luca Mazzei*

Gennaio 2012

*a Roberta,
alla mia famiglia,
ai miei amici,
al GPA.*

Sommario

1	Introduzione	1
1.1	Motivazioni della ricerca	2
1.2	Installazione telecamere fissa e mobile	4
1.3	Progetti coinvolti	5
1.4	Maggiori contributi di questa ricerca	7
1.5	Struttura della tesi	8
2	Elaborazione di immagini con camera fissa	9
2.1	Controllo di accesso a varchi	10
2.1.1	Scopo del progetto, prototipazione e sperimentazione	13
2.2	Stato dell'arte controllo e conteggio degli accessi a varchi mediante telecamera	13
2.3	Contapersone con telecamera singola	17
2.3.1	Panoramica dell'algoritmo	17
2.3.2	Classificazione	20
2.3.3	Risultati ottenuti	21
2.4	Contapersone stereo	23
2.4.1	Panoramica dell'algoritmo	23
2.4.2	Preprocessing	25
2.4.3	Image Analysis	27
2.4.4	People Counting	39
2.4.5	Risultati ottenuti	45

2.5	Conclusioni	47
3	Elaborazione di immagini con camera mobile	49
3.1	Sistema per il riconoscimento di pedoni e veicoli	50
3.2	Algoritmo per il riconoscimento di pedoni in immagini NIR	50
3.2.1	Stato dell'arte	50
3.2.2	Panoramica dell'algoritmo	51
3.2.3	Blocco Pedestrian Classifier	59
3.2.4	AdaBoost Cascade	60
3.2.5	Risultati ottenuti	60
3.3	Riconoscitore di veicoli per applicazioni ADAS	61
3.3.1	Breve stato dell'arte dei riconoscitori di veicoli	61
3.3.2	Algoritmo per il riconoscimento di veicoli	63
3.4	Adas di supporto ai sensori del veicolo	68
3.5	Sistema per il riconoscimento di tunnel stradali	69
3.5.1	Image Processing	71
3.5.2	State Update e State FSM	71
3.5.3	Validazione ingresso	81
3.5.4	Validazione uscita	82
3.5.5	Risultati ottenuti	85
3.6	Sistema per il riconoscimento e la classificazione di ambienti stradali	86
3.6.1	Algoritmo generale	87
3.6.2	Scelta delle classi e training set	87
3.6.3	Image Processing	88
3.6.4	FFT e campionamento	90
3.6.5	Scene Classifier	93
3.6.6	Classificazione mediante PCA	93
3.6.7	Classificazione mediante Neural Networks	96
3.6.8	Classificazione mediante SVM	98
3.6.9	Tracking	99
3.6.10	Risultati ottenuti	99

3.6.11	Tempi di esecuzione	103
3.7	Conclusioni	105
4	Conclusioni	107
A	Classificazione e Apprendimento Supervisionato	111
A.1	Reti Neurali	112
A.1.1	Apprendimento on-line e off-line	114
A.1.2	<i>Backpropagation</i>	115
A.1.3	Overfitting	117
A.1.4	Valutazione delle prestazioni	118
A.2	Support Vector Machines	119
A.2.1	SVM lineare	122
A.2.2	SVM non lineare	123
A.2.3	Proprietà	125
A.2.4	SVM Multiclasse	127
B	Algoritmi di boosting	129
B.1	Introduzione agli algoritmi di boosting	129
B.2	Introduzione ad AdaBoost	130
B.3	AdaBoost versione M1	131
B.4	AdaBoost versione M2	134
C	Principal Component Analysis	139
C.1	Introduzione	139
C.2	Singular Value Decomposition	140
C.3	Principal Component Analysis	140
	Bibliografia	145

Elenco delle figure

1.1	Installazioni telecamera fissa: (a) sopra una porta, (b) lungo una strada.	4
1.2	Installazioni telecamera mobile su veicoli autonomi: (a) BRAiVE, (b) Grandeur, (c) Porter.	5
2.1	Struttura del sistema di controllo di un varco	12
2.2	Prototipi di varco	12
2.3	Struttura dell'algoritmo sistema di conteggio degli accessi a un varco con singola telecamera	18
2.4	Immagini legate ai passi dell'algoritmo per il controllo di un varco con singola telecamera	19
2.5	Albero di classificatori conta persone con telecamera singola	21
2.6	Esempio di conteggio errato con incrocio	24
2.7	Esempio di conteggio corretto con passaggio di due persone	24
2.8	Esempio di conteggio corretto con incrocio	24
2.9	Struttura dell'algoritmo del sistema di conteggio degli accessi a un varco con camera stereo	25
2.10	Stadio di Preprocessing.	28
2.11	Diagramma di flusso dell'algoritmo di labeling utilizzato	29
2.12	Fase di etichettatura a sinistra l'immagine in input e a destra l'output della fase di etichettatura: (a) due persone disgiunte, (b) scena con tre persone adiacenti, (c) scenario affollato.	31

2.13	Blob slicing: (a) rappresentazione intuitiva dell'operazione di slicing, (b) immagine originale, (c) (d) (e) DSI con tagli a varie quote. . . .	32
2.14	Voronoi Splitting: (a) immagine di input, (b) maschera di Voronoi, (c) DSI tagliata dalla maschera, (d) etichettatura finale.	33
2.15	Head Detection: (a) modello per la ricerca della testa (b) griglia tri-dimensionale (c) posizionamento della griglia per il matching del template.	36
2.16	Head Detection: (a) Scenario comune, (b) esempio con eliminazione di blob non testa, (c) esempio con eliminazione blob non persona. .	40
2.17	Algoritmo di tracking utilizzato nel conta persone stereo.	43
2.18	Head Detection: attraversamento della linea da parte del centro di massa e diagramma di flusso per (a) uscita e (b) entrata	46
3.1	Struttura dell'algoritmo di riconoscimento di pedoni	52
3.2	(a) bounding box contenente un pedone, (b) immagine filtrata con Sobel, (c) immagine filtrata con Sobel verticale con istogramma verticale, (d) immagine filtrata con Sobel orizzontale con istogramma orizzontale.	53
3.3	Maschere utilizzate per il pattern matching. (a) Maschera per la ricerca delle gambe aperte. (b) Maschere piccole per la ricerca delle gambe. (c) Maschere grandi per la ricerca delle gambe. (d) Maschere per ricerca della testa.	54
3.4	Bounding box contenenti: (a) pedoni, (b) alberi e pali.	56
3.5	Risultati riconoscitore pedoni: immagine e ROC	61
3.6	Struttura dell'algoritmo riconoscimento di veicoli	63
3.7	Riconoscimento di veicoli: (a) immagine di input, (b) bounding box estratti con pre-elaborazione, (c) veicolo, (d) forma a "W" (e) bordi orizzontali (f) aree scure.	64
3.8	Risultati ottenuti riconoscitore di veicoli	67
3.9	Struttura dell'algoritmo di riconoscimento di tunnel stradali	70
3.10	Macchina a stati finiti riconoscitore tunnel stradali	70

3.11 Rilevazione ingresso e uscita : (a) ricerca del picco degli istogrammi, (b) bounding box generato partendo dagli istogrammi orizzontale e verticale, (c) confronto tra bounding box e area condivisa per la centratura	73
3.12 Validazione ingresso: (a) istogramma dei neri nel caso di entrata, frame più significativi, (b) varianza dell'istogramma dei neri e media delle percentuali sui frame	81
3.13 Validazione uscita: (a) bounding box filtrato e sogliaato prima dell'uscita da un tunnel, (b) bounding box filtrato e sogliaato all'uscita da un tunnel, (c) andamento della media dell'istogramma all'uscita da un tunnel	82
3.14 Struttura dell'algoritmo per la classificazione di ambienti stradali . .	87
3.15 Classi di scenari in ambiente automotive: (a) città, (b) autostrada, (c) extraurbano.	88
3.16 Image processing: (a) frame input, (b) frame con istogramma normalizzato, (c) frame filtrato con filtro passa-alto, (d) immagine ridimensionata.	89
3.17 Elaborazione nel dominio della frequenza: (a) suddivisione in blocchi, (b) FFT per ogni blocco, (c) maschera per filtro di campionamento, (d) FFT campionata per ogni blocco.	91
3.18 Trasformazione dei vettori di due classi tramite PCA	94
3.19 Esecuzione classificatori in città	102
3.20 Esecuzione classificatori in autostrada	103
3.21 Passaggio sotto un ponte	103
3.22 Risultati ottenuti: (a) Passaggio di fianco ad un pullman, scenario autostradale, (b) extraurbano, la PCA lo classifica città, (c) città, erroneamente classificato da PCA e NN come autostrada (d) extraurbano, correttamente classificato dalla PCA e dalle SVM Autostrada. . . .	104
A.1 Funzioni di attivazione per le reti neurali	114
A.2 Nodi interconnessi in una rete neurale	115

A.3	Correzione dell'overfitting	118
A.4	Separazione tramite SVM lineare	120
A.5	Diversi iperpiani di separazione	121
A.6	Trasformazione dello spazio di input in uno spazio multidimensionale	121
A.7	Iperpiano con il margine massimo	124
A.8	Kernel Machine	126
C.1	PCA: allineamento degli assi con le <i>principal components</i>	142

Elenco delle tabelle

2.1	Confronto tra i principali metodi per il controllo di accessi a varchi.	11
2.2	Risultati ottenuti per il sistema di accesso ai varchi con telecamera singola	22
2.3	Risultati ottenuti per il sistema di accesso ai varchi con telecamera stereo.	45
3.1	Risultati dell'algoritmo.	85
3.2	Addestramento con diverse funzioni di apprendimento	97
3.3	Composizione del test set	100
3.4	<i>Correct Detection</i> dei classificatori su tutto il test set.	100
3.5	Matrice di confusione per il classificatore con PCA.	100
3.6	Matrice di confusione per il classificatore con Reti Neurali.	101
3.7	Matrice di confusione per il classificatore con SVM.	102
3.8	Tempi di esecuzione del sistema.	105

Capitolo 1

Introduzione

*Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza*

Dante Alighieri
Inferno, Canto XXVI, vv. 118-120

L'obiettivo che si pone questa tesi di dottorato è l'analizzare e descrivere due differenti approcci della visione artificiale per l'elaborazione di immagini acquisite da telecamera fissa e mobile. L'analisi delle principali differenze tra le tecniche analizzate permette di partire da quanto si trova nello stato dell'arte e allo stesso tempo utilizzando gli algoritmi presenti irrobustire e innovare le tecniche stesse.

I due approcci con telecamera fissa e mobile permettono di avere due distinte visioni del mondo circostante. Nel caso di un approccio con telecamera fissa l'osservatore si trova a poter inquadrare una porzione del mondo che in molti casi potrebbe avere un cambiamento a volte anche repentino, ma nella maggior parte dei casi presenta un'elevata staticità dal punto di vista della scena. Un tipico sistema di visione artificiale con telecamera fissa prevede un'installazione statica, normalmente in una posizione conosciuta, con ampia conoscenza delle problematiche relative alla scena inquadrata. Sono conosciute le possibilità di cambiamenti atmosferici, di illu-

minazione, di temperatura, nel caso di installazione esterna; mentre sono conosciuti strettamente i cambiamenti e la qualità dell'illuminazione e la morfologia in nel caso di installazioni interne. In entrambi i casi si possono prevedere con un certo livello di affidabilità probabili occlusioni dell'inquadratura della telecamera o dello specifico oggetto inquadrato.

Nel caso dell'approccio con telecamera mobile si ha una visione più imprevedibile del mondo circostante. Normalmente questo approccio può essere legato al campo della robotica mobile, con un automa che muovendosi percepisce il mondo con vari sensori tra cui una o più telecamere. Per quanto riguarda il filone di ricerca con telecamera mobile in cui spazia questo lavoro di tesi di dottorato si farà riferimento al settore di ricerca dei veicoli autonomi e ai sistemi avanzati di aiuto alla guida, ADAS. La scena inquadrata, in queste condizioni, è soggetta a repentini cambiamenti per quanto riguarda la maggior parte dei parametri, l'illuminazione, i cambiamenti atmosferici, la tipologia di ambiente circostante, appunto per il cambiamento della stessa. Il mondo circostante è in costante cambiamento per effetto del movimento del veicolo, e date le grandezze in gioco dal punto di vista delle distanze il e del tempo, il cambiamento può essere anche notevole.

Date per assodate le principali differenze, gli approcci presentati basati sull'elaborazione di immagini provenienti da telecamera fissa e mobile possiedono alcuni punti di contatto, che prevedono tecniche di visione artificiali comuni. Entrambi i sistemi possono prevedere o meno calibrazione, quindi richiedere la conoscenza precisa di distanze dati punti noti del mondo associati con punti dell'immagine. Allo stesso modo per il riconoscimento di ostacoli e oggetti presenti nella scena potrebbero essere richiesti approcci mediante singola telecamera o con coppie stereo di telecamere. I sistemi che prevedono questa distinzione hanno generalmente hanno strutture completamente differenti.

1.1 Motivazioni della ricerca

Le motivazioni di questa ricerca sono da inquadrare all'interno del confronto tra tecniche differenti che si basano su approcci in alcuni casi opposti, ma che possono

avere parti seppur piccole in comune.

Dal punto di vista teorico questa ricerca ha permesso di affrontare molte tecniche in uso in settori innovativi della visione artificiale con l'introduzione di algoritmi ed approcci tipici dell'intelligenza artificiale.

I sistemi sviluppati presentati in questa tesi di dottorato attivamente affrontano principalmente due settori che hanno una ricaduta umana molto importante dal punto di vista sociale.

I sistemi che prevedono l'analisi degli accessi ad un varco possono essere inseriti all'interno dell'area di ricerca coperta dalla videosorveglianza, ed in ultima analisi possono essere anche inseriti nell'area di statistica dei flussi di persone per fini commerciali, o di supporto all'analisi di studi del mercato e delle attività a esso connesse. La videosorveglianza come settore di ricerca ha trovato negli ultimi anni una forte spinta dal punto di vista dell'innovazione. I sistemi di videosorveglianza stanno passando da un controllo dell'ambiente passivo ad un tipo di controllo attivo che prevede una segnalazione da parte di sistemi software di situazioni fortemente indicative di comportamenti definiti anomali nella scena inquadrata. Esistono però studi contrastanti sull'utilità dell'utilizzo di telecamere di videosorveglianza per quanto riguarda la lotta o la prevenzione al crimine. In [1] viene mostrato come l'utilizzo di telecamere risulti efficace come deterrente per prevenire i reati; mentre in [2], fermo restando l'importanza dei sistemi di videosorveglianza nel combattere il crimine, emerge come questi sistemi molte volte aumentino la percezione di pericolosità della zona monitorata.

Per quanto riguarda i sistemi sviluppati con telecamera mobile, notiamo come questi possono essere inseriti all'interno della branca della visione artificiale che si occupa di veicoli autonomi e sistemi per l'aiuto alla guida. Da una parte si ha sempre di più l'esigenza, soprattutto da parte delle istituzioni di diminuire con ogni mezzo la quantità di incidenti stradali soprattutto con effetti mortali, dall'altra si pensa sempre di più a realizzare sistemi che possano rendere più piacevole l'esperienza della guida. Il tema degli incidenti stradali viene spesso indicato come motivazione nella ricerca legata al campo dell'automotive, in effetti ogni anno le statistiche dell'ISTAT mostrano come il numero di morti negli incidenti scenda rispetto all'ultimo decennio

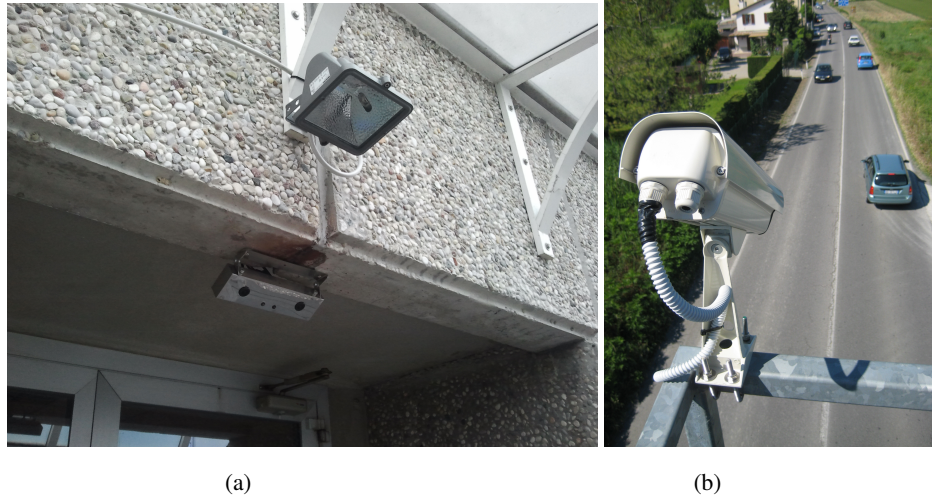


Figura 1.1: Installazioni telecamera fissa: (a) sopra una porta, (b) lungo una strada.

in linea con la media europea, -42.4% . Da un altro punto di vista oggi giorno i sistemi ADAS sono commercializzati su molti veicoli anche di medio livello con prezzi decisamente abbordabili. Su berline di case automobilistiche che hanno forti quote di mercato anche nel segmento delle utilitarie vengono presentati pacchetti con sistemi ADAS avanzati per poche migliaia di euro.

1.2 Installazione telecamere fissa e mobile

La principale differenza che si evidenzia per quanto riguarda gli approcci con telecamere fissa o mobile è stata introdotta nella sezione precedente e riguarda la tipologia di installazione. Questa influenza anche le altre caratteristiche, come la scena inquadrata e il setting dei parametri dovuti all'illuminazione. Nella figura 1.1 sono illustrate due tipiche installazioni per le telecamere fisse, la prima, in figura 1.1.(a), mostra l'installazione di una telecamera stereo sopra una porta per il controllo degli accessi, mentre la seconda, figura 1.1.(b), mostra l'installazione in una strada extraurbana per il monitoraggio dei flussi di traffico.

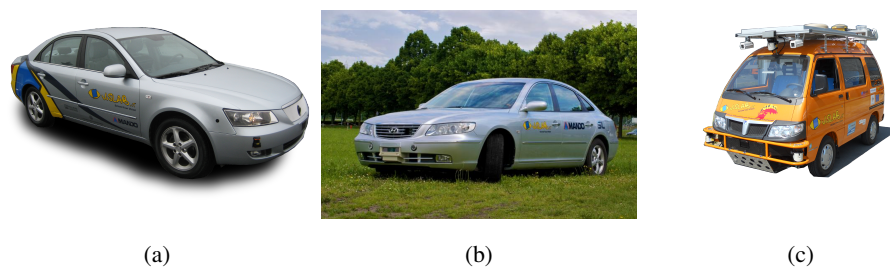


Figura 1.2: Installazioni telecamera mobile su veicoli autonomi: (a) BRAiVE, (b) Grandeur, (c) Porter.

Nella figura 1.2 sono invece mostrati 3 differenti veicoli autonomi, in cui vengono sviluppati i sistemi basati su immagini provenienti da telecamera mobile. Tutti e tre i veicoli sono prototipi del laboratorio di visione artificiale e sistemi intelligenti. In figura 1.2.(a) è mostrata BRAiVE, il veicolo testbed del laboratorio, in figura 1.2.(b) è mostrata una hyundai Grandeur utilizzata per i progetti in collaborazione con Mando Corporation. Infine, in figura 1.2.(c) è mostrato il Porter Piaggio protagonista del viaggio intercontinentale VIAC, che ha permesso di dimostrare i sistemi sviluppati dal laboratorio durante tre mesi di viaggio, con guida completamente automatica, attraversando l'Europa e l'Asia, partendo da Parma con arrivo a Shanghai per un totale di circa 13.000 Km.

1.3 Progetti coinvolti

Le attività di ricerca connesse con il percorso di dottorato sono state inquadrate all'interno di vari progetti, ognuno dei quali con parti in comune, e tutti basati su tecniche innovative di visione artificiale con uno sguardo sempre attento alla letteratura e allo stato dell'arte.

Per quanto riguarda l'area di ricerca legata alla presenza di telecamera fissa sono stati coinvolti due progetti strategici del laboratorio, PeopleCounter e Supermarket, e un progetto in collaborazione con Magneti Marelli, INDUSTRIA2015.

Il progetto PeopleCounter si propone di risolvere il problema del monitoraggio e conteggio delle persone che attraversano un varco, con dimensione arbitraria, con e senza calibrazione, con due differenti approcci confrontabili: mediante telecamera singola e mediante telecamera stereo. Il progetto Supermarket si propone in concerto con quanto ottenuto nel progetto PeopleCounter di estrarre le informazioni relative al numero di persone che sostano all'interno di un dato volume per estrapolare informazioni riguardo al tempo di attesa presso le casse di un supermercato, mediante l'utilizzo di una telecamera stereo.

Il progetto INDUSTRIA2015 è legato alla realizzazione di un sistema per la stima del numero di persone in attesa alla fermata del bus mediante una telecamera stereo in modo da poter gestire in modo ottimo l'apertura delle porte. Esso si basa sulle tecniche utilizzate nei progetti Supermarket e PeopleCounter.

I progetti dell'area di ricerca delle tecniche di visione artificiale basate su camera mobile sono inclusi all'interno della collaborazione tra il laboratorio e la MANDO Corporation. I progetti coinvolti sono:

- APPS (Active Pedestrian Protection System);
- LKS+ACC.

APPS, Active Pedestrian Protection System è finalizzato alla realizzazione di un riconoscitore di pedoni sia diurno che notturno utilizzando sia la visione artificiale che dati provenienti da laserscanner. Un veicolo prototipo è stato equipaggiato a Parma e una copia è stata allestita in Corea con una telecamera con sensibilità nello spettro del vicino infrarosso, NIR (Near Infra-Red), un laserscanner e un apposito illuminatore NIR per la visione notturna. Questo progetto si propone di evitare o, in caso di incidente, attenuare le collisioni con i pedoni che notoriamente sono gli utenti della strada più vulnerabili.

Lo scopo finale del progetto LKS+ACC è l'analisi e l'implementazione di tecniche di elaborazione di immagini e fusione sensoriale per lo sviluppo di un sistema di mantenimento di corsia, LKS (Lane Keeping Assistance), e di controllo attivo di crociera, ACC (Adaptive Cruise Control).

Entrambi i sistemi sono stati installati su un veicolo prototipo e testati in ambiente reale per avere un riscontro realistico delle prestazioni.

Tra i progetti a camera mobile è stato coinvolto un progetto strategico del laboratorio di Visione Artificiale e Sistemi Intelligenti VisLab che prevede la comprensione dello scenario circostante il veicolo *Automotive Scene Classification*. Questo progetto prevede la classificazione mediante tecniche ibride di intelligenza artificiale e visione artificiale dello scenario tramite l'analisi delle immagini provenienti da una singola telecamera posta in posizione frontale in un veicolo. Esso include anche un sottoprogetto dedicato al riconoscimento di tunnel in ambiente stradale. L'applicazione principale di questo progetto è l'informazione del sistema di guida automatica dell'ambiente circostante per migliorare i parametri dei vari sistemi in esecuzione sul veicolo in tempo reale per rendere il veicolo stesso sempre pronto ad eventuali cambiamenti di comportamento.

1.4 Maggiori contributi di questa ricerca

I maggiori contributi di questa ricerca si inseriscono all'interno dei progetti presentati in stretta connessione con le motivazioni illustrate in precedenza. Un primo e importante contributo relativo all'elaborazione di sistemi con telecamera fissa mostra come sia possibile con hardware di tipo commerciale e non altissimi costi realizzare un sistema che unisca le tecniche allo stato dell'arte sui sistemi di visione artificiale basati su sottrazione dello sfondo ed estrazione di oggetti nella scena, e sull'utilizzo di riconoscimento di ostacoli mediante disparità.

Per quanto riguarda l'approccio con telecamera mobile i contributi vanno ricercati all'interno dell'ambito industriale per quanto riguarda la prevenzione degli incidenti che coinvolgono pedoni per il progetto di frenatura automatica APPS. Per quanto riguarda generalmente tutti gli altri progetti si possono pensare applicazioni che spaziano dalla guida completamente automatica, ricordando quanto effettuato in [3], un viaggio di 13.000 Km dall'Italia alla Cina con un veicolo completamente autonomo, fino ad arrivare al semplice ausilio ai sistemi ADAS già presenti all'interno del veicolo.

In entrambi i casi si sono ottenuti dei notevoli contributi alle aree di ricerca coinvolte in termini di risultati, ma anche di innovazione delle tecniche utilizzate e presentate. Per questi motivi gran parte dei sistemi sviluppati durante il lavoro relativo all'attività di ricerca svolta sono stati oggetto di articoli presso conferenze internazionali e riviste o sono in via di pubblicazione, o sono stati utilizzati per realizzare dei prodotti industriali e commerciali.

Maggiori spiegazioni per quanto riguarda i contributi offerti da questa ricerca vengono illustrati nei capitoli dedicati.

1.5 Struttura della tesi

Questa tesi di dottorato è strutturata in due macro argomenti che rispecchiano i principali capitoli. Nel capitolo 2 verranno introdotte le tecniche di elaborazione delle immagini utilizzate nel campo di ricerca legato all'approccio con telecamera singola. Dopo aver inquadrato l'area del problema verranno introdotti nel capitolo due sistemi realizzati che fanno uso di tecniche legate all'elaborazione con telecamera singola e telecamera stereo per il controllo degli accessi ad un varco.

Nel capitolo 3 verranno introdotte le tecniche di elaborazione dell'immagine legate all'approccio con telecamera mobile. In particolare verranno presentati dei sistemi realizzati all'interno del filone di ricerca automotiva per veicoli autonomi.

Nel capitolo 4 verranno tratte le conclusioni sul lavoro e le attività di ricerca svolte riguardo i propositi presenti nelle motivazioni della ricerca stessa inserite nell'introduzione e riguardo i sistemi presentati nei capitoli precedenti.

I sistemi presentati nel capitolo 3 fanno ampio uso di tecniche di intelligenza artificiale e apprendimento automatico che saranno, per completezza, introdotti nelle appendici presenti.

Capitolo 2

Elaborazione di immagini con camera fissa

*Io stimo più il trovar un vero, benché di cosa leggiera,
che 'l disputar lungamente delle massime questioni
senza conseguir verità nissuna.*

Galileo Galilei

In questo capitolo vengono illustrate alcune tecniche di visione artificiale utilizzate nel campo di ricerca dell'elaborazione di immagini provenienti da telecamera fissa. Le applicazioni presentate nel capitolo sono state impiegate come prodotti commerciali rendendo soluzioni pratiche a problemi semplici da comprendere, ma difficili da affrontare con sistemi automatici. L'attenzione si sposterà di fatto nell'ambito delle applicazioni di videosorveglianza, principali fruitrici delle tecniche legate all'utilizzo di telecamere fisse. Nel nostro caso la classe di sistemi interessati è riferita al controllo di accesso a varchi.

2.1 Controllo di accesso a varchi

Il controllo degli accessi ad un varco trova applicazioni commerciali legate sia alla videosorveglianza che all'analisi dei flussi di persone attraverso lo stesso per considerazioni dal punto di vista economico. Una prima applicazione legata all'approccio con telecamera fissa riguarda un controllo di accesso a varchi con conteggio delle persone che vi passano attraverso.

Il conteggio automatico delle persone è divenuto negli ultimi anni un problema molto sentito in diverse applicazioni commerciali. Offrire un'applicazione affidabile, efficiente e robusta è uno dei principali obiettivi degli sviluppatori. Tramite l'analisi delle statistiche ottenute, si possono identificare problemi nell'andamento del flusso di persone, avere un riscontro riguardo all'ottimizzazione dello stesso per gestire eventuali promozioni fornite da un negozio o un grande magazzino, dimensionare il varco stesso, monitorare gli accessi, controllare il pagamento delle tariffe e altro. In aggiunta la sorveglianza del varco e dell'area nei pressi dello stesso è uno degli obiettivi di un'applicazione di questo genere.

Le prime soluzioni, rivolte a risolvere il problema del conteggio, sono state l'utilizzo di contatori a contatto, per esempio tornelli. I dispositivi di questo tipo contano una persona per volta normalmente, sono fissati, occupano spazio, risultano invasivi e molte volte rallentano il normale flusso delle persone. Rallentare il passaggio, a volte potrebbe essere un desiderio dell'installatore, però si deve considerare che impediscono totalmente il passaggio o lo rendono molto difficoltoso a persone diversamente abili, con zaini, bagagli o passeggeri. In alcune installazioni possono essere scavalcati facilmente e spesso permettono una sola direzione di marcia.

Successivamente sono state proposte soluzioni mediante l'utilizzo di sensori, come ad esempio laser o coppie di laser. Queste soluzioni risultano meno invasive, perchè non hanno un'occupazione di spazio notevole, inoltre permettono di riconoscere la direzione del passaggio usando 2 diversi raggi laser opportunamente etichettati, ma non risolvono in modo adeguato il problema. La presenza di più persone affiancate, di un gruppo di persone oppure di qualcuno che fa oscillare una borsa comporta sempre un errore di conteggio.

	Pro	Contro
Camera	Immagini disponibili Sensori passivi Nessun ingombro Ambiente non strutturato	Elaborazione Sensibilità illuminazione o occlusione
Laser	Basso costo Semplicità di elaborazione	Installazione fisica Poca espandibilità Sensibilità occlusione
Tornelli	Facilità di elaborazione Conteggio corretto	Ingombro, installazione fisica Ambiente strutturato Immagini non disponibili

Tabella 2.1: Confronto tra i principali metodi per il controllo di accessi a varchi.

Per risolvere queste limitazioni è stata introdotta la visione artificiale. L'utilizzo di telecamere, poste in posizione zenitale, riduce la possibilità di occlusione tra le persone riuscendo ad ottenere prestazioni migliori, limitando l'errore di conteggio al minimo. La telecamera può anche svolgere la funzione di video sorveglianza, quindi lo stesso dispositivo può essere sfruttato per due diversi compiti. Avendo a disposizione le immagini si potrebbe anche effettuare l'identificazione delle persone, per tenere traccia dell'ora di entrata e di uscita e calcolare il tempo in cui rimangono all'interno del locale in cui è posta la telecamera. Per un negozio, avere queste informazioni unite a quelle ricavate dai registratori di cassa sugli orari delle vendite, è utile per stabilire una strategia di pianificazione delle offerte, in modo da trovare il modo e l'orario migliori per proporre specifiche promozioni. Nel caso in cui un varco sia legato al pagamento di una tariffa in questo caso si possono identificare eventuali trasgressori. A livello prettamente commerciale si nota una certa attenzione da parte di produttori di sistemi di videosorveglianza e molta richiesta. In tabella 2.1 si ha uno schema del confronto tra i principali metodi di controllo di accesso a varchi.

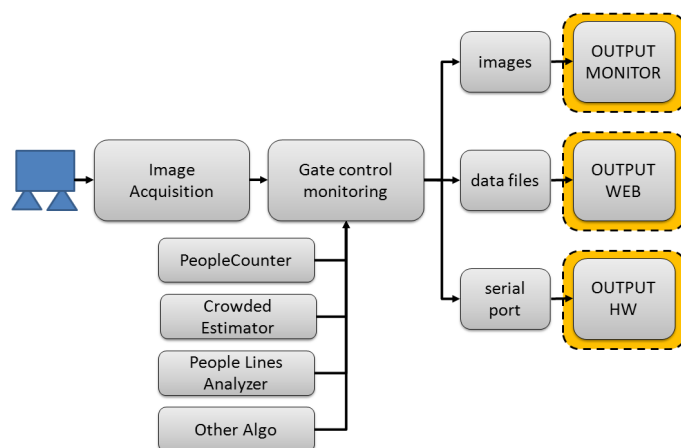


Figura 2.1: Struttura del sistema di controllo di un varco.



(a)

(b)

Figura 2.2: Prototipi di varco. (a) interno, (b) esterno.

2.1.1 Scopo del progetto, prototipazione e sperimentazione

Il progetto si propone di analizzare mediante tecniche di visione artificiale la scena in prossimità di un varco per ottenere un prodotto finalizzato alla commercializzazione. Per ottenere questo si è sviluppato un framework basato sullo schema presentato in figura 2.1 in cui è evidenziata la struttura modulare del progetto.

Le immagini acquisite da telecamera, nella figura stereo, vengono elaborate da un primo blocco di *image analysis*, dopodichè vengono inviate ad un blocco che contiene gli algoritmi a camera fissa che è possibile attivare per il monitoraggio del varco.

Tra gli algoritmi troviamo un PeopleCounter, per il conteggio delle persone, un Crowded Estimator, per la stima dell'affollamento della scena e un People Lines Analyzer per analizzare le code di persone in prossimità del passaggio.

I dati in uscita dai blocchi di controllo del varco possono essere visualizzati a video, come immagini, mostrati su un sito dedicato come statistiche tramite un browser, o immediatamente percepibili attraverso l'accensione di led posti sul varco stesso.

Per quanto riguarda la prototipazione e la sperimentazione del sistema e degli algoritmi sviluppati è stato necessario costruire un varco mobile in grado di essere spostato e trasportato facilmente e un varco fisso, esterno con illuminazione notturna. Entrambi i prototipi, mostrati in figura 2.2, sono stati utili per acquisire sequenze di immagini per lo sviluppo e il testing degli algoritmi off-line, ma anche per test in tempo reale sul campo e dimostrazioni. Il varco esterno presenta anche la possibilità di essere illuminato in modo da poter funzionare in modalità sia diurna che notturna.

2.2 Stato dell'arte controllo e conteggio degli accessi a varchi mediante telecamera

In questa sezione viene presentato un breve stato dell'arte per quanto riguarda i sistemi di conteggio di persone attraverso dei varchi realizzati con l'ausilio di tecniche di visione artificiale.

Principalmente esistono due differenti approcci per questa tipologia di applicazioni: mediante telecamera singola e mediante coppia stereo di telecamere. In en-

trambi i casi negli ultimi anni l'argomento è stato ampiamente dibattuto presentando però ancora molte possibilità di miglioramento ed innovazione.

In questo paragrafo verranno illustrati gli articoli scientifici che trattano il conteggio automatico di persone mediante telecamere. Si parte dal presupposto che la telecamera sia fissata in modo che possa riprendere la sagoma delle persone che entrano nella scena. Il problema del conteggio delle persone e dell'head detection viene spesso risolto con approcci a singola camera, con le problematiche che tale scelta presenta.

La necessità di un buon algoritmo di background subtraction che determini un trade-off ragionevole tra prestazioni e complessità computazionale spaziale e temporale, ma soprattutto che discrimini correttamente, senza generare troppi artifici, le parti relative al foreground e al background anche in casi in cui quest'ultimo si riveli soggetto a cambiamenti repentini.

La presenza di ombre proiettate dai soggetti presenti nella scena che possono estendere in maniera scorretta il blob dopo la fase di rimozione dello sfondo come affrontato in [4] è risolto mediante modellazione delle ombre e inficiare le prestazioni del sistema sviluppato. In questo lavoro viene presentato un sistema di rilevamento delle persone orientato alla videosorveglianza, in situazioni complesse. La telecamera è posizionata ad una quota molto alta per evitare occlusioni, l'applicazione è outdoor e con camera singola. Si trovano le persone mediante background subtraction e tramite una tecnica apposita si cerca di eliminare le ombre.

La gestione delle situazioni di occlusioni parziali dovute alla presenza di diverse persone nel caso di scene affollate: una possibile scelta per limitare al minimo tale fenomeno può essere quella di posizionare la telecamera in posizione privilegiata, e quindi ad una quota abbastanza alta, come è stato fatto in [4] e [5]; oltre a scenari di tipo "crowded" anche il semplice transito di due persone a distanza ravvicinata, o in un qualche modo "collegate" tramite, per esempio, un abbraccio, può dare luogo a blob che finiscono per inglobare più di una persona; anche in questo caso questo scenario va gestito correttamente ricorrendo a tecniche di segmentazione particolari come in [6] in cui viene utilizzato k-means per clusterizzare persone differenti, o affidandosi al tracking di features basati su filtri bayesiani, Kalman o filtri particellari.

Bisogna tener presente che per applicazioni semplici si posiziona la telecamera in modo zenitale per evitare occlusioni, come appunto in [6] e i [7]. In quest'ultimo lavoro la telecamera è posizionata sulle porte di un treno, per il controllo e conteggio degli accessi. Le persone vengono individuate con una tecnica morfologica.

Per quanto riguarda la realizzazione di un sistema per il controllo di varchi basato su telecamera stereo in letteratura sono presenti alcuni suggerimenti dati da articoli che presentano approcci interessanti relativi al riconoscimento di teste.

Nanda e Fujimura [8] presentano un metodo robusto per il tracking di teste in ambienti altamente affollati. Si sfrutta la mappa di profondità ottenuta dall'elaborazione tramite coppia stereo. La testa viene modellata mediante un'ellisse la cui posizione e fattori di forma vengono costantemente aggiornati tramite una ricerca locale. Il rilevamento di un'ellisse all'interno dell'immagine viene effettuato valutando le discontinuità attraverso la distanza di Chamfer, definita come un valore di somiglianza tra un'immagine e il modello. Il risultato è un sistema di tracking robusto invariante alle rotazioni ed ai cambiamenti di illuminazione. Questo metodo è interessante in quanto promette possibili generalizzazioni ad altre forme, per esempio mani.

In [9], van Oosterhout propone un sistema di head detection basato su immagini stereo. Dopo la fase di estrazione degli elementi nella scena (foreground) si cerca una corrispondenza tra il template del modello con l'immagine. Questa tecnica è utilizzata anche in ambienti affollati.

Un sistema stereo per il conteggio delle persone all'interno di centri commerciali, è descritto in [10]: viene effettuata una ricostruzione 3D della scena filtrando i punti che ricadono all'interno di un volume di interesse. Il progetto permette di identificare la parte del torso e della testa dei clienti ottenendo percentuali di affidabilità molto elevati.

Il tracking affidabile di persone tramite una coppia stereo a colori, è proposto da Krumm et al. in [11]. Questo articolo fa parte del progetto EasyLiving di Microsoft ed ha come obiettivo la realizzazione di un tracking da utilizzare in scene di vita quotidiana. L'applicazione è in grado di classificare diverse azioni come camminare, sedersi, correre anche in caso di occlusioni. Il progetto ha lo scopo di rendere l'ambiente domestico interattivo con l'utente in base alle azioni compiute.

In [12] è presentato un metodo per il tracking di persone all'interno di spazi affollati mediante il riconoscimento di teste. Le differenti tonalità di colore vengono sfruttate per separare le varie parti del corpo all'interno delle silhouette individuando il capo dalle altre parti del corpo rilevate. Il tracking degli individui si suddivide in classi temporali differenti: breve termine (l'utente rimane all'interno dell'area inquadrata), medio termine (l'utente esce e rientra nell'inquadratura nel giro di pochi minuti) e lungo termine (dove il soggetto può assentarsi dalla scena per ore o addirittura giorni). Il tracking a breve termine è effettuato attraverso l'analisi della posizione e dimensione di regioni di interesse, mentre gli inseguimenti a medio e lungo termine si basano su statistiche relative alla comparsa/scomparsa della persona all'interno della scena.

Infine in [13] Kelly et al. si discute un modello per il rilevamento ed inseguimento di pedoni all'interno di generiche scene affollate. Il rilevamento delle persone avviene attraverso un processo di clusterizzazione 3D mediante tecniche di region growing [14]; il calcolo delle soglie utilizzate durante questa fase deriva da vincoli di tipo biometrico. La fase di tracking è ottenuta attraverso l'utilizzo di un schema di associazione pesato in base a criteri di cardinalità massima.

Verranno ora approfonditi i pro e i contro di un approccio mediante coppia stereo. La scelta di utilizzare sistemi stereo permette di superare alcuni limiti che si hanno quando si utilizzano tecniche a singola telecamera:

- Il processo di separazione del foreground e background [15] può risultare meno dispendioso, garantendo risultati efficienti grazie allo sfruttamento delle informazioni fornite dalla mappa di profondità ricavabile dalla scena.
- I dati ottenuti tramite tecniche stereo sono invarianti ai cambi repentini di illuminazione e alla presenza di ombre. Le ombre che vengono proiettate sul terreno verranno eliminate con l'operazione di ground subtraction.
- Eventuali occlusioni tra le persone possono essere individuate e gestite in maniera efficiente ed esplicita.

- Consente la possibilità di ricavare nuove feature, caratteristiche, per risolvere problemi durante la fase di tracking grazie all'informazione di profondità dell'immagine.

Benchè la visione stereo presenti evidenti vantaggi, questo tipo di approccio non è esente da inconvenienti:

- Le informazioni di profondità ricavate possono essere affette da rumore che deve essere preventivamente soppresso (o tanto meno tenuto in considerazione) per non inficiare la qualità dei dati ricavati.
- Non è possibile ottenere l'informazione di profondità quando si hanno regioni con texture uniformi.

2.3 Contapersone con telecamera singola

Il sistema è stato realizzato mediante una singola telecamera in posizione zenitale sopra al varco da controllare. La telecamera è di tipo analogico con sensibilità nello spettro del visibile, a colori. Il sistema è basato su un algoritmo che fa uso delle comuni tecniche di background subtraction, anche conosciute come foreground extraction.

2.3.1 Panoramica dell'algoritmo

In figura 2.3 è illustrata la struttura dell'algoritmo del sistema di conteggio degli accessi a un varco con singola telecamera.

Il blocco *frame acquisition* si occupa di ottenere l'immagine al tempo corrente proveniente dalla telecamera, l'immagine così ottenuta viene fornita in ingresso al blocco responsabile della generazione e dell'aggiornamento del background.

Il blocco *background update* si occupa di generare l'immagine di background al frame corrente. La costruzione del background avviene con un metodo base di background subtraction, con alcune ottimizzazioni per alleggerire il carico computazionale per questo motivo si può considerare la tecnica utilizzata come una media

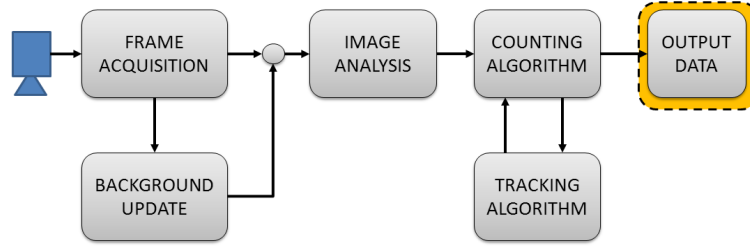


Figura 2.3: Struttura dell'algoritmo sistema di conteggio degli accessi a un varco con singola telecamera

temporale ottimizzata. Consideriamo la generazione di un'immagine di background tramite media temporale illustrata nell'equazione:

$$B_{i+1} = \left(\sum_{k=i-N}^i F_k \right) / N \quad (2.1)$$

Dove B_{i+1} è l'immagine di background al tempo $i + 1$, F_k sono i k frame precedenti con $k \in i - N, i$, quindi con N arbitrario che indica la coda di frame da considerare. L'operazione considera i pixel per ogni immagine all'interno della finestra temporale, mediando quindi nel tempo il valore di ogni singolo pixel del frame. Lo svantaggio di questo algoritmo è che richiede una quantità di memoria $N \cdot \text{size}(\text{frame})$ e tempo di calcolo proporzionale al numero di frame e alla loro dimensione in pixel.

Per rimediare agli svantaggi della soluzione appena descritta si può migliorare il calcolo del background attuale, a regime, quando si ha la coda piena, in questo modo:

$$B_{i+1} = (B_{i-1} \cdot N - F_{i-N} + F_i) / N \quad (2.2)$$

dove F_{i-N} è l'ultima immagine considerata dalla finestra mobile nel calcolare la media.

In questo modo si abbatta drasticamente (di ordini di grandezza) il tempo di esecuzione della funzione di calcolo del background mantenendo i vantaggi che la tecnica della media temporale porta. In figura 2.4.(b) è illustrato il risultato di questo metodo.

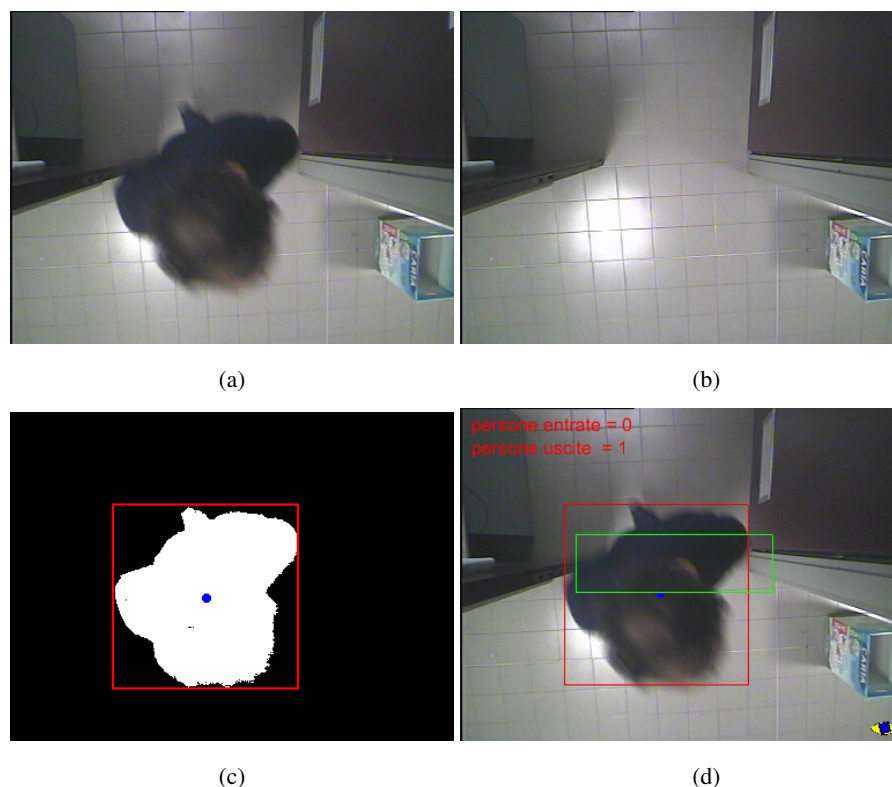


Figura 2.4: Immagini legate ai passi dell'algoritmo: (a) input, (b) background (c) differenza con estrazione del blob, (d) output.

Il blocco di *image analysis* riceve in input l'immagine differenza, che contiene il foreground della scena analizzata. All'interno di questo blocco vengono effettuate le operazioni necessarie per l'estrazione dei blob che rappresentano gli oggetti in movimento della scena inquadrata. Per ogni blob individuato tramite una procedura di etichettatura con una tecnica di tipo *flood fill* vengono estratte le informazioni relative al centro di massa, all'area, e al *bounding box*.

Nel blocco *counting algorithm* vengono effettuate le operazioni di conteggio. In input al blocco vengono forniti la lista dei blob presenti nella scena con le associa-

zioni storiche indicate dal blocco di *tracking algorithm*. Il conteggio in entrata o in uscita viene effettuato tramite il confronto del centro di massa di ogni blob presente nella scena con gli estremi di una regione di attraversamento modificabile lato utente. La regione di attraversamento è rappresentata da due linee per rafforzare il conteggio nel caso in cui una persona permanga a cavallo della linea di attraversamento incrementando i conteggi in entrambe le direzioni.

Il blocco di *tracking algorithm* associa i blob a frame differenti, considerando le posizioni dei loro centri di massa e le aree per l'associazione. La previsione della posizione avviene tramite un filtro di Kalman.

Il blocco di *output data* provvede a trasmettere in output le immagini e i dati del sistema.

Nella figura 2.4 sono riassunte le elaborazioni dei blocchi principali dell'algoritmo sulle immagini acquisite dalla telecamera. In 2.4.(a) è illustrata l'immagine acquisita al frame corrente, in 2.4.(b) è mostrato il background attuale. Mentre in 2.4.(c) è illustrata l'immagine differenza calcolata a partire dalle immagini del frame attuale e del background con in seguito una sogliatura. In questa immagine vengono evidenziati il blob etichettato, in bianco, il proprio bounding box, in rosso, e il centro di massa, in blu. Nella figura 2.4.(d) viene mostrato l'output finale dell'algoritmo con il conteggio delle persone passate attraverso il varco, l'area di passaggio e la parte dell'immagine corrispondente al blob con bounding box e centro di massa evidenziati.

2.3.2 Classificazione

Si verificano alcuni casi problematici in cui la semplice analisi basata su background extraction non risulta robusta, comportando un notevole aumento delle miss detection. Queste situazioni sono legate alla presenza di più persone che attraversano il varco, ad esempio due persone affiancate o due persone che si incrociano in direzioni opposte. Esse sono state affrontate utilizzando delle tecniche di intelligenza artificiale, utilizzando dei classificatori basati su alberi di decisione. Come caratteristiche salienti su cui addestrare i classificatori sono state escluse la semplice area del blob poichè avendo un'area inquadrata molto ristretta non è affatto semplice distinguere solamen-

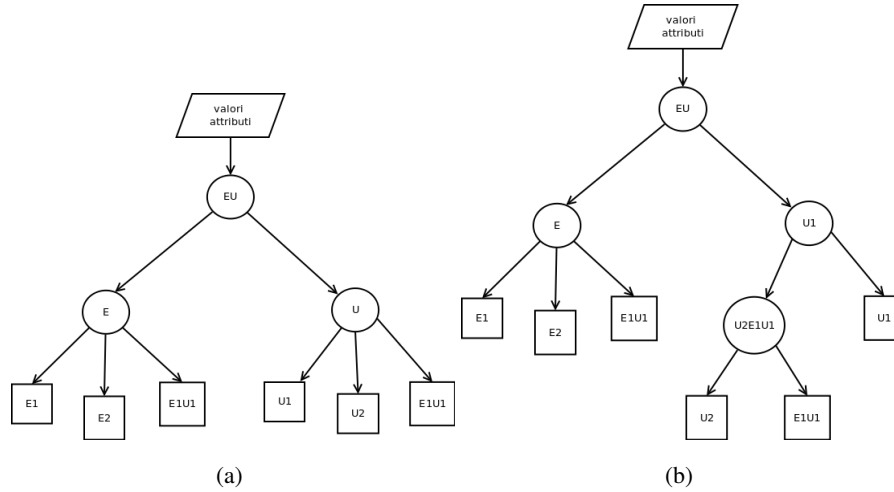


Figura 2.5: Albero di classificatori conta persone con telecamera singola: (a) caso con 1 blob, (b) caso con 2 blob

te con questa informazione tra una o più persone. Sono state quindi considerate come feature l'area del blob unita all'inclinazione del blob, ottenuta a partire dal calcolo degli assi d'inerzia, e il rapporto tra gli autovalori della matrice d'inerzia.

Le classi utilizzate dagli alberi di decisione per la classificazione sono visualizzate all'interno degli alberi mostrati in figura 2.5

2.3.3 Risultati ottenuti

In questa sezione verranno mostrati i risultati ottenuti dal sistema di conteggio degli accessi presso il varco con singola telecamera fissa zenitale. In un primo momento saranno analizzate le prestazioni dal punto di vista del funzionamento del sistema, in seguito saranno analizzati brevemente i tempi di calcolo confrontando l'esecuzione degli algoritmi su due sistemi differenti, infine verranno mostrate alcune immagini in cui vengono evidenziati dei passaggi complessi. Nella tabella 2.2 sono riassunti i risultati ottenuti dal sistema di controllo di accessi presso un varco mediante una telecamera singola a colori. Le correct detection sono state calcolate mediante confronto con ground thruth sulle sequenze annotate da un operatore umano. In totale hanno

Classe	Totale eventi	Senza AI	Singolo Classificatore	Albero di Classificatori
singola entrata	77	100	85.7	83.1
singola uscita	84	100	88.1	95.2
entrata doppia	46	0	67.4	82.6
uscita doppia	44	0	59.1	81.8
incrocio	25	0	92	76
Totale	276	58.33	79.71	85.85

Tabella 2.2: Risultati ottenuti per il sistema di accesso ai varchi con telecamera singola, percentuali di correct detection.

attraversato il varco 391 persone, delle quali in entrata 194 e in uscita 197. Ogni riga della tabella è associata ad un differente evento presente nella scena. Questa suddivisione è utile per ragionare sulle prestazioni del sistema considerando le tecniche di intelligenza artificiale inserite per affrontare le situazioni problematiche.

Si nota come nel caso di singola entrata e di singola uscita il sistema senza l'ausilio di tecniche di intelligenza artificiale riesca a funzionare con un'alta affidabilità; mentre nel caso in cui la situazione assuma delle caratteristiche più complesse sbagli clamorosamente.

L'utilizzo di un singolo classificatore o di un albero di classificatori permette di affrontare le situazioni complesse peggiorando di poco le situazioni semplici. In tutti i casi le prestazioni rimangono buone. Per quanto riguarda l'utilizzo dei classificatori si nota come l'utilizzo delle due differenti tipologie abbia un funzionamento parzialmente migliorativo sull'intero sistema. Una tipologia di classificatore migliora le prestazioni solamente su alcune classi e le peggiora su altre. Si può pensare di introdurre una tipologia di classificatore o eliminarla in base all'ambiente che si vuole monitorare, e al contesto, considerando gli eventi che con maggiore probabilità avverranno.

Per quanto riguarda i tempi di calcolo il sistema è stato testato su due differenti calcolatori:

- processore Intel Core2 Duo P8600 a 2.4 GHz, 4 GB memoria RAM;
- processore Intel Pentium M 740 a 1.73 GHz, 2 GB memoria RAM.

I tempi ottenuti per l'elaborazione complessiva dell'algoritmo sono stati rispettivamente di 26 ms e 49 ms.

Verranno analizzate tre situazioni di passaggio attraverso il varco che mostrano come nel caso in cui si abbiano incroci o passaggi multipli si possano avere errori o rilevazioni corrette. Nell'esempio in figura 2.6 è mostrato il caso di un incrocio nei pressi del varco con conteggio errato, la persona sulla destra presenta un abbigliamento con colore troppo simile al background, quindi il sistema non lo riconosce fallendo il conteggio. Nell'esempio in figura 2.7 stanno entrando due persone affiancate: in questo caso il blob è abbastanza grande perchè l'evento venga classificato correttamente.. Nell'esempio in figura 2.8 si ha un incrocio sulla soglia del varco, per effetto dell'etichettatura i blob vengono fusi in uno unico, in questo caso il contapersone senza l'utilizzo di classificatori fallirebbe, al contrario grazie all'utilizzo degli stessi si ha un conteggio corretto.

2.4 Contapersone stereo

Come confutato dai risultati ottenuti, utilizzando un controllo degli accessi basato su telecamera singola e con un approccio di tipo background subtraction che possa avere prestazioni in tempo reale soddisfacenti si hanno risultati soddisfacenti nel caso di singole persone che attraversano, ma nel caso di affollamento o nel caso di passaggi multipli o incroci non è semplice trovare un sistema affidabile e soprattutto scalabile.

Per migliorare le prestazioni, incrementando i risultati in termini di rilevazioni corrette nel caso di luoghi affollati si è sviluppato un sistema di controllo degli accessi a varchi con telecamera stereo.

2.4.1 Panoramica dell'algoritmo

La figura 2.4.1 illustra la struttura dell'algoritmo del sistema di conteggio degli accessi a un varco con telecamera stereo. Il sistema presenta un algoritmo abbastanza

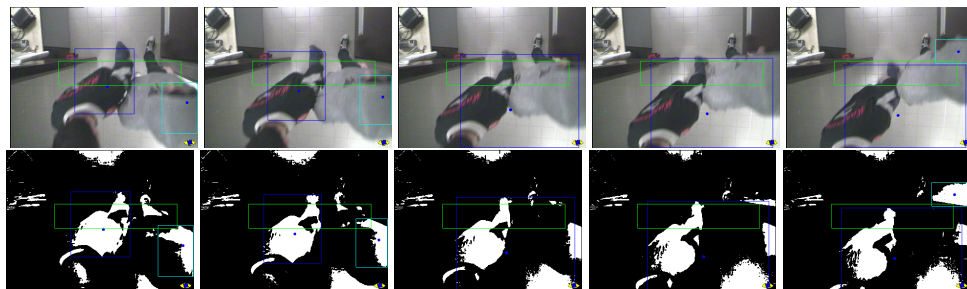


Figura 2.6: Esempio di conteggio errato con incrocio

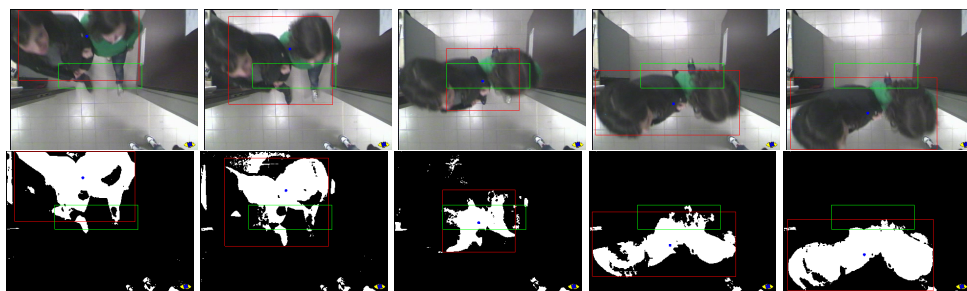


Figura 2.7: Esempio di conteggio corretto con passaggio di due persone

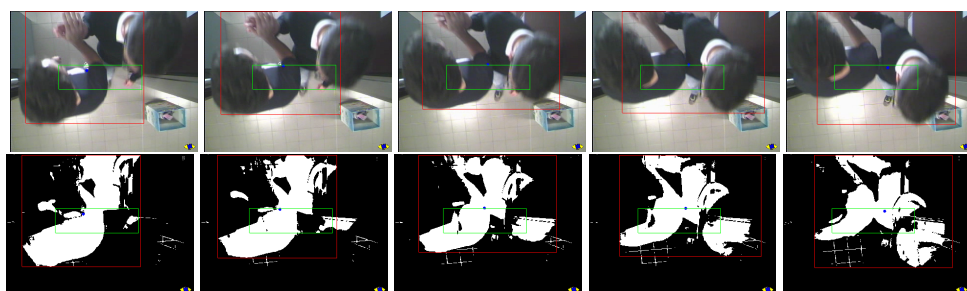


Figura 2.8: Esempio di conteggio corretto con incrocio

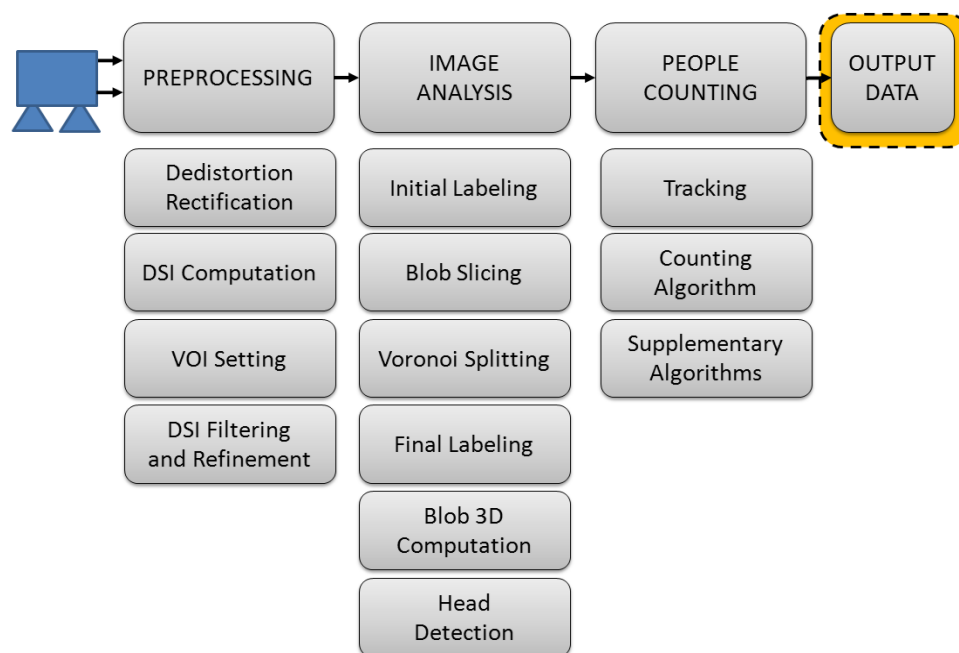


Figura 2.9: Struttura dell'algoritmo del sistema di conteggio degli accessi a un varco con camera stereo

complesso, composto da tre stadi e molti blocchi, è costruito per essere modulare, in modo da poter inserire, togliere o inibire alcune parti mantenendo le caratteristiche base del progetto. Nelle sezioni successive verranno analizzati nel dettaglio i blocchi dell'algoritmo.

2.4.2 Preprocessing

Il primo stadio del sistema riceve le due immagini dalla telecamera stereo sincronizzate e applica alcuni algoritmi di elaborazione necessari per le elaborazioni negli stadi successivi.

Distortion removal and Rectification

Questo blocco agisce sulle immagini per rimuovere la distorsione introdotta dalla lente. Nel caso delle telecamere utilizzate per la sperimentazione si è utilizzato uno strumento software interno al laboratorio che ha permesso di generare un file che permettesse di poter applicare ad ogni immagine acquisita una trasformazione per fare in modo che la distorsione fosse corretta con il minor errore possibile. Le immagini vengono poi rettificate per fare in modo che si possano utilizzare al blocco che esegue il calcolo della disparità.

DSI Computation

Il blocco fornisce in uscita un'immagine di disparità confrontando le immagini di input alla testa stereo rettificate e dedistorte. L'immagine di disparità (DSI, disparity space image) è un'immagine in cui i pixel assumono un colore con valore dal rosso al viola in base alla distanza che hanno dall'osservatore. I valori calcolati a questo passo sono fondamentali per la ricostruzione in tre dimensioni della scena osservata dalla telecamera.

VOI Setting

Non tutta la scena inquadrata è di interesse per cui mediante un volume poligonale tridimensionale si può scegliere quale porzione individuare come zona di interesse. I volumi possono essere molteplici e modificabili anche a tempo di esecuzione. Si può definire per ogni volume una zona di attraversamento che diventerà fondamentale in fase di conteggio degli accessi.

DSI Filtering and Refinement

Il blocco finale del primo stadio, utilizzando tutte le informazioni dei primi tre blocchi prepara i dati per il secondo stadio. All'interno di questo blocco si ha un filtraggio della DSI considerando solamente i punti che ricadono all'interno dei volumi di interesse indicati. Questo avviene per filtrare la parte che non interessa e risparmiare

onerosi calcoli ai blocchi seguenti. La seconda parte di questo blocco si occupa di raffinare l'immagine DSI filtrata per coprire alcuni buchi al suo interno mediante una media nello spazio della disparità. Non è tra gli scopi di questa tesi l'implementazione di un algoritmo per il calcolo della disparità, per ottenere l'immagine DSI si è fatto riferimento a quanto realizzato in [16] e [17].

La figura 2.10 illustra tramite le immagini le elaborazioni effettuate dai blocchi dello stadio di *Preprocessing*. In figura 2.10.(a) e 2.10.(b) sono mostrate le immagini provenienti dalla coppia stereo di telecamere. Si può notare come l'ottica sia molto distorta e le immagini non siano tra loro allineate. In figura 2.10.(c) e 2.10.(d) sono mostrate le immagini in uscita dal blocco di *Distortion Removal and Rectification*, si nota come queste siano correttamente dedistorte e come fra di loro siano rettifiche. La figura 2.10.(e) mostra l'immagine di disparità ottenuta grazie alle immagini delle telecamere destra e sinistra opportunamente analizzate. Infine in figura 2.10.(f) è mostrata la DSI filtrata all'interno del volume di interesse, questa è stata anche opportunamente elaborata per fare in modo di chiudere alcuni piccole mancanze di informazione utilizzando le informazioni dei pixel adiacenti per generare le informazioni mancanti.

2.4.3 Image Analysis

Il secondo stadio del sistema è composto dagli algoritmi di analisi dell'immagine. Verranno elaborate le informazioni derivanti dalla mappa di disparità, allo scopo di identificare correttamente le persone presenti nella scena distinguendole da altri oggetti.

Initial labeling

In questo blocco si effettua un'etichettatura preliminare (*Initial Labeling*) delle componenti connesse presenti nella mappa di disparità, considerando la stessa proiettata sul piano XY , per semplificazione computazionale, estraendo dall'immagine proiettata in due dimensioni i blob presenti. Durante questa fase si considerano connesse due regioni se queste condividono almeno un pixel. La labelizzazione viene effettuata

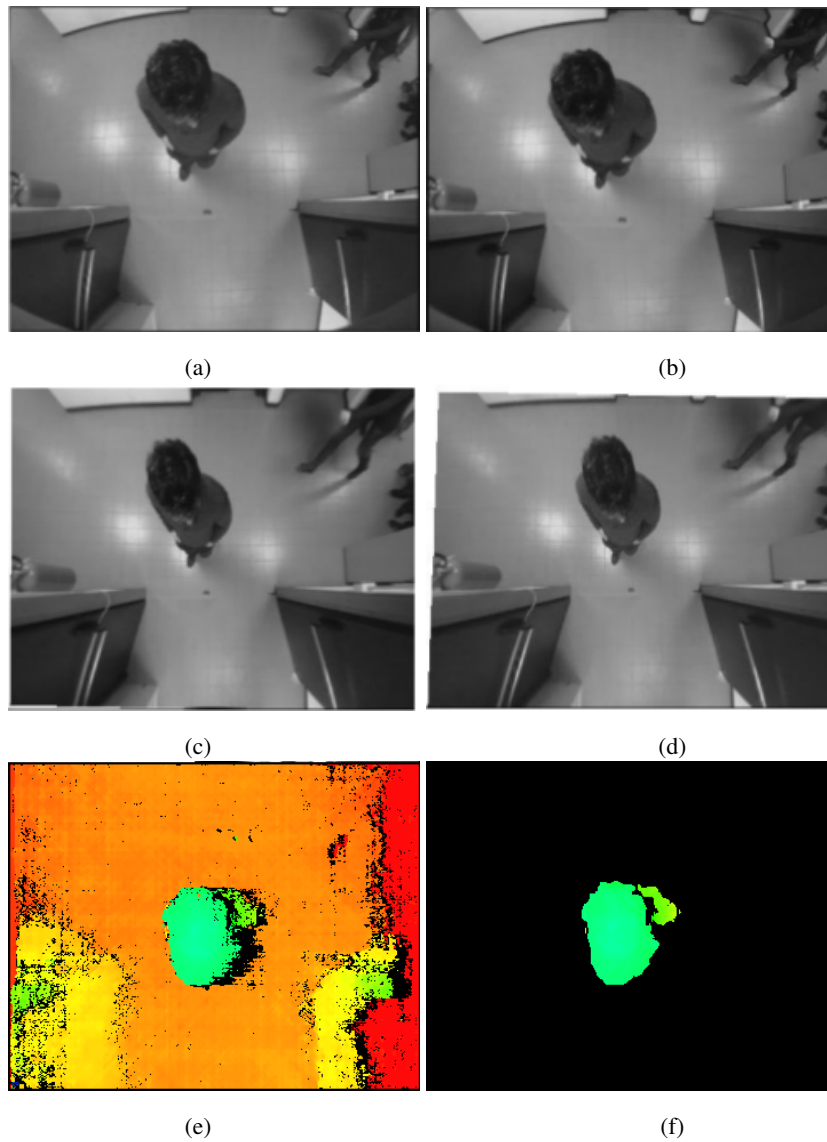


Figura 2.10: Stadio di Preprocessing: (a) immagine di input sinistra, (b) immagine di input destra (c) immagine dedistorta e rettificata sinistra e (d) destra, (e) immagine DSI (f) immagine DSI filtrata e raffinata.

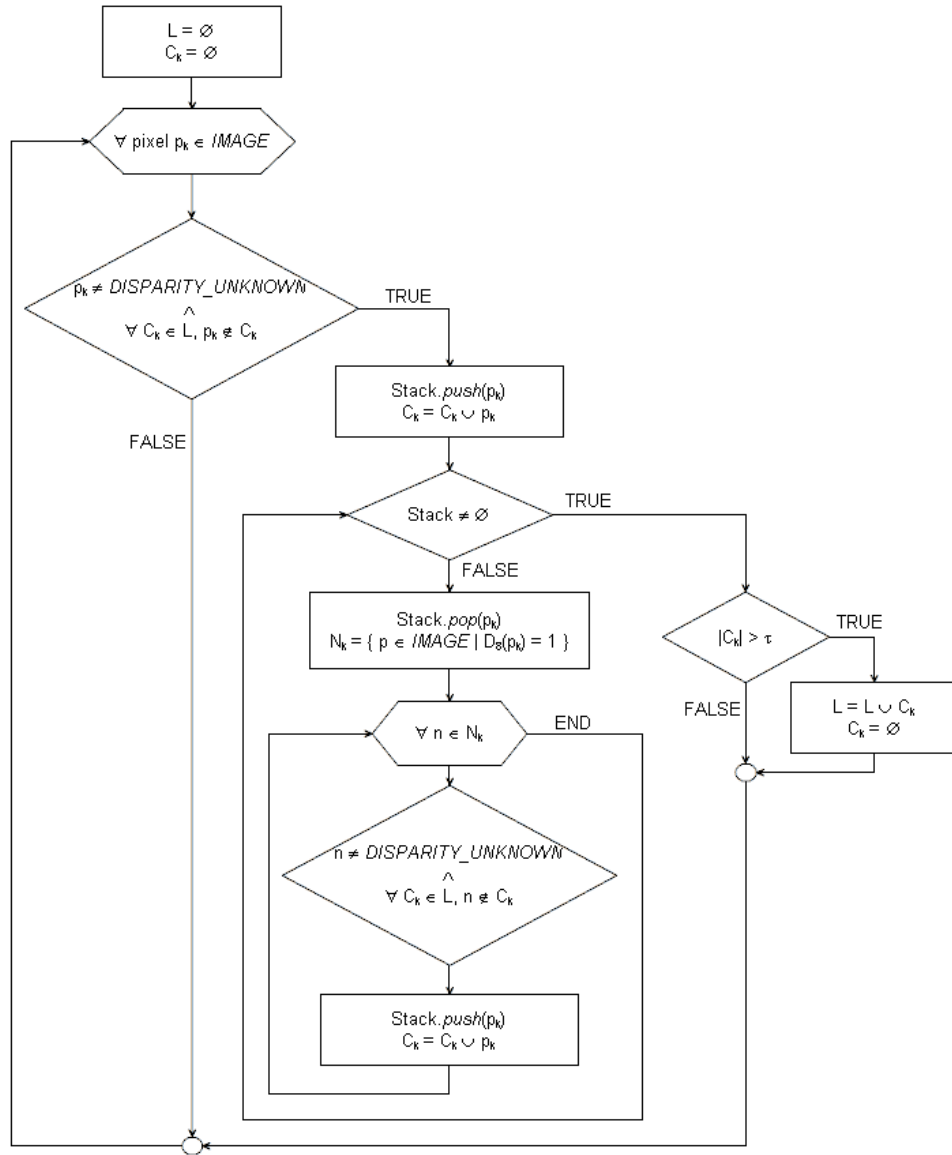


Figura 2.11: Diagramma di flusso dell'algoritmo di labeling utilizzato

mediante una tecnica di espansione *flood fill* in figura 2.11 è mostrato il diagramma di flusso dell'algoritmo.

Durante il rilevamento delle componenti connesse vengono calcolate le proprietà relative ad ogni blob come la posizione dei centri di massa e i limiti del bounding boxes. Introducendo una soglia (ricavata sperimentalmente) sulla dimensione minima dei blob, si riesce a eliminare quelle componenti più piccole, rumorose, che possono disturbare l'elaborazione.

Dall'immagine in figura 2.12.(a) è possibile notare come l'operazione di labelizzazione si comporti al meglio quando le persone che attraversano il varco risultano significativamente separate. Quando però transitano gruppi di persone affiancate, figura 2.12.(b) e 2.12.(c), il sistema li unisce in un blob unico. È necessario quindi un metodo per separare correttamente gli individui all'interno della scena.

Blob slicing

Questo blocco permette di avere una separazione dei blob uniti nella prima fase di etichettatura sfruttando le informazioni date dalla mappa di disparità per quanto riguarda l'altezza degli ostacoli rilevati all'interno del volume di interesse. La tecnica presentata consiste nel sezionare a diverse altezze le componenti connesse. Per ogni blob vengono calcolate l'altezza minima e massima. La differenza tra il valore massimo e minimo viene suddiviso in N differenti valori di quota. Partendo dal valore minimo a salire verso l'alto viene effettuato un taglio proiettando i punti della DSI con valore minimo pari alla quota indicata sul piano XY e invocando la funzione di etichettatura vista in precedenza. Al termine di ogni taglio il risultato della labelizzazione restituirà il numero di componenti connesse ottenute. Alla fine di tutti gli N tagli il blob originario verrà sostituito con l'insieme di blob a cardinalità maggiore ottenuto dal taglio a quota minima.

L'effetto che la fase di slicing ha sui blob ricavati è rappresentato nelle immagini della figura 2.13. Questa tecnica di troverà prima o poi una quota di taglio in grado di separare le persone all'interno di essa, per esempio all'altezza della testa o delle spalle nel caso di due persone che passano abbracciate tra di loro. Se, alla fine dell'o-

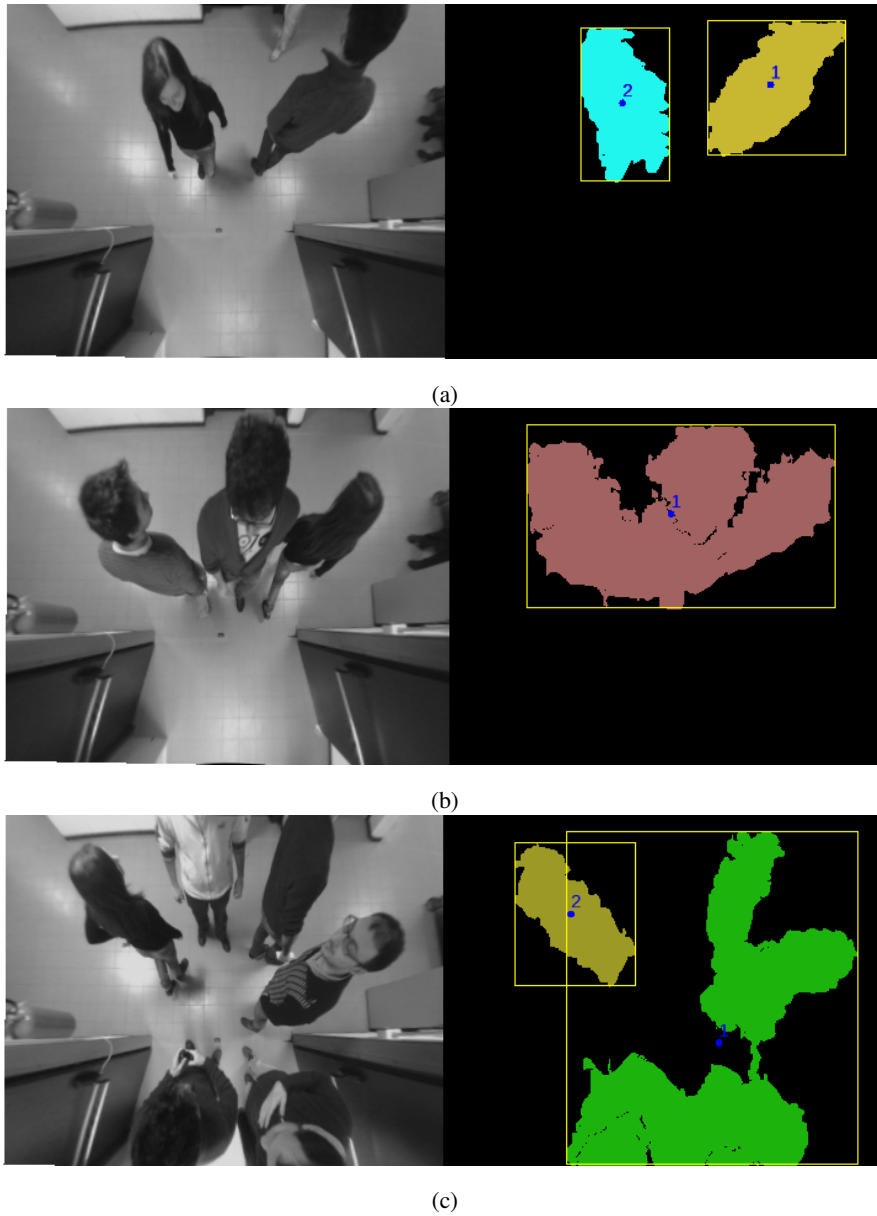


Figura 2.12: Fase di etichettatura a sinistra l'immagine in input e a destra l'output della fase di etichettatura: (a) due persone disgiunte, (b) scena con tre persone adiacenti, (c) scenario affollato.

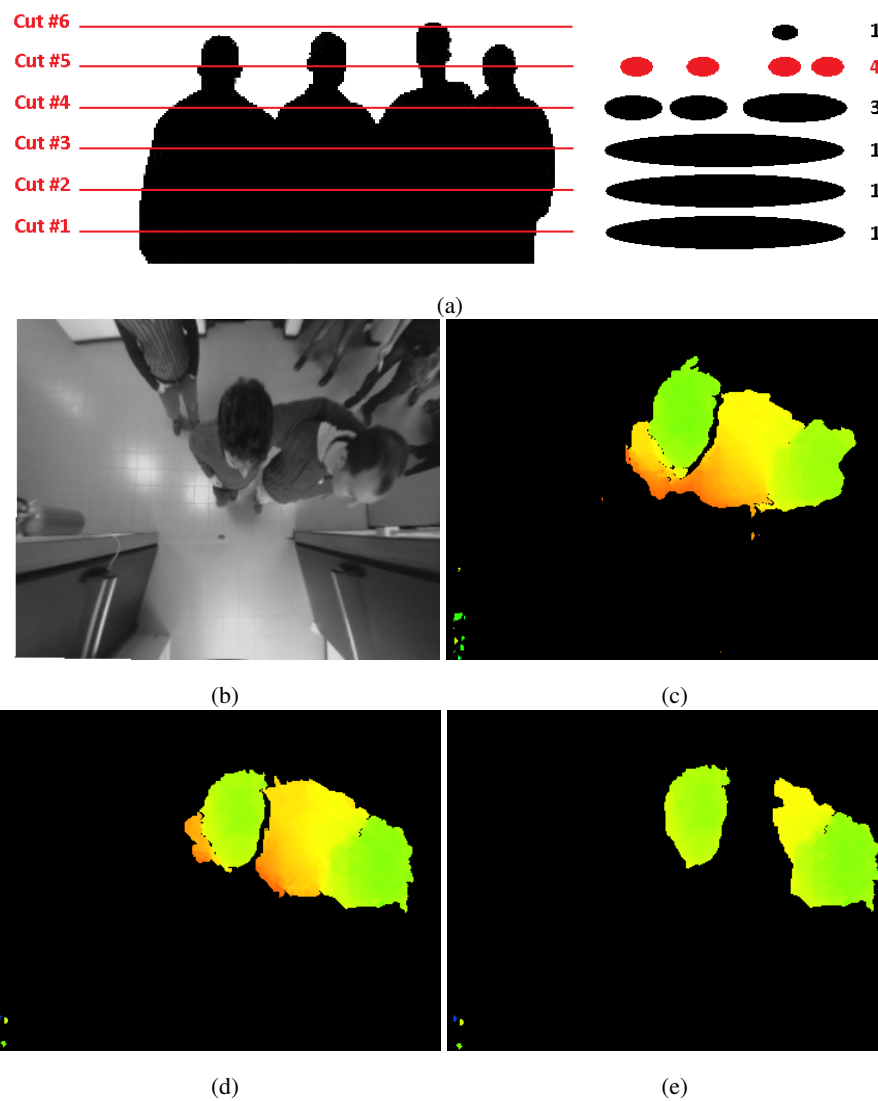


Figura 2.13: Blob slicing: (a) rappresentazione intuitiva dell'operazione di slicing, (b) immagine originale, (c) (d) (e) DSI con tagli a varie quote.

perazione di separazione, rimane un solo blob siamo sicuri che all'interno della scena è presente un'unica persona.

La tecnica di slicing presenta però un problema di perdita d'informazione dato dal fatto che si considerano via via quote superiori e quindi proiezioni sul piano XY che eliminano i punti sottostanti.

Voronoi splitting

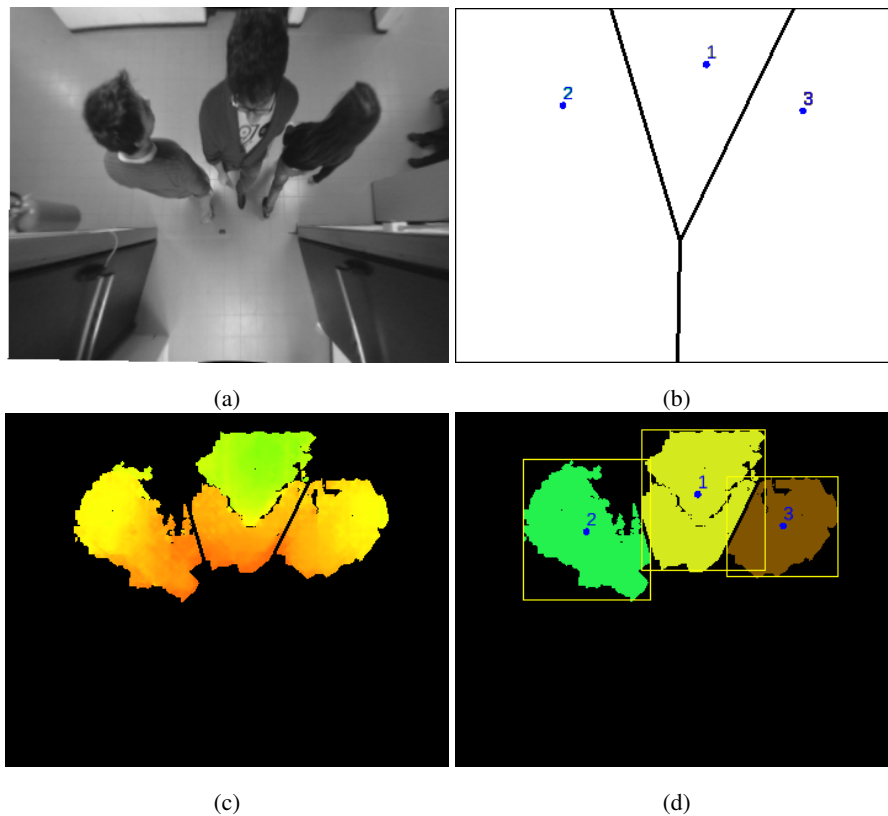


Figura 2.14: Voronoi Splitting: (a) immagine di input, (b) maschera di Voronoi, (c) DSI tagliata dalla maschera, (d) etichettatura finale.

Per ovviare al problema di perdita di informazione emerso nella fase di *slicing* si è introdotto un metodo per la separazione dei blob connessi che utilizza le informazioni sui centri di massa dei blob generati al passo precedente, ma con la mappa di disparità fornita in ingresso al secondo stadio, quindi quella iniziale. La tecnica di Voronoi Splitting, permette di conservare l'informazione fornita dall'immagine di disparità ma contemporaneamente separare le regioni connesse. Per raggiungere questo scopo si effettua una partizione della DSI mediante tassellatura di Voronoi. I punti dello spazio metrico vengono divisi in base alla distanza che hanno da un insieme di punti di controllo. Se si considera un piano, dato un insieme finito di punti S , il diagramma di Voronoi per S è la partizione del piano che associa una regione $V(p)$ ad ogni punto $p \in S$ in modo tale che tutti i punti di $V(p)$ siano più vicini a p che ad ogni altro punto in S . Nel nostro caso, i punti di controllo della tassellazione di Voronoi, sono i centri di massa dei blob, ottenuti dalla fase di slicing. Per poter realizzare la tassellazione in maniera efficiente, viene generata una maschera (delle stesse dimensioni della DSI) che possa essere messa in relazione logica con l'immagine di disparità ottenuta in La tassellazione di Voronoi rappresenta una soluzione efficace al problema del partizionamento della DSI. Inoltre è una tecnica di separazione efficiente, in quanto effettuabile in $O(n \log n)$. Nella figura 2.14 vengono illustrate le immagini relative alle operazioni effettuate nel blocco, in 2.14.(a) è presentata l'immagine di input; mentre in 2.14.(b) e 2.14.(c) vengono mostrate le immagini della maschera di Voronoi e della DSI dopo l'operazione logica rispettivamente, infine in 2.14.(d) è mostrato l'output dell'ultima fase di etichettatura.

Final Labeling

Separati i blob della DSI, si effettua un'ulteriore fase di labeling per ricavare le componenti connesse. Grazie alla tassellazione di Voronoi le componenti connesse della DSI sono separate per cui il labelizzatore assegnerà etichette differenti, senza la perdita di alcuna informazione. La figura 2.14.(d) mostra l'effetto di questo blocco.

Blob 3D Computation

Per ogni regione etichettata, vengono effettuate le seguenti operazioni:

- Calcolo dei punti 3D della componente connessa: grazie a questa operazione siamo in grado di ricostruire la silhouette tridimensionale del blob attraverso l'ottenimento delle coordinate 3D del mondo.
- Calcolo dei limiti 3D: dopo aver ricavato la lista dei punti nelle coordinate mondo, vengono estratte i valori minimi e massimi della silhouette nello spazio, ottenendo il volume occupato dal blob. Queste informazioni saranno utili all'Head Detection per discriminare le persone da altri oggetti presenti nella scena.

Head Detection

L'Individuazione delle teste (head detection) permette di risolvere il problema legato al soprannumero di regioni connesse create in fase di slicing. L'obiettivo finale dell'applicazione sarà quella di inseguire, mediante tracking, i centri di massa dei blob della scena per poter prevedere la loro traiettoria. Per evitare di commettere errori di inseguimento si deve eliminare l'eccessiva segmentazione di un blob quando contiene un'unica persona (e quindi non necessita di ulteriore suddivisione). Un'osservazione che si può fare è che nonostante il blob possa essere erroneamente suddiviso in troppe parti, sicuramente solo una di queste conterrà la testa della persona. Di conseguenza è necessario rendere disponibili per la fase di tracking solamente quelle componenti che presentano al loro interno la presenza del capo, mentre le altre verranno ignorate. Il modulo di head detection che verrà presentato è stato appositamente sviluppato per assolvere questo compito. Una tecnica comune per il rilevamento di teste, si basa sul riconoscimento della caratteristica "forma ad omega", Ω shape, individuabile nella zona testa-spalle di ogni persona. Tuttavia l'approccio che si è pensato di seguire ricerca una determinata configurazione spaziale dei punti mondo, all'interno delle silhouette 3D. Una volta a disposizione la lista dei punti mondo, e i relativi limiti spaziali, è possibile ricercare la testa attraverso l'utilizzo di un modello specifico.

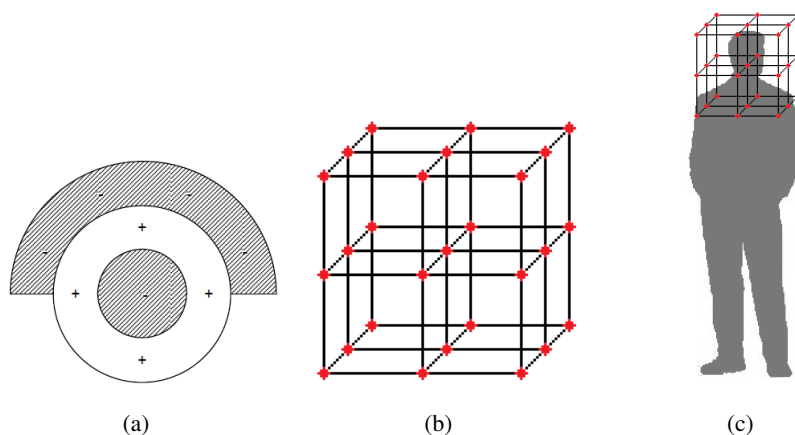


Figura 2.15: Head Detection: (a) modello per la ricerca della testa (b) griglia tridimensionale (c) posizionamento della griglia per il matching del template.

Template matching

Lo spaccato del modello utilizzato per la fase di matching è indicato in figura 2.15.(a), questo modello consiste in due bound sferici concentrici. La zona evidenziata dai simboli +, definisce la minima e la massima dimensione delle teste entro cui possono essere rilevate. Entrambi hanno valori plausibili ricavati da considerazioni antropometriche. Dato che l'altezza della testa è di circa $1/8$ dell'altezza totale di una persona si è pensato di utilizzare 0.16 m come valore per il raggio minore, corrispondente alla dimensione della testa di una persona alta circa 1.30 m ($0.168 = 1.28\text{ m}$). Il valore scelto per il raggio maggiore è di 0.24 m, corrispondente ad una persona alta circa 1.90 m ($0.248 = 1.92\text{ m}$). Per rafforzare il requisito di sfericità del modello si arricchisce questo schema con altre due regioni, denotate dal simbolo "-". La regione più interna esclude nuvole di punti piani o sparpagliate, mentre la seconda zona esterna, enfatizza il vincolo che questa composizione sferica di punti non debba avere connessioni nella parte superiore (eccetto le spalle), ma può essere connessa inferiormente. Solo le nuvole di punti disposti della testa combaceranno al meglio con questo modello.

Spatial Search

La regione di ricerca non è rappresentata da tutto il volume occupato dal blob ma si estende da una quota di circa $3/4$ dell'altezza del blob, fino al punto ad altezza massima. Successivamente questo volume viene discretizzato in R^3 vertici. Ogni vertice rappresenta il punto di centrale utilizzato per ricercare la testa mediante il modello precedentemente descritto (figura 2.15.(b) e 2.15.(c)). Si è scelto $R = 5$. Ogni volume sarà ispezionato dal template sferico in 125 locazioni differenti (5^3). La risoluzione scelta garantisce buoni risultati in termini di precisione.

Candidate Selection

Per ogni vertice della griglia si calcola il template il cui punto centrale coincide col vertice medesimo. Per valutare se effettivamente questi punti rappresentano una testa è necessario soddisfare determinate proprietà. Denominati con P l'insieme di punti compresi nella regione positiva "+" e con N l'insieme di punti compresi nelle regioni "-", possiamo definire i seguenti vincoli:

- Il numero di punti in P deve essere sufficiente, per cui deve essere superiore una determinata soglia.
- Il rumore dal quale è affetta la DSI può causare la caduta di punti in P all'interno di regioni "negative" e analogamente, punti appartenenti ad N possono cadere in regioni "positive". Si impone il vincolo che il numero di punti P diviso il numero di punti totali sia maggiore ad un certo valore ricavato sperimentalmente. In questo modo il candidato che non rispecchia la disposizione indicata dal template, sarà scartato.
- Infine vengono fatte considerazioni sulla distribuzione dei punti all'interno della crosta sferica. Viene ricavato il centro di massa dei punti dell'insieme P , e si controlla che il centroide giaccia al di sopra del punto centrale corrente del modello. Così facendo si privilegiano i punti mondo, del blob, in corrispondenza di teste e si escludono altri possibili candidati in posizioni peggiori o scorrette.

te (per esempio distribuzioni di punti concave o nuvole di punti in prossimità delle spalle).

Per chiarezza possiamo descrivere quanto appena detto in maniera più rigorosa, definiamo:

- I raggi minimo e massimo della regione sferica: $innerRay$, $outerRay$
- Il punto di centro corrente sul quale è posizionato il template: $C(c_x, c_y, c_z)R^3$
- Gli insiemi dei punti appartenenti alla regioni positiva e negativa come:

$$P = p \equiv (p_x, p_y, p_z) \in R^3 | D^2(p, C) \leq outerRay^2 \cap D^2(p, C) \geq innerRay^2$$

$$\begin{aligned} N &= N_1 \cup N_2 \longrightarrow \\ \begin{cases} N_1 = \{n \equiv (n_x, n_y, n_z) \in R^3 | D^2(n, c) < innerRay^2\} \\ N_2 = \{n \equiv (n_x, n_y, n_z) \in R^3 | D^2(n, c) < outerRay^2 \cap n_z \geq c_z\} \end{cases} \end{aligned}$$

Dove la relazione $D^2(a, b)$ indica la distanza euclidea quadratica, per una coppia di punti $a, b \in R^3$ è finita come segue:

$$D^2(a, b) = (a_x b_x)^2 + (a_y b_y)^2 + (a_z b_z)^2 \quad (2.3)$$

Il motivo per cui è stata scelta la forma quadratica rispetto a quella classica, è perchè la prima risulta più leggera in termini computazionali, poichè non prevede il calcolo della radice quadrata che è molto dispendioso. Per ottimizzare ulteriormente i calcoli vengono calcolati gli elevamenti a potenza semplicemente come prodotti ripetuti dello stesso termine. La formula utilizzata è la seguente:

$$D^2(a, b) = (a_x b_x) * (a_x b_x) + (a_y b_y) * (a_y b_y) + (a_z b_z) * (a_z b_z) \quad (2.4)$$

I vincoli elencati in precedenza si traducono quindi nelle seguenti relazioni:

1. $|P| \geq \tau_1$, dove $\tau_1 \in \mathbb{N}^+$ ed è una soglia scelta arbitrariamente che rappresenta il numero medio di punti di una testa, ricavata sperimentalmente.

2. $\rho = \frac{|P|}{|P|+|N|} \geq \tau_2 \Leftrightarrow |P| + |N| \neq 0$, dove $\tau_2 \in [0, 1]$, e ρ rappresenta il valore reale di verosimiglianza di una testa. Sperimentalmente è stato ricavato il valore di $\rho = 0.7$
3. $\gamma_z \geq \Gamma \equiv (\gamma_x, \gamma_y, \gamma_z) \in R^3$, dove il punto Γ rappresenta il centroide dei punti di P la quale i -esima coordinata è definita come $\gamma_i = \frac{1}{|P|} \sum_{p \in P} p_i \Leftrightarrow |P| \neq 0$

Finita la ricerca, per selezionare il miglior candidato si utilizza una politica di tipo *WTA* (Winner Takes All). Il vertice CR^3 , per il quale il template ha totalizzato il maggior valore di vero somiglianza rispettando tutti i vincoli, viene scelto come centro effettivo della testa. In questo modo è possibile delimitare, con una certa precisione, il volume occupato dalla testa mediante una sfera di raggio $(innerRay + outerRay)/2$ centrata in C . Se, in tutte le possibili posizioni di ricerca, il template non riesce a raccogliere punti a sufficienza e/o i punti al suo interno non rispettano i vincoli imposti, allora concluderemo che il blob esaminato non contiene una testa al suo interno.

Nella figura 2.16.(a) possiamo notare il funzionamento comune del blocco di *head detection*, per ogni blob che contiene una persona la testa viene riconosciuta. In 2.16(b) e 2.16(c) notiamo come il blocco non abbia correttamente trovato alcuna testa all'interno di blob che non contengono persone, nel primo caso il blob contiene solamente un braccio alzato; mentre nell'ultimo caso contiene una cassettera per attrezzi. Le informazioni ricavate tramite il blocco di *head detection* (vero somiglianza, numero di punti positivi, posizione centro della testa, ecc) vengono salvate, per ogni componente, all'interno del proprio descrittore insieme a tutte le altre informazioni già raccolte.

2.4.4 People Counting

Nel terzo e ultimo stadio dell'algoritmo vengono eseguite le operazioni di alto livello. L'obiettivo della nostra applicazione è quello di identificare quante persone hanno attraversato il varco o permangono nei pressi dello stesso. Dallo stadio precedente di *Image Analysis* si sono ereditate le informazioni degli elementi nella scena, e nel caso di scena affollata, si è potuto discriminare quante persone erano presenti. Le operazioni si basavano esclusivamente sull'immagine corrente. Mentre in questa

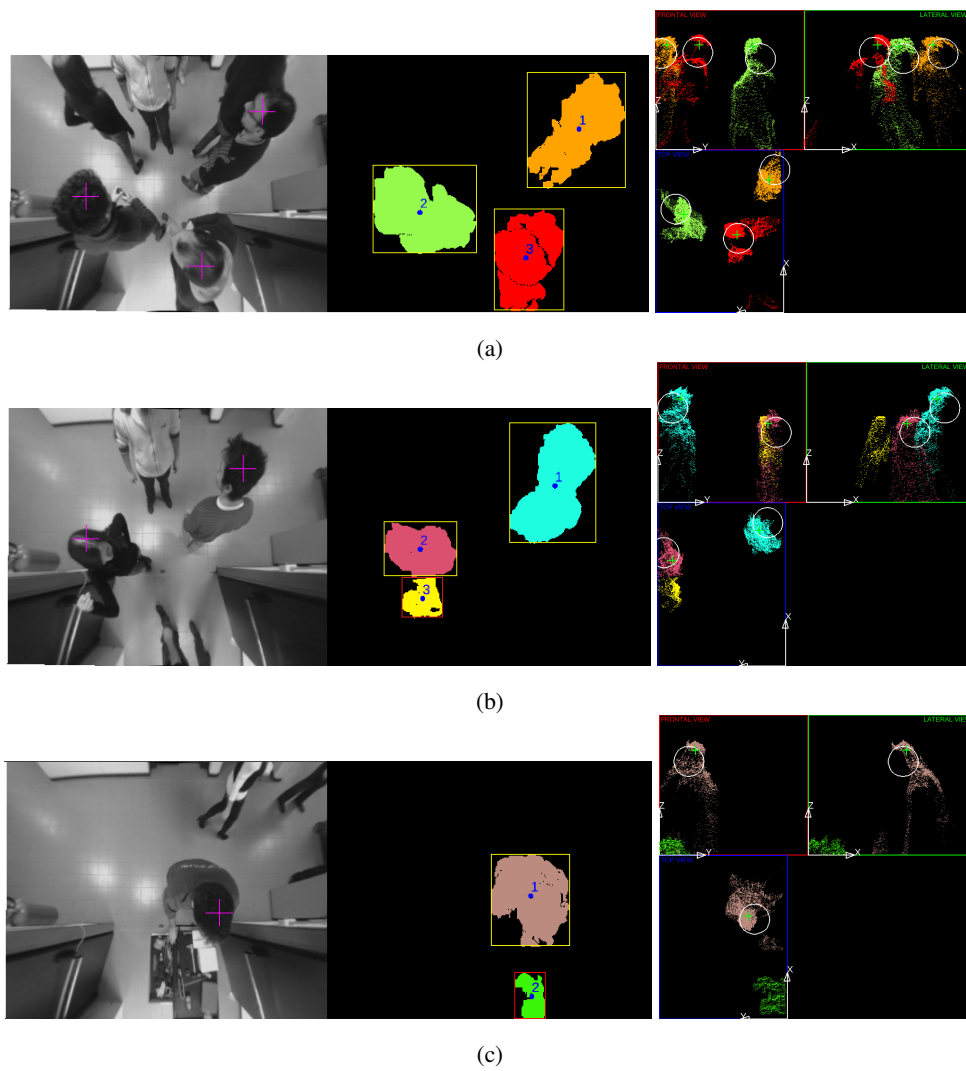


Figura 2.16: Head Detection: (a) Scenario comune, (b) esempio con eliminazione di blob non testa, (c) esempio con eliminazione blob non persona.

fase, l'insieme dei processi è indirizzato alla classificazione, identificazione, e localizzazione condotta allo scopo di ottenere un processo decisionale che si basi sui dati forniti dalle immagini precedenti. Ogni blob delle immagini precedenti, verrà associato ai blob dell'immagine corrente mediante tecniche di tracking per poter tracciare gli spostamenti e in caso di occlusione (quando un individuo nasconde un altro nella scena) mantenere l'informazione. Concluse le seguenti operazioni sarà possibile implementare un semplice meccanismo di conteggio per quantificare il numero di persone nella scena.

Tracking

Il tracking, è un processo che permette di individuare gli oggetti che si muovono nella scena, fornendo in uscita la loro posizione ed associare gli oggetti tra fotogrammi consecutivi, o in generale seguenti. Molti sistemi utilizzano un modello del moto che descrive come l'immagine, potrebbe cambiare negli istanti futuri. Nel nostro caso, gli oggetti da inseguire sono le persone che hanno ottenuto una classificazione valida, e per il quale è stata riconosciuta la testa durante la fase di head detection

Per tenere traccia di un blob nel tempo sono stati utilizzati due vettori:

- un vettore che contiene i blob nell'immagine attuale;
- un vettore che contiene le informazioni sui blob storici presenti nei precedenti K frame.

Usando solo questi due vettori si è in grado di tener traccia degli spostamenti delle persone nella scena. Per diminuire l'errore introdotto in fase di identificazione, si è pensato di basarsi sulle informazioni storiche dei blob, per predire la loro posizione futura, mediante l'utilizzo del filtro di Kalman. Si utilizza una caratteristica di ogni blob chiamata età, che rappresenta il periodo di tempo in cui lo stesso rimane presente nella scena inquadrata. L'età viene incrementata quando un oggetto non è identificato, ovvero non viene assegnata una corrispondenza univoca tra la posizione nei frame precedenti a quella attuale. Le ragioni per cui un oggetto non è identificato sono molteplici:

- la persona esce dalla scena;
- vi è occlusione tra persone;
- non è stato possibile identificare la testa, per cui viene eliminato il blob. Questa precauzione è stata presa perchè si preferisce avere dei *falsi positivi* (oggetti non identificati nella scena), piuttosto che dei *falsi negativi* (oggetti che vengono riconosciuti nella scena, ma non presenti).

Quando la variabile supera una determinata soglia, significa che il blob non è presente nella scena per parecchi frame, per cui si procede con l'eliminazione. Ogni blob presenta al suo interno un filtro di Kalman. Il filtro di Kalman, è stato implementato per poter predire la posizione del blob nel frame corrente, in quelli successivi. Il filtro di kalman si comporta come un osservatore dello stato, e tiene conto della posizione e la velocità degli oggetti. È caratterizzato da due fasi:

- *Predizione* dello stato futuro.
- *Correzione* della predizione precedente con la posizione del blob nell'immagine corrente.

L'alternanza di queste operazioni permette di diminuire l'errore di tracciamento. Nel caso in cui un blob non venga identificato per le ragioni precedentemente descritte, non viene fatta l'operazione di correzione, poichè non si è sicuri di dove sia effettivamente il blob. L'algoritmo di tracking utilizzato è sintetizzato in figura 2.17. Dopo la fase di *head detection* i blob a cui non è stata riconosciuta la testa vengono eliminati, per cui si passa alla fase di associazione composta da tre casi:

- CASO A: la lista dei blob storici è vuota, mentre nella scena sono stati identificati degli oggetti. Si copiano le informazioni nel blob attuale in quello storico. Questo avviene sempre all'inizio del sistema o quando per un pò di tempo nessuno staziona nel volume di interesse.
- CASO B: sia la lista dei blob storici che quella dei blob attuali sono piene. Si ha una fase di associazione basata sul principio di vicinanza. Siccome le persone

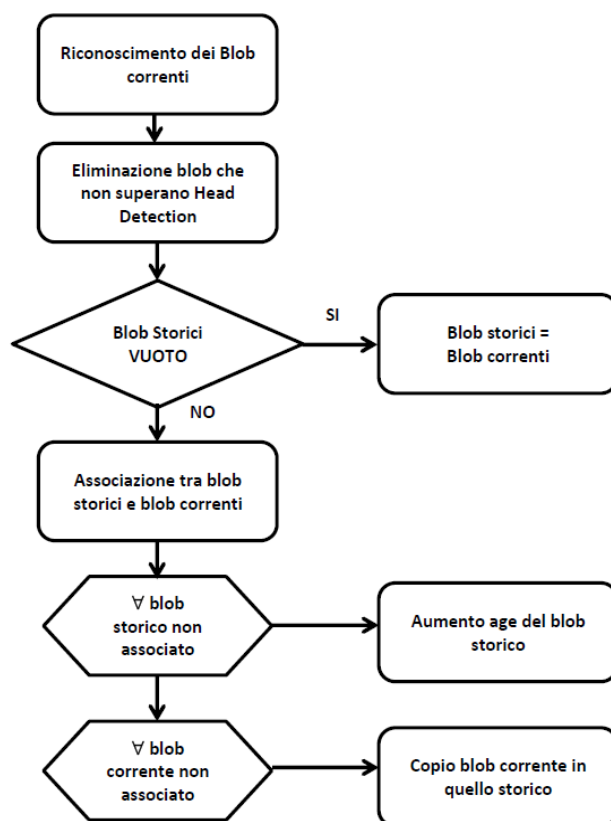


Figura 2.17: Algoritmo di tracking utilizzato nel conta persone stereo.

si muovano camminando alla velocità di $5km/h$ (comunemente considerata la velocità media di un essere umano adulto) si può calcolare lo spostamento della persona da un frame al successivo come:

$$s = v \cdot t = \frac{5}{3.6} \cdot \frac{1}{25} = 0.05m$$

Il calcolo è stato considerato supponendo un frame rate di 25 Hz. La misura ricavata di 0.05, corrisponde a circa 10 pixel nell'immagine mentre il fattore di scala è di circa 200 pixel/m. Considerando plausibile che una persona possa an-

che correre anche ad una velocità doppia di quella usata nel precedente calcolo, si arriva a dedurre che un blob si può spostare da un frame all'altro di circa 20 pixel. Per cui la vicinanza tra un frame e l'altro è un buon parametro di identificazione. Per ridurre l'errore si è pensato di utilizzare sia l'informazione della posizione nell'istante precedente, sia l'informazione predetta dal filtro di Kalman. Ottenute tutte le distanze tra tutti i frame, si procede con un algoritmo di ottimizzazione, per assegnare l'identificativo dei blob storici. Quando si ottiene una corrispondenza univoca, si azzerla la variabile *age* e si copia la posizione attuale nella lista dei movimenti. Eseguo infine l'operazione di correzione del moto con Kalman.

- CASO C: nel caso in cui la lista dei blob storici sia piena, mentre la lista dei blob attuali è vuota, si passa ad incrementare l'age dei blob storici.

Terminata questa fase, si ricercano i blob non assegnati dalla lista storica e dalla lista attuale. Per i blob della lista storica viene incrementato l'age. Per ogni blob della lista attuale non associato, viene copiato nella lista storica assegnandogli un nuovo identificativo. Quando l'age supera una determinata soglia, viene eliminato dalla lista storica.

Counting Algorithm

Il progetto permette di analizzare le code e verificare il numero di persone che transitano all'interno del volume di interesse. Per poter contare le persone che entrano o escono dal varco è stata utilizzata una linea virtuale chiamata Cross Region. La persona viene classificata come entrata o uscita, a secondo del moto della sua direzione ottenuta col tracking. In letteratura sono presenti alcune soluzioni che propongono una o più linee. Si è scelto di utilizzare una linea sola poichè si hanno le informazioni dettagliate del moto. L'implementazione necessita di un solo flag e non aumenta il carico computazionale. Ad ogni oggetto viene assegnato il flag opportuno a secondo della sua posizione nella scena:

- Se l'oggetto si trova al di sotto della Cross Region: *flagUp=false, flagDown=true*

- Se l'oggetto si trova al di sopra della Cross Region: *flagUp=true*, *flagDown=false*

Quando invece un oggetto oltrepassa la cross region si hanno le seguenti situazioni rappresentate in figura 2.18, gli algoritmi sono illustrati nei rispettivi grafici.

Supplementary Algorithms

Questo blocco rappresenta le possibili personalizzazioni per il sistema di controllo di accesso ai varchi, tenendo presente la struttura generale dell'algoritmo. Si possono inserire in questo blocco alcune funzioni:

- Segnalazione della stima di persone all'interno del volume di interesse: per applicazioni legate all'apertura delle porte su un veicolo per il trasporto urbano.
- Riconoscitore di traiettorie: per il controllo dello spostamento delle persone nell'area di interesse.
- Segnalazione del volume di interesse più libero: nel caso in cui si vogliano monitorare più volumi di interesse e ricavare informazioni sulla quantità di persone che permangono al loro interno, ad esempio per suggerire quale coda risulta probabilmente più veloce.

2.4.5 Risultati ottenuti

Sequenza	Totale Persone	Ground Truth (IN/OUT)	Sistema (IN/OUT)	Correct Detection	Errore
Sequenza1	171	87/84	86/83	99.08%	0.02%
Sequenza2	163	78/78	78/76	99.04%	0.06%
Sequenza3	108	53/55	53/55	100%	0.0%
TOTALE	442	218/224	217/214	99.065%	0.035%

Tabella 2.3: Risultati ottenuti per il sistema di accesso ai varchi con telecamera stereo.

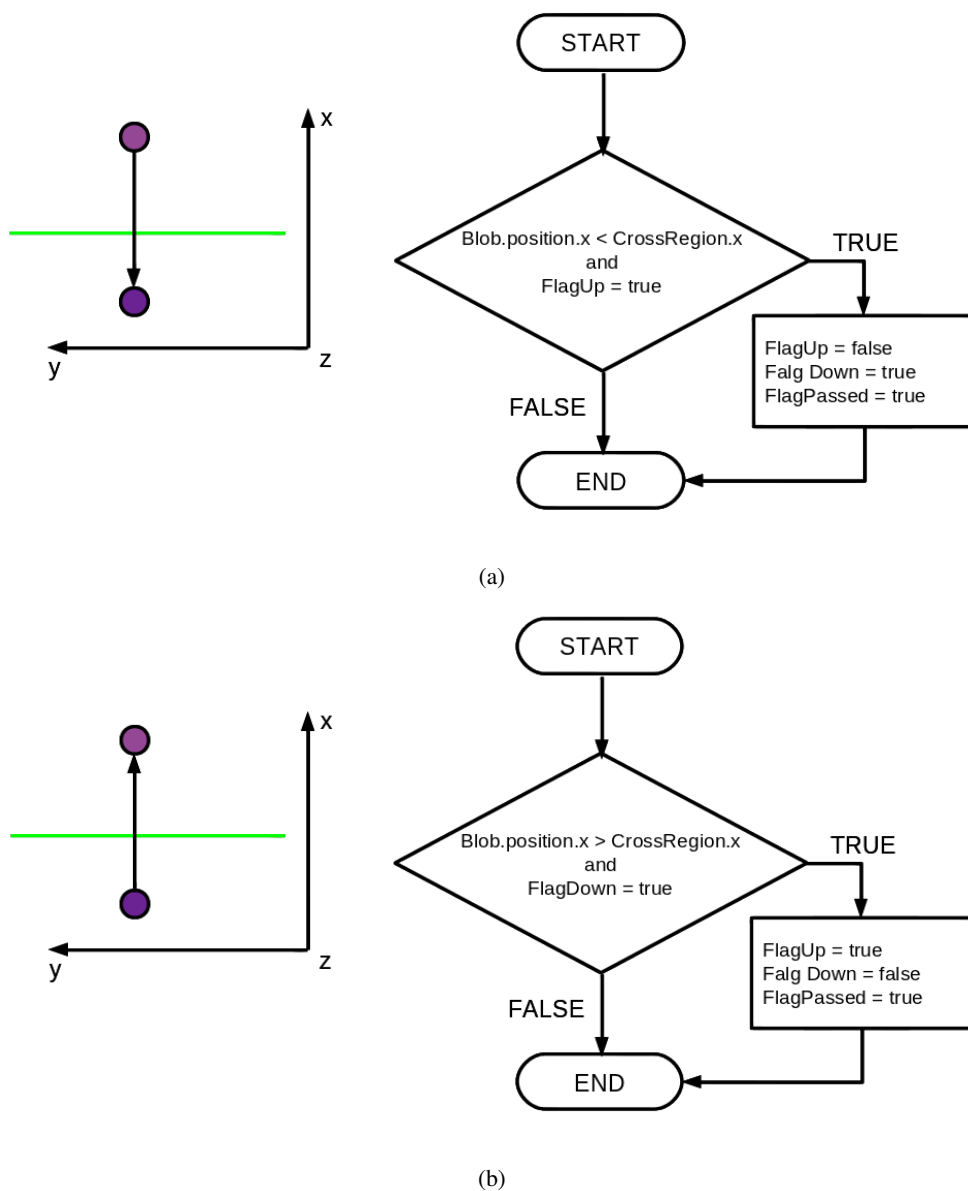


Figura 2.18: Head Detection: attraversamento della linea da parte del centro di massa e diagramma di flusso per (a) uscita e (b) entrata

La tabella 2.3 mostra i risultati ottenuti con l'algoritmo per il conteggio delle persone che attraversano un varco mediante l'utilizzo di una coppia stereo di telecamere. Sono stati effettuati tre giorni di riprese per ottenere tre sequenze con una quantità di persone sufficiente a validare i risultati. In totale sono passate attraverso il varco un totale di 442 persone di cui 218 in entrata e 224 in uscita, il ground truth è stato ricavato mediante annotazione delle sequenze da un operatore umano. Come si può vedere dalla tabella le rilevazioni corrette hanno una percentuale molto alta, pari al 99.065%, questo può far pensare che le tecniche sviluppate risultano essere appropriate per il buon funzionamento del sistema.

2.5 Conclusioni

Le attività di ricerca connesse alla camera fissa hanno mostrato di produrre ottimi risultati sia dal punto di vista delle prestazioni che dal punto di vista delle innovazioni prodotte espandendo l'area di ricerca legata alla videosorveglianza. I sistemi realizzati nell'ambito di questa attività di ricerca sono divenuti dei prodotti commerciabili da parte dello spin-off VisLab s.r.l del Dipartimento di Ingegneria dell'Informazione dell'Università degli Studi di Parma. I risultati sono confrontabili con quanto si può trovare in letteratura e le tecniche vanno ad incrementare ed irrobustire gli algoritmi presenti nello stato dell'arte. I sistemi presentano un alto livello di modularità, e la possibilità di impiego in molte situazioni.

Per quanto riguarda il confronto tra i due maggiori approcci utilizzati si può affermare che i sistemi di tipo stereo riescono sicuramente a migliorare le prestazioni in termini di risultati e di precisione del controllo della scena, a scapito però della semplicità di installazione tipica dei sistemi con singola telecamera. Per avere informazioni precise sul movimento e sulla localizzazione delle persone è inoltre da considerare un primo processo di calibrazione, che però molte volte può essere necessario solamente nel caso di un approccio con telecamere stereo, poichè queste necessitano della conoscenza dei punti mondo per l'associazione con i punti dell'immagine per la generazione dell'immagine di disparità.

In generale, considerando le motivazioni che hanno spinto il lavoro di questa parte di tesi di dottorato bisogna ritenersi soddisfatti per i risultati ottenuti in termini di prestazioni, considerando che si possono in ogni caso sempre migliorare gli algoritmi considerando nuove tecniche o nuovi approcci.

Capitolo 3

Elaborazione di immagini con camera mobile

*Chi ha l'occhio, trova quel che cerca anche a occhi chiusi.
L'occhio non vede cose ma figure di cose che significano altre cose.*

Italo Calvino

In questo capitolo verranno illustrati alcuni sistemi sviluppati durante l'attività di ricerca mediante l'utilizzo di immagini acquisite mediante camera mobile. I sistemi sviluppati sono legati principalmente al campo automotive e in particolare ai due maggiori settori di ricerca: i veicoli autonomi e i sistemi di assistenza alla guida avanzati, ADAS (Advanced Driver Assistance Systems).

Le situazioni descritte esprimono risultati innovativi nel solco dello stato dell'arte, apportando un significativo contributo scientifico. Alcuni tra i sistemi che verranno illustrati nel capitolo sono già stati presentati presso conferenze internazionali.

Per la realizzazione degli algoritmi si è fatto largo uso di tecniche di intelligenza artificiale e di apprendimento automatico; queste sono illustrate nelle appendici per facilitare la comprensione del campo di ricerca.

3.1 Sistema per il riconoscimento di pedoni e veicoli

In questa sezione verranno presentate due sistemi sviluppati in collaborazione con Mando Corporation.

Il primo prevede la protezione attiva dei pedoni, gli utenti della strada più cagionevoli e indifesi nel caso di incidente, il secondo prevede la realizzazione unicamente mediante tecniche di visione artificiale di un sistema di assistenza alla guida avanzato che inglobi entrambi i sistemi di controllo del mantenimento della corsia, *lane keeping assistance*, e di controllo attivo di crociera, *active cruise control*.

All'interno di questi progetti l'attività di ricerca si è focalizzata sullo sviluppo di un riconoscitore di pedoni per l'attivazione della frenata in caso di prevista collisione, per scongiurare o attutire la collisione e sullo sviluppo di un riconoscitore di veicoli per il controllo di crociera.

3.2 Algoritmo per il riconoscimento di pedoni in immagini NIR

Il riconoscimento dei pedoni, in letteratura noto come *pedestrian detection*, utilizzando la visione artificiale non è da considerarsi un compito banale. Le persone indossano vestiti varie forme e colori che le rendono difficili da distinguere rispetto alla scena che li contiene. Inoltre, i pedoni possono trovarsi in qualunque punto della scena inquadrata, soprattutto per quanto riguarda l'ambiente stradale, frequentemente portano oggetti come borse, ombrelli, scatole, che modificano quella che può essere interpretata come loro forma.

In questo capitolo verrà presentato un sistema per il riconoscimento di pedoni basati su classificatori deboli in cascata secondo quanto indica lo stato dell'arte.

3.2.1 Stato dell'arte

Lo stato dell'arte di questa branca della visione artificiale risulta molto ricco di proposte, alcune presentano dei metodi associati ed affermati ed utilizzati su larga scala in questi ultimi anni. Tutti i sistemi si possono dividere in due passi, un passo di

estrazione dell'area di interesse, chiamato generazione delle ipotesi, *hypotesis generation*, ed un passo di classificazione basato su tecniche di intelligenza artificiale e apprendimento automatico chiamato validazione dell'ipotesi *hypotesis validation*.

Differenti metodi sono stati sviluppati per avere la classificazione migliore, normalmente essi sono basati sulla ricerca nell'intera area di fronte al veicolo. I potenziali candidati, nella fase di generazione delle ipotesi, vengono localizzati mediante l'utilizzo di caratteristiche quali forma, simmetria, texture, movimento, periodicità del movimento delle gambe, come indicato in [18].

Si può pensare di introdurre un ulteriore livello di rimozione dei pedoni che non rappresentano un pericolo, come indicato in [19], in cui si evidenzia l'utilizzo delle informazioni date dalla previsione delle traiettorie e quindi delle possibili collisioni. L'approccio descritto in [20] è differente perchè focalizzato solamente sull'identificazione di situazioni di chiaro pericolo, limitando la ricerca dei pedoni in aree dello scenario specifiche. Utilizzando un laser scanner è possibile ricostruire l'ambiente, localizzare aree pericolose e ricercare i pedoni unicamente in queste aree.

Normalmente partendo dalle ipotesi generate, i candidati vengono validati mediante varie tecniche di classificazione, questa risulta spesso la parte più critica. Principalmente nel caso dell'elaborazione di immagini nello spettro NIR si utilizzano differenti metodi, molti basati sul meta-algoritmo AdaBoost [21] con e sulle Support Vector Machines [22]. Una panoramica ampia dei metodi utilizzati in base alle immagini acquisite e in base alle richieste specifiche di calcolo e di risposta in tempi di esecuzione si può trovare in [23] e [24], due studi che in modo analitico riassumono le tecniche maggiormente utilizzate.

3.2.2 Panoramica dell'algoritmo

La figura 3.1 mostra lo schema dell'algoritmo del riconoscitore di pedoni, le immagini acquisite dalla telecamera subiscono una prima pre-elaborazione responsabile anche dell'estrazione delle aree di interesse *bounding box* che contengono le ipotesi per il riconoscitore. All'interno dei *bounding box* sono contenuti pedoni e non pedoni. Le aree di interesse sono estratte mediante l'ausilio di un laserscanner.

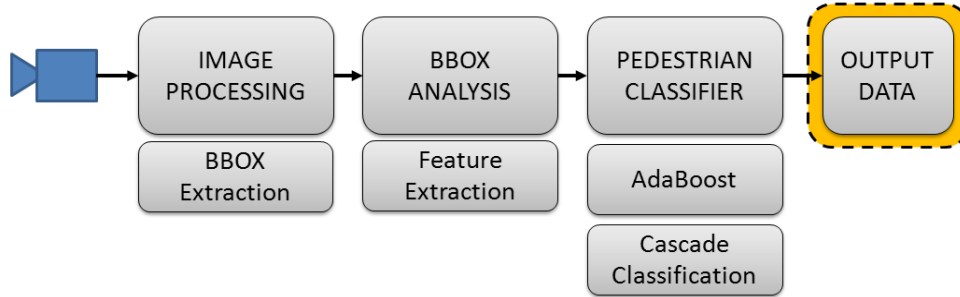


Figura 3.1: Struttura dell'algoritmo di riconoscimento di pedoni.

La fase di analisi del bounding box contiene gli algoritmi per l'estrazione delle caratteristiche salienti, *feature*, utilizzate in seguito dal classificatore AdaBoost. Le caratteristiche utilizzate verranno descritte nei paragrafi successivi.

Il classificatore è formato da due blocchi funzionali principali, il blocco che utilizza l'algoritmo AdaBoost per decidere se il bounding box deve essere classificato come pedone o come non pedone, e un blocco di *Cascade Classification* che si occupa di rafforzare la classificazione mediante l'utilizzo di una cascata di classificatori, sempre basati sull'algoritmo AdaBoost.

Image Processing

Un esempio di immagine del bounding box estratto ed utilizzato come immagine di input è mostrato in figura 3.2.(a). La figura 3.2.(b) mostra l'immagine filtrata con un filtro di estrazione dei contorni di Sobel, e binarizzata per avere la possibilità di estrarre delle caratteristiche basate sull'immagine dei bordi. Nelle figure 3.2.(c) e 3.2.(d) sono mostrate le operazioni di estrazioni dei contorni verticali e orizzontali, rispettivamente con evidenziati gli istogrammi del numero di pixel bianchi per riga e per colonna. Ogni bounding box ha dimensione 24×48 pixel.

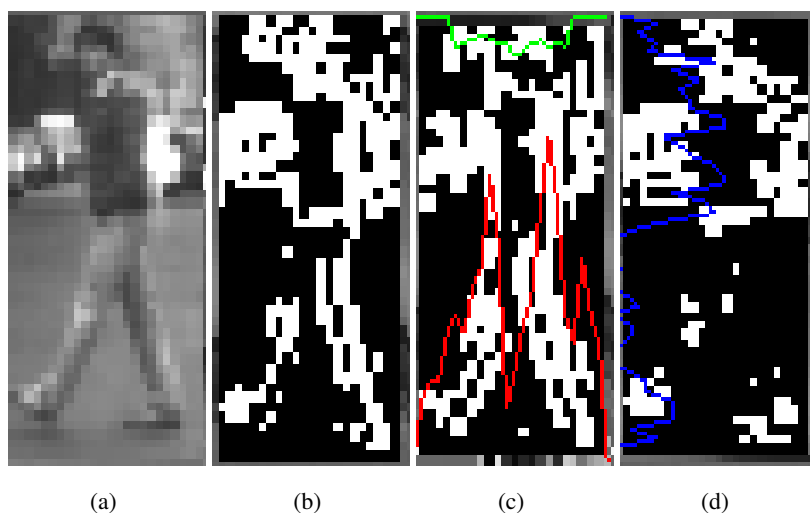
Bounding Box Analysis

Figura 3.2: (a) bounding box contenente un pedone, (b) immagine filtrata con Sobel, (c) immagine filtrata con Sobel verticale con istogramma verticale, (d) immagine filtrata con Sobel orizzontale con istogramma orizzontale.

Il blocco dell'algoritmo descritto in questa sezione utilizza delle feature ad-hoc estratte tramite l'esame approfondito delle caratteristiche dei pedoni utilizzando un numero considerevole di sequenze. Le sequenze sono state registrate nel traffico urbano e su strade extraurbane con differenti condizioni di illuminazione e ambientali, con sole, nebbia, pioggia, al mattino, pomeriggio, crepuscolo, e sera.

Le feature individuate per la classificazione sono basate su peculiarità dei pedoni, come la forma e l'aspetto delle parti del corpo e sono riassunte nella seguente lista:

- Open Legs shape
- Left step legs, little mask
- Right step legs, little mask
- Left step legs, big mask

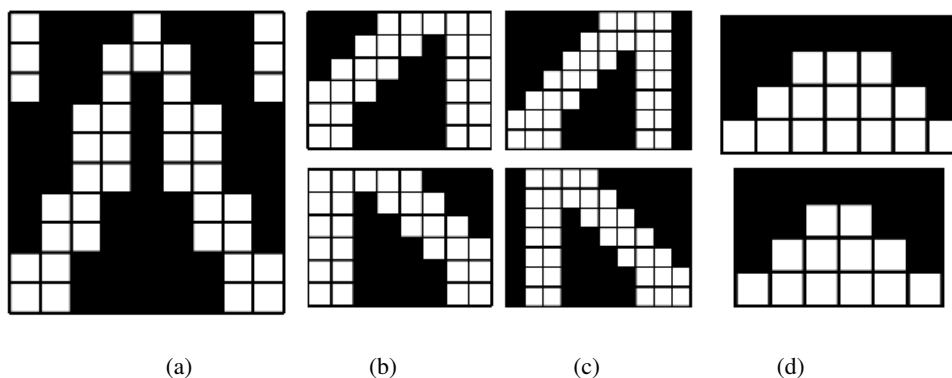


Figura 3.3: Maschere utilizzate per il pattern matching. (a) Maschera per la ricerca delle gambe aperte. (b) Maschere piccole per la ricerca delle gambe. (c) Maschere grandi per la ricerca delle gambe. (d) Maschere per ricerca della testa.

- Right step legs, big mask
- Head top, little mask
- Head top, big mask
- Head, with circle formula
- Pedestrian's M function
- Horizontal border function
- Tree and pole search function

Considerando le immagini in figura 3.2 in ingresso al blocco si nota come le feature hanno uno stretto legame con queste. L'immagine di Sobel 3.2.(b) viene utilizzata con le feature basate su maschere e pattern matching, per identificare la testa e le gambe. Le immagini dei bordi orizzontali e verticali 3.2.(c) e 3.2.(d) vengono utilizzate per calcolare gli istogrammi del numero di pixel bianchi per ogni riga o colonna dell'immagine dai quali verranno estratte le ultime tre feature.

Legs shape on maximal aperture

La forma delle gambe di un pedone che attraversa la strada in posizione frontale rispetto al veicolo è una caratteristica molto forte. Nel momento in cui si ha il cambio di passo le gambe disegnano un profilo a “V” rovesciata; l’algoritmo cercherà il miglior match tra questa feature e l’immagine dei bordi di Sobel tramite il confronto con una maschera di dimensione 10×9 pixel, illustrata nella figura 3.3.(a). Il confronto avviene nella parte inferiore del bounding box.

Forward step legs shape

L’utilizzo di una maschera singola non è abbastanza indicativa delle gambe, che cambiano forma molte volte durante il cammino. Altre feature possono essere estratte utilizzando una maschera simile alla precedente che cerchino di individuare il passo verso destra e verso sinistra. In questo caso vengono utilizzate 4 differenti maschere riassunte nelle figure 3.3.(b) e 3.3.(c); queste maschere sono di due dimensioni diverse: 8×6 pixel e 10×7 pixel. Anche in questo caso la sovrapposizione avviene con la parte inferiore dell’immagine contenuta nel bounding box filtrata con Sobel.

Head Presence

Una caratteristica immancabile in un pedone è la presenza della testa, molto semplice da individuare solo sulla carta dato che in molti casi risulta occlusa da ombrelli o con una forma non immediatamente riconoscibile per effetto di copricapi. Nei casi in cui la testa risulti immediatamente riconoscibile però si può utilizzare una maschera con una forma circolare e con una forma semicircolare per individuare la parte superiore della testa. Le maschere utilizzate a questo scopo sono illustrate dalla figura 3.3.(d), e hanno una dimensione di 6×4 e 7×4 pixel per quanto riguarda le calotte superiori. Per quanto riguarda la ricerca di una testa di forma circolare il cerchio da ricercare all’interno dell’immagine ha un raggio di 3 pixel con un numero totale di 16 pixel. La ricerca avviene nel terzo superiore del bounding box.

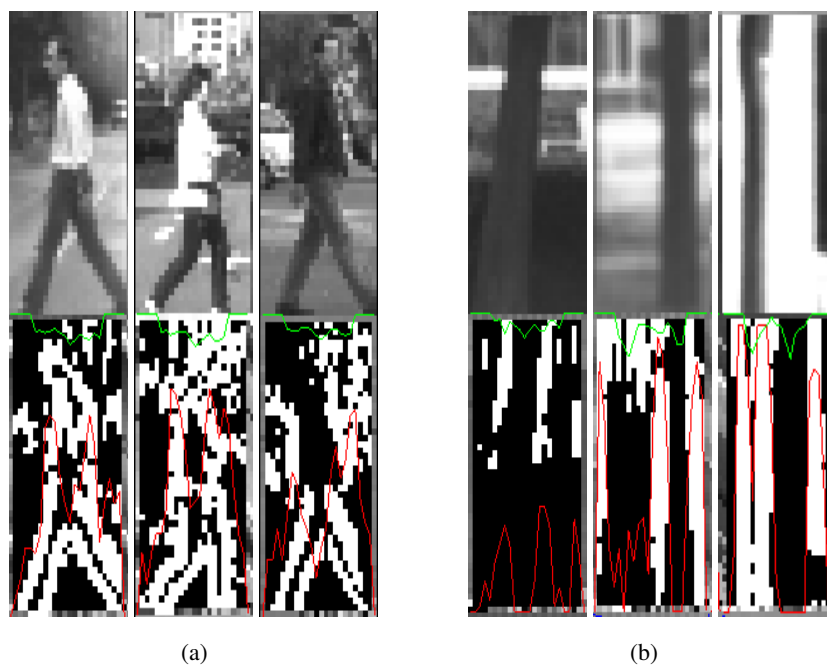


Figura 3.4: Bounding box contenenti: (a) pedoni, (b) alberi e pali.

M shaped pedestrian histogram

Nell'immagine dei bordi verticali binarizzata che include un pedone con una forma ben definita è possibile individuare la presenza di una forma dell'istogramma molto simile ad una "M".

In figura 3.4.(a) si nota molto bene questa forma, nella curva sovrapposta all'immagine che rappresenta l'istogramma. Considerando un alto numero di immagini è possibile ricavare dei valori di riferimento per avere un indice di verosimiglianza e

di regolarità per quanto riguarda la forma.

$$\begin{aligned}
 f((x_m, y_m), (x_{MS}, y_{MS}), (x_{MD}, y_{MD})) = & \\
 & + \left(1 - \left| \frac{y_m - 14.0}{16} \right| \right) \\
 & + \left(1 - \left| \frac{y_{MS} - y_{MD}}{26} \right| \right) \\
 & + \frac{1}{2} \left(1 - \left| \frac{|y_{MS} - y_m| - 10}{30} \right| \right) \\
 & + \frac{1}{2} \left(1 - \left| \frac{|y_{MD} - y_m| - 10}{30} \right| \right) \\
 & + \left(1 - \left| \frac{|x_{MS} - x_{MD}| - 8}{4} \right| \right) \quad (3.1)
 \end{aligned}$$

La formula 3.1 mostra la funzione con cui si ottengono i valori di verosomiglianza, che sono compresi nell'intervallo $[0, 1]$. Le soglie sono state ricavate sperimentalmente.

Horizontal borders

L'istogramma del numero di pixel bianchi per riga ottenuto dall'immagine dei bordi orizzontali binarizzato viene utilizzato come ulteriore feature. Viene calcolata la media aritmetica del primo terzo superiore del bounding box, come indicato dalla formula:

$$f(x) = \frac{\sum_{k=0}^{K/3} x_k}{K/3} \quad (3.2)$$

Dove x_k è il valore dell'istogramma con l'indice $k \in K$ che rappresenta il valore del bordo considerato nell'insieme dei K valori di bordo orizzontale. Questa feature permette di scartare tutti quei bounding box che non presentano bordi verticali nella parte superiore o che hanno una media troppo bassa come nel caso di pali o fusti di albero. Al contrario nella situazione in cui il bounding box contenga veicoli o parti di essi sperimentalmente si è riscontrata un'alta media di bordi orizzontali.

Tree

Spesso i classificatori confondono alberi e pali con i pedoni, poichè hanno molte caratteristiche in comune: entrambi sono oggetti con forma fortemente allungata e verticale e riempiono il bounding box. La figura 3.4.(b) mostra alcuni esempi di bounding box che contengono alberi o pali con evidenziati gli istogrammi della quantità di pixel bianchi per riga nell'immagine dei bordi di Sobel. Con queste informazioni possiamo ottenere una feature addizionale che tiene conto della possibilità di avere un albero o un palo in un bounding box.

La formula generale per la ricerca di un palo o di un albero all'interno del bounding box è la seguente:

$$f() = \begin{cases} f_1 & \text{if } y_{MS} \cdot y_{MD} \geq \tau \\ f_2 & \text{if } \overline{y_{MS} + y_{MD}} \geq \tau \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

Nel caso della presenza di due massimi nell'immagine dei bordi verticali come mostrato in figura 3.4.(b) viene ricercato il minimo e infine calcolata la somiglianza con parametri ricavati sperimentalmente, tramite la funzione f_1 :

$$\begin{aligned} f_1((x_m, y_m), (x_{MS}, y_{MS}), (x_{MD}, y_{MD})) &= + \left(1 - \left| \frac{y_m}{14} \right| \right) \\ &+ \frac{1}{2} \left(1 - \left| \frac{|y_{MS} - y_m| - 42}{32} \right| \right) \\ &+ \frac{1}{2} \left(1 - \left| \frac{|y_{MD} - y_m| - 42}{32} \right| \right) \\ &+ \left(1 - \left| \frac{|x_{MS} - x_{MD}| - 6}{4} \right| \right) \end{aligned} \quad (3.4)$$

Tuttavia nel caso in cui sia presente un solo massimo viene ricercata la presenza di un palo in accordo alla formula f_2 :

$$\begin{aligned} f_2((x_m, y_m), (x_M, y_M)) &= + \left(1 - \left| \frac{y_m}{14} \right| \right) \\ &+ \frac{1}{2} \left(1 - \left| \frac{|y_M - 42|}{32} \right| \right) \end{aligned} \quad (3.5)$$

In entrambi i casi il voto finale viene normalizzato nell'intervallo tra 0 e 1 per coerenza con le altre feature.

3.2.3 Blocco Pedestrian Classifier

Il blocco del classificatore di pedoni utilizza l'algoritmo AdaBoost sui valori di match delle feature estratte nei blocchi precedenti. L'insieme di feature ottenuto è stato sottoposto all'analisi ONE Way ANOVA per fare in modo che lo stesso risultasse sensato, cioè che dato il set costruito si possa in qualche modo trovare una distinzione tra le classi. In ogni caso, utilizzando una singola feature non è possibile discriminare tra le classi, o utilizzando delle soglie fissate, per questo motivo è stato implementato un'algoritmo che classifica il contenuto del bounding box basato su AdaBoost. Per ogni caratteristica viene assegnato un peso e quattro differenti tipologie di sogliature sono state utilizzate per la classificazione:

- minimum (1 threshold)
- maximum (1 threshold)
- included (2 thresholds)
- excluded (2 thresholds)

Le soglie sono state ricavate grazie all'addestramento del classificatore su un training set appositamente costruito. Il voto finale di classificazione finale, considerando tutte le feature introdotte, permette di discriminare tra due classi, pedoni e non pedoni. Se indichiamo con $h(x)$ l'ipotesi sulla regione di interesse x abbiamo:

$$h(x) = \sum_{i=0}^N v_i(x) \quad (3.6)$$

Dove v_i è il voto dell' i -esimo classificatore, matching di ogni feature, per $i = 0 \dots N$ con N numero delle feature totale.

3.2.4 AdaBoost Cascade

Per incrementare le prestazioni e le potenzialità dell'algoritmo di riconoscimento sono stati inseriti degli stadi di classificatori basati su AdaBoost in cascata, inseriti all'interno del blocco *AdaBoost Cascade*.

In questo blocco i voti ottenuti con il classificatore sviluppato con feature ad-hoc vengono uniti con quelli ottenuti con un classificatore basato su feature di Haar per ottenere un ulteriore livello di classificazione. I voti forniti dal blocco precedente vengono quindi utilizzati come input per avere un secondo tipo di classificatore. Siccome il numero di output in uscita dagli algoritmi citati risulta molto basso, 2 per il classificatore basato su Haar feature, e 1 per il metodo sviluppato con feature ad-hoc e siccome molti test hanno mostrato che singolarmente i metodi usati non raggiungono una classificazione efficiente, sono state utilizzate delle combinazioni non lineari come input per il blocco di classificazione in cascata. In accordo con la classica formula di decisione illustrata dall'equazione 3.6 l'ipotesi finale viene ottenuta considerando una somma pesata dei classificatori selezionando i pesi nella fase di training. Le combinazioni non-lineari sono state realizzate variando i pesi e normalizzando

3.2.5 Risultati ottenuti

Dopo una fase di addestramento del classificatore i risultati sono stati ottenuti con 10 ore di sequenze in differenti condizioni climatiche, ambientali e nei diversi momenti della giornata. Sulle 10 ore di sequenze è stato effettuato un ground truth da parte di un operatore umano. In figura 3.5.(a) è illustrato il funzionamento del sistema, con la rilevazione corretta di un pedone che attraversa la strada, e un altro più avanti, mostrati all'interno di un bounding box che li classifica correttamente. La figura 3.5.(b) mostra le ROC curves per il sistema di riconoscimento di pedoni realizzato con feature ad-hoc confrontato con il sistema realizzato con feature di Haar e dopo l'unione di entrambi con il metodo cascade. Si nota come il metodo basato su semplici feature di Haar risulta molto preciso, più del metodo basato su feature ad-hoc, ma soprattutto come la cascata dei due risultati raggiungere un alto livello di classificazioni corrette. Il sistema inoltre risulta essere molto leggero computazionalmente e può per questo

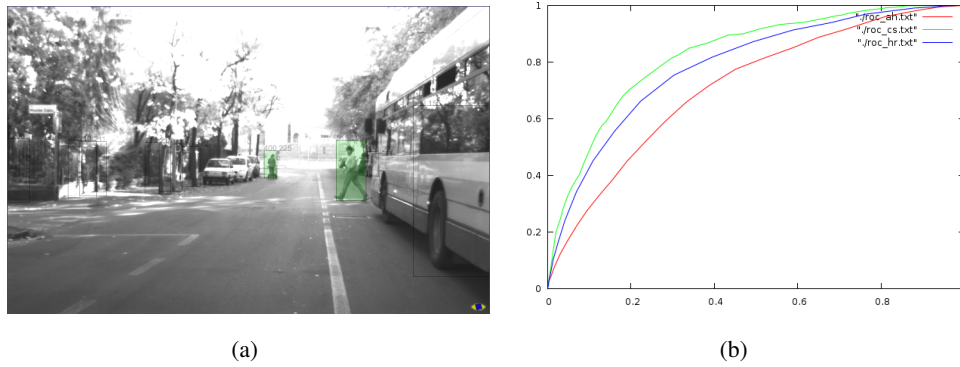


Figura 3.5: Risultati ottenuti per il riconoscitore di pedoni: (a) output, (b) ROC curves relativo al riconoscitore di pedoni con feature ad-hoc, feature di Haar e cascata di classificatori.

essere utilizzato in tempo reale richiedendo sempre meno di 70 ms/frame su un PC con buone performance. Il tutto considerando anche l'elaborazione dei dati da laser scanner rimanendo però molto legato al numero di regioni di interesse da analizzare.

3.3 Riconoscitore di veicoli per applicazioni ADAS

Analogamente a quanto visto per i pedoni si può pensare di ricavare delle feature dall'osservazione delle regioni di interesse di un'immagine per rilevare la presenza di veicoli. In questa sezione viene presentato un sistema per il riconoscimento e la classificazione di veicoli mediante tecniche di elaborazione dell'immagine.

3.3.1 Breve stato dell'arte dei riconoscitori di veicoli

Il riconoscimento di veicoli è una tematica molto dibattuta e cruciale per la guida automatica e per l'area di ricerca che riguarda le applicazioni automotive a largo raggio. Normalmente assumono molta importanza per capire le intenzioni dei veicoli nel mondo circostante, le loro posizioni, e le possibili traiettorie, a volte anche per attuare comportamenti che evitino incidenti o collisioni. Per quanto riguarda l'ambi-

to della visione artificiale si utilizzano telecamere per il riconoscimento dei veicoli, molto spesso i dati iniziali vengono fusi con quelli provenienti da altri sensori che restituiscono la distanza, come laserscanner o radar. Anche i veicoli, analogamente ai pedoni, sono molto difficili da riconoscere, ne esiste una varietà molto alta e possono essere inquadrati secondo diverse angolature. Nel caso illustrato in questa tesi di dottorato si utilizzeranno solamente tecniche di visione artificiale, per cui, secondo quanto riportato in letteratura riguardo a questo sottoinsieme delle tecniche di *vehicle detection*, si deve ricordare che l'approccio generale per lo sviluppo di un sistema per il riconoscimento di veicoli comprende due fasi distinte [25, 26]:

1. HG: hypotesis generation;
2. HV: hypotesis verification.

La prima fase si occupa di fornire le regioni candidate a contenere dei veicoli, in modo veloce per essere elaborate nella seconda fase, in letteratura esistono tre approcci per questa fase:

- basati su conoscenza: impiegano un metodo basato su una conoscenza a priori per ipotizzare la localizzazione del veicolo nell'immagine, utilizzando informazioni come simmetria, ombre, colori, bordi verticali e orizzontali e texture;
- basati su visione stereo, utilizzando una ricostruzione tridimensionale della scena e la inverse perspective mapping, IPM;
- basati sul movimento: mediante il flusso ottico si possono considerare le parti dell'immagine che si muovono o che sono ferme in relazione con il movimento del veicolo stesso.

Una volta ottenuta una lista di regioni di interesse che contengono le ipotesi si passa alla fase di validazione delle stesse, anche per questa fase si possono distinguere approcci differenti, ne esistono due maggiormente utilizzati in letteratura:

- validazione basata su modelli: si genera un modello per il veicolo e si confronta con quanto contenuto nei bounding box con un voto in base al confronto tra i due.

- validazione basata sull'apparenza: normalmente è legato ad approcci con tecniche di apprendimento automatico. Si addestrano dei classificatori con differenti tecniche direttamente sull'immagine ritagliata delle ipotesi, per discriminare il contenuto. Si possono utilizzare approcci basati su reti neurali, PCA, AdaBoost, SVM.

3.3.2 Algoritmo per il riconoscimento di veicoli

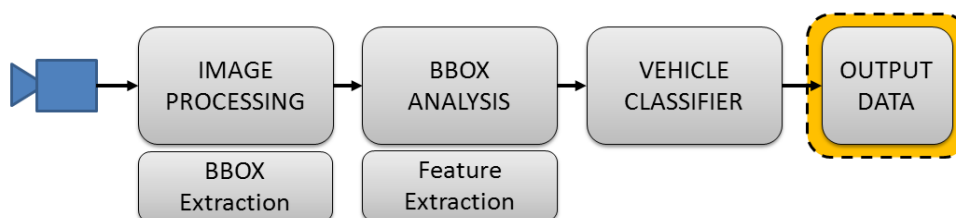


Figura 3.6: Struttura dell'algoritmo riconoscimento di veicoli

Alla luce di quanto emerso dall'analisi dello stato dell'arte l'algoritmo che viene presentato in questa sezione utilizza un approccio basato su conoscenza per la fase di generazione delle ipotesi; mentre, per la classificazione vera e propria si utilizza un metodo ad-hoc utilizzando delle caratteristiche salienti elaborate mediante l'analisi di sequenze reali in scenario autostradale. Le specifiche di progetto limitano molto l'approccio da utilizzare, l'ambiente stradale da considerare è unicamente autostradale, si devono considerare veicoli fino a 80 m e non si possono utilizzare metodi che facciano uso di tecniche di apprendimento automatico. La struttura dell'algoritmo sviluppato è mostrata in figura 3.6.

Blocco Image Processing

Questo blocco si occupa di estrarre le regioni di interesse, eseguendo la fase di *hypothesis generation* secondo un approccio basato su conoscenza a priori, utilizzando la

fusione sensoriale con un radar installato sul veicolo e mediante l'analisi delle simmetrie, della presenza di ombre e di bordi orizzontali e verticali. Queste operazioni non sono comprese nella parte sviluppata all'interno del progetto relativa a questa tesi di dottorato e per questo sono solo accennate.

Blocco BBox Analysis

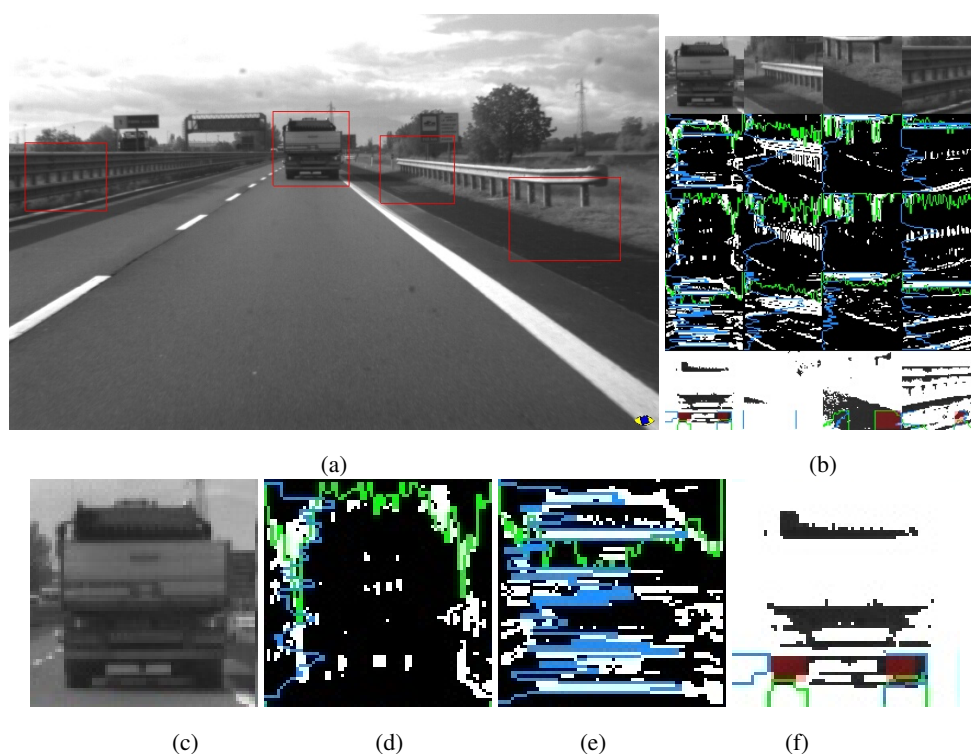


Figura 3.7: Riconoscimento di veicoli: (a) immagine di input, (b) bounding box estratti con pre-elaborazione, (c) veicolo, (d) forma a “W” (e) bordi orizzontali (f) aree scure.

Questo blocco effettua la fase di *hypotesis validation* mediante l'analisi delle regioni di interesse fornite dal blocco precedente. La validazione dei veicoli viene

eseguita con un classificatore che utilizza feature ad-hoc

Per ogni bounding box vengono effettuate alcune operazioni di pre-processing. La prima operazione svolta è un ricampionamento per avere aspect ratio, altezza e larghezza fissati ed avere quindi la stessa risoluzione e poter usare delle soglie fissate per ogni bounding box. La dimensione scelta è di 52×52 pixel per poter considerare i veicoli con alta confidenza fino a 80 m, come mostrato in figura 3.7.(c). In seguito per l'ottenimento delle feature vengono eseguiti dei filtri di Sobel per l'estrazione dei bordi e un'estrazione delle aree scure, mediante una sogliatura che si adatta al valor medio di luminosità dell'immagine. In figura 3.7.(b) sono illustrati i bounding box estratti dall'immagine di input e le operazioni di pre-elaborazione relative.

L'insieme delle feature è formato dalle seguenti caratteristiche salienti:

- presenza delle ruote;
- calcolo di una particolare forma a “W” nell'istogramma dei bordi verticali;
- calcolo di una particolare forma a “W” nell'istogramma dei bordi totali;
- presenza di bordi orizzontali all'interno dell'immagine del veicolo.

Presenza delle ruote

La presenza delle ruote è una feature leggermente forte. Per rilevarle si effettua sul bounding box un filtraggio delle aree scure, dopodichè si cerca nella parte inferiore del bounding box un rettangolo scuro di dimensione adeguata. La figura 3.7.(f) mostra l'estrazione delle aree scure dell'immagine con le ruote del veicolo rilevate nella parte inferiore. Una parte del voto finale è dato dalla distanza dei centri di massa delle ruote rilevate in questo modo:

$$v_1 = 1.0 - |\delta - \delta_{mod}| / \delta_{max} \quad (3.7)$$

Dove v_1 è il primo voto ottenuto, δ è la distanza tra i centri di massa δ_{mod} è la distanza ottimale e δ_{max} è la distanza massima. Il secondo voto v_2 viene calcolato in base alla similarità con un modello ideale di ruota secondo la formula:

$$v_2 = -(1.0/\eta_{th})\eta + 1 \quad (3.8)$$

Dove η è la similarità, η_{th} è la soglia sul valore della similarità. Tutti i parametri sono ottenuti sperimentalmente e i valori dei voti sono normalizzati nell'intervallo tra 0 e 1. Il voto finale che viene assegnato è frutto del rilevamento delle due aree scure rappresentanti le ruote, e delle informazioni geometriche relative alla forma, la somma dei due voti illustrati in precedenza.

W shape feature

Le immagini ottenute dal filtraggio di Sobel vengono binarizzate, da queste immagini vengono poi estratti gli istogrammi del numero di punti bianchi per riga e per colonna. In questo modo si possono analizzare le forme degli istogrammi che mostrano una specie di impronta digitale del contenuto del bounding box. Nell'immagine in figura 3.7.(d) si nota come l'istogramma dei bordi verticali abbia una forma ben determinata che è comune a tutti i veicoli processati in questo modo. Si calcolano le distanze tra i picchi dell'istogramma, la lunghezza di questi e il valor medio dell'istogramma tra questi per ricavare dei valori da confrontare con altri valori ricavati sperimentalmente per indicare un voto per la classificazione relativamente a questa feature. Questa ricerca è applicata all'immagine dei bordi verticali e dei bordi totali.

Presenza di bordi orizzontali

L'ultima feature riguarda la presenza dei bordi orizzontali nella parte del bounding box che rappresenta un veicolo. In figura 3.7.(e) è mostrata la pre-elaborazione per l'estrazione di questa feature. Il voto tiene conto della presenza dei bordi calcolandone la media e la varianza e confrontandole con un valore ricavato sperimentalmente.

Blocco Vehicle Classifier

Per ogni bounding box fornito dal blocco di validazione delle ipotesi questo blocco effettua la classificazione del contenuto per discriminare tra le classi veicolo o non-veicolo tramite una somma pesata secondo la formula:

$$h_i = \sum_{i=1, \dots, N} w_i x_i \quad (3.9)$$

Con $i \in 1, \dots, N$ che rappresenta l'indice del bounding box del frame attuale. Dove $h \in [0, 1]$ è la classe dell'ipotesi, $w_i \in [0, 1]$ è il peso ottenuto in base alla rilevanza della feature, $x_i \in x_1, \dots, x_N$ è il voto della feature estratta dall'immagine, all'interno dell'insieme delle feature. Il voto così ottenuto viene normalizzato nell'intervallo tra 0 e 1 e confrontato con una soglia ottenuta sperimentalmente per discriminare l'appartenenza alla classe veicolo o non veicolo.

Risultati ottenuti

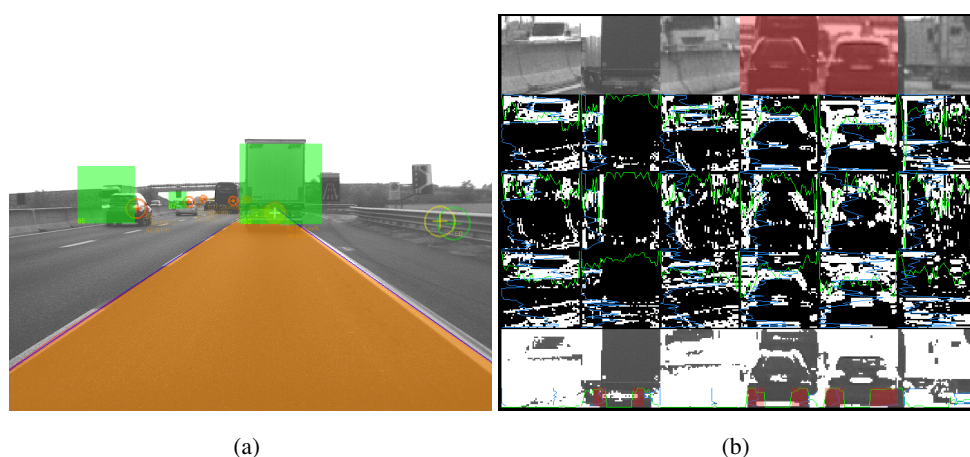


Figura 3.8: Risultati ottenuti: (a) output dell'algoritmo, (b) bounding box relativi al frame.

La figura 3.8 mostra i risultati ottenuti, in figura 3.8.(a) è mostrato l'output dell'algoritmo, con i bounding box classificati correttamente evidenziati nell'immagine. Nella figura 3.8.(b) sono mostrati i bounding box durante le fasi di elaborazione ed estrazione delle feature relative al frame a fianco, con evidenziati i veicoli riconosciuti. Si nota che un veicolo non viene riconosciuto correttamente probabilmente poiché il bounding box non risulta ben centrato. L'operazione di riconoscimento è basata unicamente sul frame e non prevede alcuna forma di tracking.

3.4 Adas di supporto ai sensori del veicolo

In questa sezione verranno illustrati nel dettaglio due sistemi basati su elaborazione di immagini con telecamera fissa non calibrata all'interno di un veicolo che acquisisce la scena in posizione frontale. I sistemi coinvolgono tecniche pure di elaborazione di immagini e anche tecniche di intelligenza artificiale per il riconoscimento dell'ambiente circostante il veicolo. I due sistemi implementati sono:

- Sistema per il riconoscimento di tunnel stradali;
- Sistema per il riconoscimento e la classificazione dello scenario.

Il sistema che ha previsto la realizzazione di un algoritmo per il riconoscimento di tunnel in ambito stradale offre molti benefici dal punto di vista sia della guida automatica che degli ADAS. Dal punto di vista della guida automatica è importante principalmente avvisare i sensori di tipo telecamera del repentino cambio di illuminazione e quindi di visibilità per cambiare i parametri. Dal punto di vista dei sistemi ADAS questo sistema permette di aumentare il comfort alla guida prevedendo l'accensione o lo spegnimento dei fari o modulando i tergicristalli, in ultima analisi la corretta individuazione di tunnel stradali può essere estremamente utile per informare i sistemi di navigazione satellitare dell'imminente mancanza di segnale per attivare altri sistemi che utilizzino tecniche di dead reckoning.

Le considerazioni appena illustrate possono valere anche nel caso in cui si voglia conoscere l'ambiente stradale in cui il veicolo si trova a navigare. Poter distinguere tra ambito urbano, extraurbano, autostradale oppure off-road permetterebbe di modificare al meglio i parametri per un utilizzo ottimale dei sensori montati sul veicolo e per la fusione sensoriale. Si può far riferimento ad un veicolo con molti sensori capaci di avere una visione del mondo esterno a 360°, con un certo livello di sovrapposizione, e quindi di robustezza per la navigazione autonoma. I sensori che si utilizzano normalmente per la realizzazione di veicoli automatici sono molto eterogenei: telecamere e laserscanner per la visione artificiale, lidar, sonar, ma anche sistemi inerziali, odometri e sistemi di navigazione satellitare per la ricostruzione e la pianificazione del moto. Tutti questi sensori hanno punti di forza e debolezza che variano

in base allo scenario in cui è calato il veicolo, per questo potrebbe essere importante avere una conoscenza della tipologia di strada in cui si trova il veicolo.

3.5 Sistema per il riconoscimento di tunnel stradali

articolo tunnel Il sistema è organizzato come una macchina a stati finiti, FSM *finite state machine*. Ogni stato del sistema individua l'ambiente o la situazione in cui si trova il veicolo:

- *OUT*: il veicolo non si trova in una galleria;
- *ENTRANCE*: il veicolo si trova in prossimità dell'entrata di un tunnel; in questo stato si possono attivare eventuali procedure per l'ingresso del veicolo nel tunnel;
- *IN*: il veicolo si trova dentro al tunnel, quindi la procedura di ingresso può considerarsi finita;
- *EXIT*: questo è lo stato opposto allo stato entrance, qui il veicolo è in prossimità dell'uscita del tunnel, si può ripristinare lo stato iniziale del veicolo e del sistema.

Il sistema gestisce le transizioni tra gli stati utilizzando sistemi differenti e il flusso principale di esecuzione del programma tocca tutti i quattro stati nel caso di transito in un tunnel. La struttura del sistema sviluppato è mostrata in figura 3.9, mentre il flusso di esecuzione dell'algoritmo con il rapporto tra la macchina a stati finiti e i blocchi funzionali su cui è basato il blocco fondamentale dell'algoritmo è mostrato in figura 3.10 Lo stato di partenza viene deciso da un algoritmo specifico e, superata la fase di inizializzazione dello stato, ogni frame viene validato tramite l'algoritmo *validate*, che classifica il frame attuale calcolando una percentuale. Il successivo stato è *ENTRANCE*, in cui ad ogni frame viene applicata la procedura *validate entrance*, che in caso di esito positivo porta il sistema nello stato *IN*, cioè il veicolo si trova nel tunnel. La procedura di validazione in questo caso è identica a quella che si ha

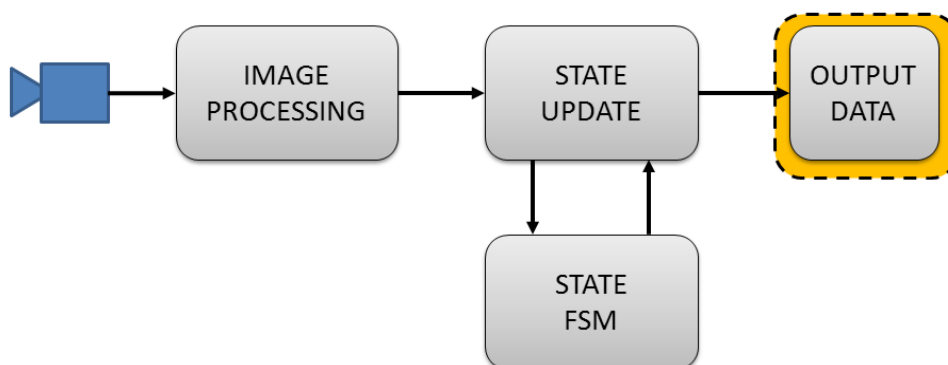


Figura 3.9: Struttura dell'algoritmo di riconoscimento di tunnel stradali

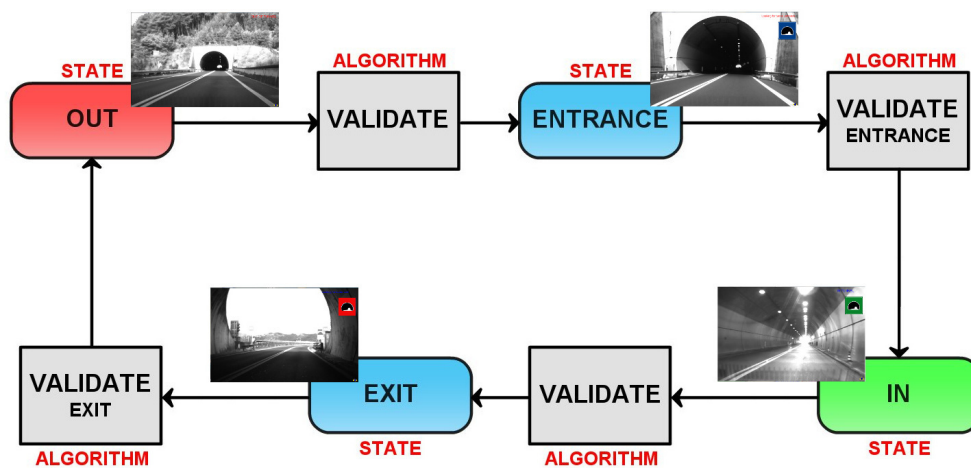


Figura 3.10: Macchina a stati finiti dell'algoritmo, con blocchi funzionali.

nello stato *OUT*, ma il sistema viene portato nello stato *EXIT*, nel quale viene eseguita una ulteriore validazione, *validate exit*, per decidere il cambiamento di stato. Lo stato successivo torna ad essere *OUT*, siccome il veicolo è uscito dalla galleria. La classificazione di ogni frame porta ad avere un valore numerico in percentuale, tramite il quale si può decidere se il cambiamento di stato può avvenire. Questa percentuale viene inserita in una coda con lunghezza fissa (ma diversa per alcuni stati), sulla quale viene calcolata la media aritmetica. Dunque, quando il sistema è a regime, la coda contiene le percentuali degli ultimi n frames: se il veicolo si stà avvicinando all'entrata o all'uscita di un tunnel le percentuali calcolate saranno crescenti e anche la media della coda nel tempo crescerà. Quando la media oltrepassa una certa soglia fissata per ogni stato si ha la transizione nello stato successivo. Questo procedimento permette di tenere traccia della potenziale entrata o uscita, mantenendo uno storico dei frame precedenti, e permette di ignorare alcune situazioni di falsi positivi che permangono sull'immagine per pochi frame.

3.5.1 Image Processing

Il blocco di *image processing* si occupa di acquisire l'immagine corrente dalla telecamera posta all'interno del veicolo in posizione frontale. Ogni immagine acquisita viene poi passata al blocco successivo per l'elaborazione.

3.5.2 State Update e State FSM

I due blocchi fondamentali dell'algoritmo sono mostrati in figura 3.10, in cui viene evidenziato lo stretto legame tra i quattro possibili stati del sistema e gli algoritmi che gestiscono le transizioni. Le prime operazioni da eseguire sull'immagine sfruttano i toni di grigio e riguardano:

- il calcolo degli istogrammi orizzontali e verticali, che contano il numero di pixel neri per individuare l'entrata di un tunnel nello stato *OUT*;
- il calcolo degli istogrammi orizzontali e verticali dei pixel bianchi per trovare l'uscita nel caso in cui il sistema si trovi nello stato *IN*;

Le soglie sul bianco e sul nero utilizzate sono calcolate dinamicamente per ogni frame e il procedimento è mostrato in seguito. Nel caso degli stati di transizione gli istogrammi considerati sono diversi:

- nello stato *ENTRANCE* la validazione viene fatta tramite un istogramma dei neri con soglia più alta, e tramite la valutazione dell'andamento temporale di una proprietà dell'istogramma, la varianza;
- nello stato *EXIT* il procedimento è più articolato e implica un ritaglio dell'immagine, un filtraggio preventivo della porzione ritagliata e una valutazione dell'istogramma dei bianchi ottenuto sull'immagine filtrata.

Setup dello stato iniziale

La procedura per la scelta dello stato iniziale è basata sulla luminosità dell'immagine, e in particolare sulla media aritmetica della luminosità nella parte superiore del frame. Questo valore viene memorizzato in una coda con lunghezza fissa n ; la media sulla coda viene sogliata per determinare lo stato iniziale. Quindi questo procedimento richiede inizialmente n frame, e successivamente l'applicazione esegue il normale flusso di lavoro partendo dallo stato iniziale. Gli unici valori possibili per questo stato sono gli stati stazionari, *IN* e *OUT*.

Soglie sul nero e sul bianco

Il procedimento per ottenere i valori di soglia con cui costruire gli istogrammi differisce nei casi di tonalità chiare o scure di grigio. Per i toni chiari la soglia è calcolata semplicemente come il valore massimo dei pixel nel frame corrente a cui viene sottratta una quantità molto bassa, definibile da file di configurazione e modificabile da pannello. Per questa applicazione, con immagini a toni di grigio nel vicino infrarosso, è sufficiente sottrarre 3 o 4 al valore massimo. La soglia sui toni medi e scuri è invece più critica, siccome le condizioni di luce e gli ambienti all'esterno di una galleria possono essere molto variabili, e potrebbero presentare oggetti di tonalità e luminosità simile all'ingresso della galleria. È stato definito un algoritmo per il calcolo della

soglia dinamica su ogni frame, che permette anche di avere una maggiore precisione nella sogliatura. Inizialmente vengono calcolati la luminosità media dell'immagine, come media aritmetica sui valori dei pixel, e il minimo valore dei pixel dell'immagine, con cui si può definire una percentuale P che rappresenta la differenza tra il pixel più scuro e la luminosità media, come mostrato nella formula:

$$P = \frac{|brightness_{average} - value_{Min}|}{255} \quad P \in [0, 1] \quad (3.10)$$

Quando la percentuale P è alta la soglia dovrà essere sensibilmente più alta del valore minimo, dato l'alto livello della luminosità media del frame e quindi, dato un valore $BlackSum$ da file di configurazione si ottiene una soglia dove la percentuale P pesa il valore da sommare al valore del pixel più scuro. Questa soglia va poi saturata per evitare di considerare oggetti troppo chiari o troppo scuri, la formula 3.11 mostra tutti i casi possibili:

$$blackThreshold = \begin{cases} th_{max} & \text{if } th \geq th_{max} \\ th_{min} & \text{if } th \leq th_{min} \\ value_{Min} + P \cdot blackSum & \text{otherwise} \end{cases} \quad (3.11)$$

Rilevazione ingresso e uscita

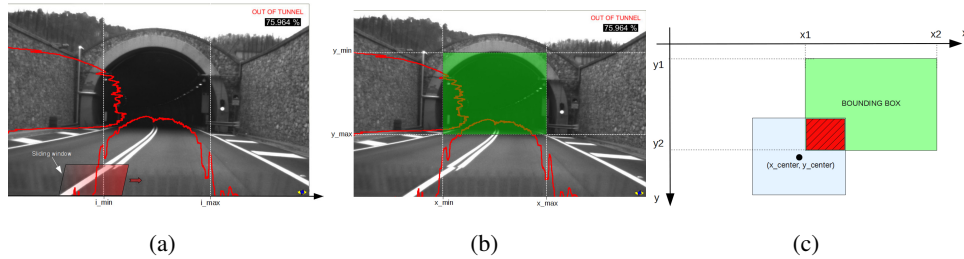


Figura 3.11: Rilevazione ingresso e uscita : (a) ricerca del picco degli istogrammi, (b) bounding box generato partendo dagli istogrammi orizzontale e verticale, (c) confronto tra bounding box e area condivisa per la centratura

Questa procedura di rilevazione utilizza lo stesso algoritmo sia per le tonalità chiare di grigio che per quelle scure. Questo viene utilizzato in modo duale con le

opportune soglie per calcolare gli istogrammi dell'immagine nei due casi. La rilevazione dell'ingresso (o dell'uscita) di un tunnel avviene tramite l'elaborazione degli istogrammi del frame attuale e il calcolo di un bounding box che delimita la zona dell'immagine in cui risiede l'entrata. Il box è rappresentato da quattro coordinate, (x_{min}, y_{min}) e (x_{max}, y_{max}) come mostrato in figura 3.11.(b) per il caso dei toni scuri, ricavate da una analisi sugli istogrammi per trovare la porzione principale e individuare i relativi limiti. Questi limiti forniscono degli indici sui vettori degli istogrammi che rappresentano, sul piano immagine, la zona dove con più probabilità si trovano i pixel scuri (chiari) e quindi dove si può localizzare l'uscita o entrata del tunnel. La procedura descritta in questo paragrafo viene eseguita solamente negli stati *OUT* e *IN*, di seguito vengono elencati i passi principali:

- calcolo del massimo e della porzione più grande dell'istogramma orizzontale;
- calcolo del massimo e della porzione più grande dell'istogramma verticale;
- calcolo del bounding box tramite le coordinate orizzontali e verticali;
- valutazione del bounding box;
- calcolo della percentuale di classificazione del bounding box.

Per quanto riguarda i primi due punti la stessa procedura viene applicata sui due istogrammi. Gli indici restituiti da questa procedura, i_{min} e i_{max} , sono gli estremi della porzione più significativa dell'istogramma e vengono usati nella successiva elaborazione, descritta dettagliatamente nell'algoritmo 1. Questi sono i punti di partenza per trovare l'estensione del picco dell'istogramma, che rappresenta una zona scura (chiara) dell'immagine e quindi una possibile entrata (uscita). L'istogramma viene percorso dall'inizio a i_{min} e da i_{max} fino all'ultimo indice, ricercando un differenza tra il valore della cella corrente e il valore massimo assoluto che sia maggiore di una certa soglia.

Algoritmo 1 Ricerca estensione picco massimo nell'istogramma**Require:** *ist* istogramma (orizzontale o verticale);**Require:** i_{min} primo indice della porzione di istogramma in cui si trova il picco;**Require:** i_{max} secondo indice della porzione di istogramma in cui si trova il picco;**Require:** *maxValue* valore del picco;

```

1: for  $i = i_{max}, \dots, ist.size$  do
2:   if ( $\| (ist[i] - maxValue) \| > validHistogram \cdot maxValue$ ) then
3:     // trovato un dislivello significativo rispetto al picco massimo
4:     // setto la seconda componente x del bounding box;
5:      $x_2 = i$ ;
6:   end if
7:   // controllo sui bordi dell'immagine per evitare che il box esca...;
8: end for
9: for  $i = i_{min}, \dots, 0$  do
10:  if ( $\| (ist[i] - maxValue) \| > validHistogram \cdot maxValue$ ) then
11:    // trovato un dislivello significativo rispetto al picco massimo
12:    // setto la prima componente x del bounding box;
13:     $x_1 = i$ ;
14:  end if
15:  // controllo sui bordi dell'immagine per evitare che il box esca...;
16: end for
17: // la stessa procedura può essere applicata sull'istogramma verticale per trovare
    $y_1$  e  $y_2$ 
18: return  $x_1, x_2$ 

```

I valori x_1 e x_2 , con y_1 e y_2 nel caso dell'istogramma verticale, permettono di individuare il bounding box che rappresenta con buona precisione l'entrata della galleria. Questo box ha delle proprietà quali la dimensione, il rapporto di forma e la centratura che permettono di discriminare tra i box meno significativi e quelli generati dall'entrata di un tunnel valido. Queste proprietà sono utilizzate per la procedura di valutazione e il calcolo della percentuale, come mostrato nell'algoritmo 2.

Algoritmo 2 Validazione frame**Require:** x_1, x_2 coordinate x del bounding box;**Require:** y_1, y_2 coordinate y del bounding box;**Require:** *oldBox* bounding box trovato nel frame precedente;

```

1: if ( $x_1 = x_2$  or  $y_1 = y_2$ ) then
2:   non è stato trovato un box;
3:   return 0.0;
4: end if
5:
6: // rapporto di forma:
7: if ( $box.ratio \geq 0.769$  and  $box.ratio \leq 2.5$ ) then
8:    $P_{filter} += 0.35$ ;
9:   crea un nuovo box  $\rightarrow box(x_1, x_2, y_1, y_2)$ 
10:
11: // area rispetto all'immagine:
12: if ( $box.area \geq c \cdot imageArea$ ) then
13:    $P_{filter} += 0.25$ ;
14: end if
15:
16: // area rispetto al box del frame precedente:
17: if ( $box.area \geq 0.8 \cdot oldbox.area$ ) then
18:    $P_{filter} += 0.4$ ;
19: end if
20:
21: // calcolo percentuali:
22:  $P_{center} = \frac{Area(box, centerBox)}{centerBox.area}$ 
23:  $P_{dimension} = \frac{box.area}{imageArea}$ 
24:  $P_{Validate} = \frac{P_{dimension} + 1.1 \cdot P_{filter} + 1.2 \cdot P_{center}}{3.3}$ 
25: return  $P_{Validate}$ 
26: end if
27: return 0.0

```


La funzione $Area(box, centerBox)$ calcola l'area condivisa tra due rettangoli: uno è il bounding box estratto dagli istogrammi mentre il secondo è un box centrato sull'immagine. La porzione di area tratteggiata in figura 3.11.(c) indica quindi una percentuale sull'area complessiva del box centrale, che andrà a comporre la percentuale finale di valutazione. Questo valore, P_{center} , fornisce una misura della centratura del bounding box, che dovrebbe aumentare mentre il veicolo si avvicina all'imboccatura del tunnel. Infatti l'entrata del tunnel sarà sempre più centrata nell'immagine fino a che il veicolo si troverà di fronte all'entrata, dove la centratura sarà massima, e quindi sarà massima l'area condivisa tra i due box. Di conseguenza si avrà una indicazione sempre più affidabile della presenza di una galleria all'aumentare della sovrapposizione tra i due box.

La percentuale finale è composta da tre contributi, la cui media pesata è normalizzata per essere considerata come una probabilità di classificazione, come mostrato nella formula 3.12.

$$P_{Validate} = \frac{P_{dimension} + 1.1 \cdot P_{filter} + 1.2 \cdot P_{center}}{3.3} \quad (3.12)$$

La percentuale finale è composta quindi da:

- $P_{dimension}$: percentuale relativa alla dimensione del box rispetto all'immagine;
- P_{filter} : percentuale relativa al filtraggio del box secondo alcuni filtri che aggiungono un contributo fisso: assume il valore massimo se il box soddisfa tutti i filtri con cui viene controllato;
- P_{center} : percentuale di centratura del bounding box, è massima quando il box è centrato nell'immagine.

P_{filter} a sua volta è dato dalla somma di alcune parti:

- una parte dovuta al rapporto di forma del bounding box; sono state eseguite alcune prove sperimentali per dimensionare questo filtro ed è stato verificato che il rapporto di forma deve essere compreso tra 0.769 e 2.5 per filtrare i box che non hanno forme simili all'entrata o uscita di una galleria;

- una parte data dalla dimensione del box; questa deve essere maggiore di una certa soglia che dipende dalla dimensione della finestra immagine, per riuscire a filtrare i box più piccoli;
- un contributo fisso sommato alla percentuale se il box nel frame corrente ha una dimensione maggiore dell'80 % della dimensione del box trovato al frame precedente. Questo permette di tenere traccia dell'ingrandimento del box che si ha quando il veicolo si avvicina all'entrata; quindi la percentuale risultante cresce con l'avvicinarsi all'entrata e con la persistenza del box nel frame.

Il valore della percentuale finale viene inserito nello storico delle percentuali, e viene calcolata la media degli ultimi n valori, dove n è la dimensione costante della coda FIFO utilizzata per memorizzare lo storico. Ad ogni frame la media corrente viene poi confrontata con una soglia. Il confronto decreta il passaggio di stato oppure la permanenza nello stato attuale. Nel caso degli stati *ENTRANCE* e *EXIT* la procedura descritta viene sostituita da due procedure più specifiche, di cui una è mostrata nei paragrafi relativi.

Individuazione picco massimo nell'istogramma

Questa procedura è necessaria per ottenere una indicazione della posizione del picco con valore massimo nell'istogramma, sia verticale che orizzontale. La procedura è descritta nell'algoritmo 3. Il vettore che costituisce l'istogramma viene diviso in 3 parti uguali, e ogni parte viene scansionata per mezzo di una media aritmetica calcolata su una finestra mobile che scorre sul vettore. Il valore massimo di questa media su tutto il vettore permette di identificare la parte dell'istogramma che contiene il picco, i cui estremi vengono calcolati (i_{min} e i_{max}). Durante la ricerca viene data priorità alla parte centrale, siccome nella maggioranza dei casi il veicolo inquadra l'entrata della galleria mentre si sta avvicinando e quindi è più probabile trovare l'entrata nella parte centrale dell'immagine. Di conseguenza, se la parte centrale dell'istogramma ha un valore massimo sopra una certa soglia, questa viene considerata sufficiente per gli scopi dell'algoritmo e viene immediatamente restituita, siccome contiene il picco cercato. La figura 3.11.(a) mostra un caso tipico dove l'istogramma viene analizzato

e la parte centrale viene restituita. Se la parte centrale non è predominante sulle altre due si passa a valutare i valori trovati per le tre parti e il maggiore determina la parte dell'istogramma che viene restituita, insieme al valore massimo raggiunto dalla media su finestra mobile. Questo metodo restituisce il limite inferiore e superiore della regione di interesse tramite gli indici i_{min} e i_{max} , insieme al valore massimo trovato nell'istogramma, oppure il valore dell'istogramma nella parte centrale se questa è sopra la soglia definita. Questi indici vengono poi usati per ricercare la massima dimensione del picco dell'istogramma che rappresenta l'entrata del tunnel.

Algoritmo 3 Ricerca picco massimo**Require:** *ist* istogramma;

```

1: // scansione della parte sinistra;
2: crea nuova finestra mobile  $\rightarrow winLeft$ 
3: for  $i = 0, \dots, imageWidth.size - winLeft.size$  do
4:   calcola valore medio sulla finestra che parte da  $i \rightarrow mean$ 
5:   if ( $mean > meanMax$ ) then
6:      $winLeft.i_{max} = i + winLeft.size$ 
7:      $winLeft.i_{min} = i$ 
8:      $winLeft.maxValue = mean$ 
9:   end if
10: end for
11:
12: // scansione della parte centrale;
13: // riempimento di  $winCenter$ ;
14:
15: // scansione della parte destra;
16: // riempimento di  $winRight$ ;
17:
18: if ( $winCenter.maxValue > meanThreshold$ ) then
19:   return  $winCenter$ 
20: end if
21: if ( $((winCenter.maxValue \geq 1.5 \cdot winLeft.maxValue) \text{ and } (winCenter.maxValue \geq 1.5 \cdot winRight.maxValue))$ ) then
22:   return  $winCenter$ 
23: end if
24: if ( $winLeft.maxValue > winRight.maxValue$ ) then
25:   return  $winLeft$ 
26: else
27:   return  $winRight$ 
28: end if

```

3.5.3 Validazione ingresso

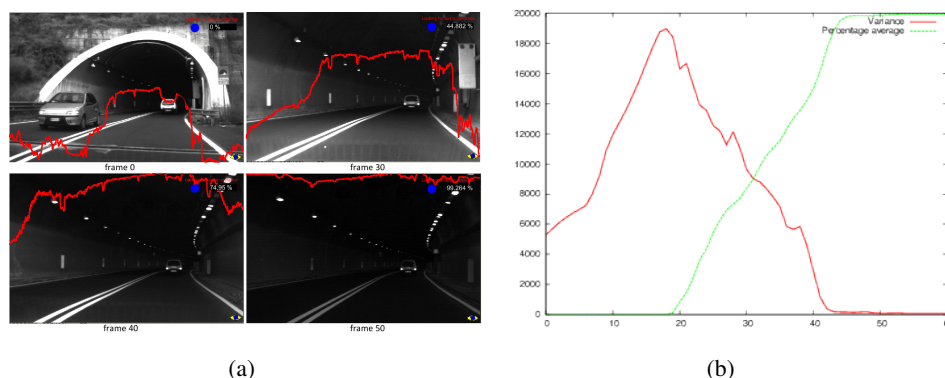


Figura 3.12: Validazione ingresso: (a) istogramma dei neri nel caso di entrata, frame più significativi, (b) varianza dell'istogramma dei neri e media delle percentuali sui frame

Il riconoscimento del passaggio all'interno del tunnel richiede una procedura di valutazione diversa dalle altre, che sfrutta le condizioni ambientali che si trovano tipicamente in una galleria. In particolare le condizioni di luce sono molto simili in tutte le gallerie, almeno di giorno, e considerando immagini a toni di grigio nel vicino infrarosso anche l'illuminazione della galleria non influisce molto sui grigi. Per questo è possibile calcolare l'istogramma orizzontale dei neri e analizzarne la forma frame dopo frame. La soglia sul nero utilizzata in questo istogramma è più alta rispetto agli altri casi, per avere più informazione contenuta nell'istogramma.

Tra le proprietà dell'istogramma quella più significativa in questo caso è la varianza, che ha un andamento preciso durante l'entrata in un tunnel.

Infatti, come mostrato in figura 3.12(a), l'istogramma si appiattisce quando il veicolo è nel tunnel, data l'uniformità dei colori e della luminosità, e questo permette di individuare il punto in cui la varianza scende sotto una certa soglia. La soglia è data da una percentuale sulla varianza massima calcolata per ogni frame da quando lo stato *ENTRANCE* è stato attivato. Infatti la varianza aumenta quando il veicolo si trova fuori e poi diminuisce fino a raggiungere un minimo locale quando il veicolo è appena

entrato nel tunnel. La figura 3.12.(b) illustra l'andamento della varianza nell'entrata di un tunnel: il frame 0 indica l'attivazione dello stato *ENTRANCE*, mentre il veicolo è entrato completamente nel tunnel dal frame 40 in poi.

Quando la varianza scende sotto la soglia le percentuali restituite per ogni frame diventano più alte, e vengono inserite in una coda che tiene traccia dei valori nel tempo, seguendo lo stesso meccanismo usato negli altri stati. Se la media della coda supera una soglia predefinita in un certo frame, quel frame determina il passaggio dallo stato *ENTRANCE* allo stato *IN*. In verde in figura 3.12.(b) è mostrata la media riscalata delle percentuali, che consente di decidere il frame in cui avviene il cambiamento di stato, in questo caso il frame 35. La percentuale di valutazione in questo caso è calcolata nel modo seguente:

$$P = 1.0 - \frac{var}{varMax} \quad (3.13)$$

dove *varMax* rappresenta la massima varianza calcolata su tutti i frame, dal momento dell'attivazione dello stato *ENTRANCE* fino al frame attuale. In questo modo quando la varianza diminuisce la media delle percentuali aumenta fino ad oltrepassare la soglia prestabilita e si ha il passaggio di stato del sistema.

3.5.4 Validazione uscita

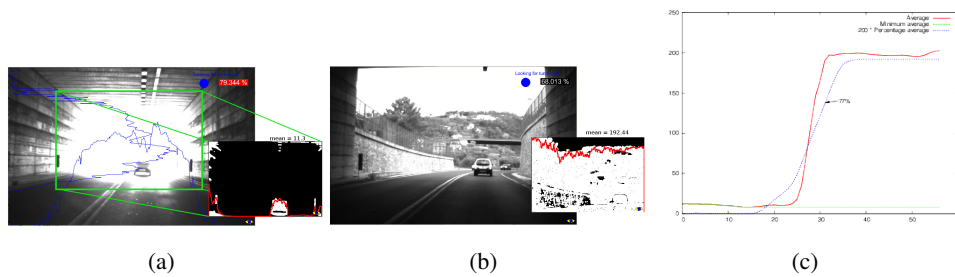


Figura 3.13: Validazione uscita: (a) bounding box filtrato e sogliaato prima dell'uscita da un tunnel, (b) bounding box filtrato e sogliaato all'uscita da un tunnel, (c) andamento della media dell'istogramma all'uscita da un tunnel

La procedura di validazione dell'uscita da un tunnel richiede alcuni accorgimenti che permettono di riconoscere la situazione cercata nel più alto numero di casi possibile. Un istogramma sull'immagine non è più sufficiente per discriminare il frame di uscita con sufficiente precisione: infatti le condizioni ambientali e luminose che si trovano all'esterno del tunnel non sono facilmente prevedibili, dunque le proprietà dell'istogramma non hanno un andamento ben definito e non possono essere utilizzate da sole per verificare la posizione del veicolo. Quindi la procedura si articola in alcune fasi:

- individuazione del bounding box che rappresenta l'uscita;
- filtraggio dell'immagine contenuta nel bounding box, con filtro di Sobel;
- sogliatura dell'immagine filtrata;
- costruzione dell'istogramma sull'immagine filtrata e calcolo della media;
- validazione dello stato tramite controllo sulla media.

Filtraggio e sogliatura

Durante lo stato *IN* viene calcolato un bounding box per verificare la presenza di un'uscita dal tunnel. Quando si ha la transizione di stato *EXIT* la posizione del box viene mantenuta, mentre le dimensioni vengono ridotte di una quantità costante per cercare di "inquadrare" esattamente la porzione dell'immagine in cui è presente l'uscita. Questa parte di immagine viene ritagliata e costituisce una nuova immagine. Questa viene a sua volta filtrata con un filtro di Sobel con finestra di dimensione 3×3 , che evidenzia i bordi contenuti nell'immagine. Il risultato del filtro viene soglia con una soglia binaria molto bassa, in modo che si ottengano più bordi possibile in uscita tra tutti quelli contenuti nel risultato del filtro di Sobel. Su quest'ultima immagine viene calcolato l'istogramma orizzontale dei punti bianchi, che verrà poi usato per discriminare la situazione corrente del veicolo, cioè se si trova completamente dentro al tunnel oppure sta per uscire. Questo procedimento porta ad avere un'immagine

simile a quella mostrata in figura 3.13.(a): quando il veicolo si trova ancora nel tunnel il bagliore dato dalla luminosità dell'uscita porta a saturare la camera e il colore risultante è uniforme. Quindi nell'immagine filtrata saranno presenti pochi contorni. In figura 3.13.(b) è mostrata invece la stessa porzione di immagine ma filtrata quando il veicolo si trova ormai in prossimità dell'uscita. Qui si notano molti più contorni e quindi l'immagine risultante è molto diversa.

Infatti la media dell'istogramma dei bianchi è molto più alta e può essere usata come parametro per riconoscere la situazione attuale, in confronto con il valore che assumeva precedentemente. Questo genera un valore percentuale che rappresenta la classificazione del frame attuale come frame interno oppure esterno al tunnel, e permette di individuare il frame in cui scatta la transizione di stato.

Valutazione istogramma

L'ultima fase del passaggio di stato verso l'uscita del tunnel consiste nella valutazione dell'istogramma calcolato in precedenza, attraverso la sua media. Per questo è utile rappresentare l'andamento del valor medio dell'istogramma durante la transizione verso l'esterno del tunnel. La figura 3.13.(c) mostra l'andamento crescente, che ha un massimo nel punto in cui il veicolo è vicino all'uscita (circa al frame 30 in questo caso).

Ad ogni frame il valore corrente della media viene confrontato con il valore minimo calcolato fino al frame attuale, e da qui si ottiene la percentuale come:

$$P = \frac{average - average_{min}}{average} \quad (3.14)$$

Questo valore viene inserito in una coda di percentuali, la cui media determina il passaggio di stato tramite confronto con una soglia predefinita. L'algoritmo 4 illustra brevemente la procedura utilizzata.

Algoritmo 4 Validazione istogramma nello stato *EXIT***Require:** istogramma;

```

1: Per ogni frame:
2: calcolo della media dell'istogramma  $\rightarrow average$ 
3: if il frame attuale è il primo nello stato EXIT then
4:    $average_{min} = average$ 
5:   return 0.0
6: end if
7: if ( $average < average_{min}$ ) then
8:    $average_{min} = average$ 
9: end if
10:  $P = \frac{average - average_{min}}{average}$ 
11: return  $P$ 

```

3.5.5 Risultati ottenuti

Sequence	% Correct Detection	Transitions	False Positives [frames]
first	100.0	80	13
second	83.3	48	0
third	100.0	32	350
total	94.4	160	363

Tabella 3.1: Risultati dell'algoritmo.

Il sistema è stato testato con un calcolatore con le seguenti caratteristiche:

- Cpu: Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz;
- Ram memory: 8 GiB DIMM DDR2 Synchronous 800 MHz (1.2 ns);
- Video Card: GeForce 9800 GTX/9800 GTX+ 512 MB memory.

Sono state utilizzate tre differenti sequenze nei test, con un totale di 54686 frame ad una frequenza di 21 FPS. Le sequenze sono state acquisite durante il giorno e contengono molti tunnel di lunghezze varie, con molte entrate ed uscite dagli stessi in condizioni di traffico normale. I tunnel hanno entrate ed uscite con differenti forme, rettangolari e circolari, e si trovano in ambito extraurbano, autostradale e anche urbano. Le immagini sono state acquisite con una telecamera con sensibilità nello spettro del visibile con coda NIR, non calibrata, monocromatica e con guadagno variabile. Il sistema ha dato prova di essere in grado di rilevare correttamente il proprio stato con una percentuale correct detection del 94,4 %, come mostrato in tabella 3.1. I casi di miss-detection sono apparsi nel caso di tunnel in ambito urbano, in questa situazione il tunnel si trova nei pressi di alti palazzi con strade strette e con curve. In sequenze con la presenza di alberi a bordo strada e sole contro la telecamera sono stati riscontrati alcuni falsi positivi.

Un parametro fondamentale per giudicare le performance di questo algoritmo è il tempo di esecuzione, infatti per informare gli altri processi che si occupano del funzionamento degli ADAS si devono rispettare tempistiche hard real-time. Il sistema presenta tempi medi di esecuzione di 750 μ secondi, su un calcolatore di media potenza indicato in precedenza, restando sempre sotto 1 ms di tempo massimo di esecuzione; quindi si può ritenere soddisfatto il vincolo imposto.

3.6 Sistema per il riconoscimento e la classificazione di ambienti stradali

In questa sezione verrà illustrato un approccio per lo sviluppo di un sistema di visione artificiale per la classificazione delle immagini acquisite da telecamera fissa a bordo di un veicolo, non calibrata, in tre classi di scenari in ambiente stradale, mediante tre differenti algoritmi di apprendimento automatico.

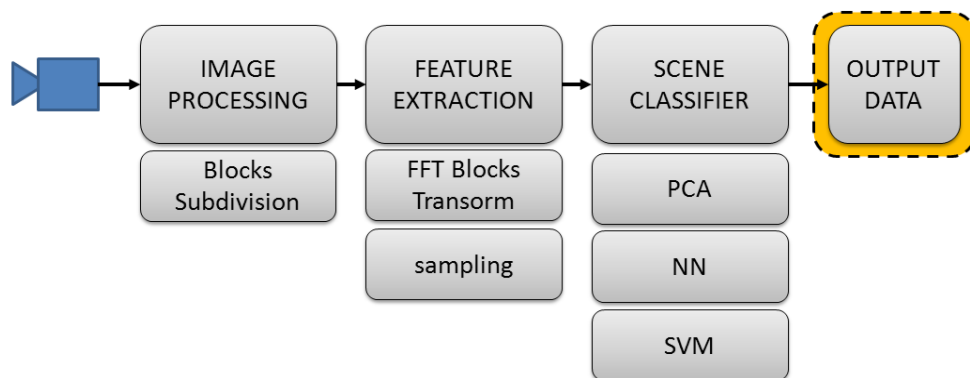


Figura 3.14: Struttura dell'algoritmo per la classificazione di ambienti stradali.

3.6.1 Algoritmo generale

La figura 3.14 mostra lo schema a blocchi del progetto. Il sistema di classificazione delle immagini sviluppato in questa tesi è stato progettato per avere uno strato comune di estrazione delle caratteristiche dominanti in ciascuna immagine. Questo livello di *pre-processing* si occupa di elaborare l'immagine acquisita, tramite diversi filtri e modifiche dell'istogramma della scala di grigi, ed è seguito da un algoritmo di *features extraction* che trasforma le aree dell'immagine nel dominio della frequenza e cattura diversi campioni che costituiranno i dati su cui lavoreranno tutti i livelli successivi.

Dopo il primo strato sono stati realizzati tre classificatori che lavorano in parallelo sui campioni: il primo utilizza la PCA e le pdf gaussiane, il secondo le reti neurali e il terzo le SVM. Ciascuno deve essere in grado di produrre un risultato, assegnando all'immagine di input una delle tre classi fra cui si vuole discriminare.

3.6.2 Scelta delle classi e training set

Il progetto prevede la distinzione tra tre differenti classi di ambienti stradali: città, autostrada e extraurbano. I tre ambienti, visibili in figura 3.15 hanno le seguenti

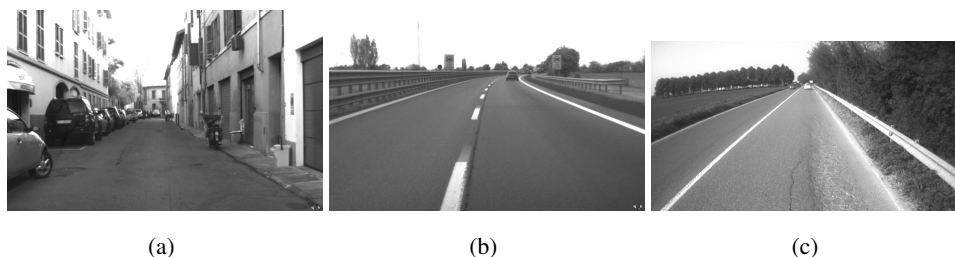


Figura 3.15: Classi di scenari in ambiente automotive: (a) città, (b) autostrada, (c) extraurbano.

caratteristiche:

- Città: è un ambiente con strade asfaltate, una sola carreggiata e di ampiezza limitata, ad una o due corsie, sempre con palazzi ai lati, dove sono presenti bivi, incroci, altri veicoli, pedoni e motocicli, figura 3.15.(a).
- Extraurbano: si tratta di strade di campagna, asfaltate ma non più grandi di due corsie, con alberi e prati ai lati e la possibilità che ci siano campi o erba alta, figura 3.15.(b).
- Autostrada: una tipica immagine di autostrada comprende due carreggiate da almeno due corsie ciascuna, con guard-rail che la dividono, campi o alberi radi fuori strada. Si preferisce evitare che ci siano sovrappassi che occludono il cielo o un alta vegetazione ai lati, figura 3.15.(c).

Il training set è formato da immagini significative delle classi indicate. Sono state raccolte un totale di 6949 immagini, ogni immagini differisce nel tempo di almeno 1 secondo dalla precedente, per evitare di avere immagini troppo simili. In particolare nel training set compaiono 2698 frame della classe città, 2046 della classe autostrada e 2205 della classe extraurbano.

3.6.3 Image Processing

L'input dell'applicazione è un immagine in scala di grigi o a colori, di qualsiasi dimensione proveniente da una telecamera inserita all'interno di un veicolo in posizione

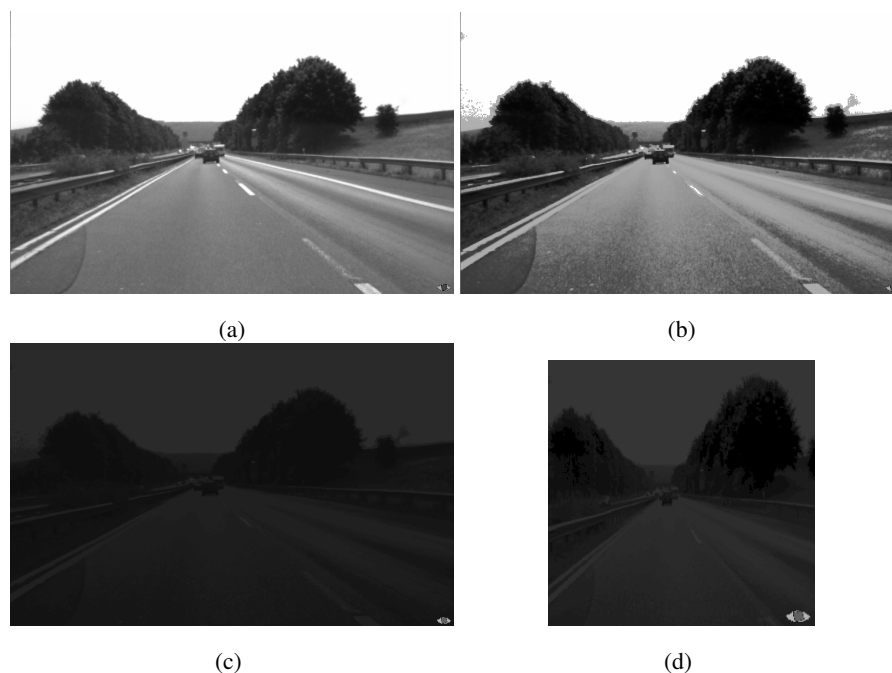


Figura 3.16: Image processing: (a) frame input, (b) frame con istogramma normalizzato, (c) frame filtrato con filtro passa-alto, (d) immagine ridimensionata.

frontale.

La prima operazione svolta nel blocco di processing consente nella conversione dell'immagine di input in scala di grigi, con valore del singolo pixel a 8 bit. In figura 3.16.(a) è illustrato un frame acquisito dalla telecamera convertito in scala di grigi.

Normalizzazione dell'istogramma

Una volta convertita l'immagine si passa alla normalizzazione dell'istogramma dei livelli di grigio. L'istogramma indica l'escursione della luminosità dell'intera immagine, evidenziando la presenza di zone molto scure o molto chiare oppure informa sul buon bilanciamento dell'immagine. Questo procedimento permette di ottene-

re un maggior contrasto con un istogramma uniforme, permettendo di espandere la dinamica dei livelli di grigio. Il risultato di questo blocco è mostrato in figura 3.16.(b).

Filtro passa-alto

Allo scopo di attenuare le piccole differenze di contrasto si applica un filtro la cui risposta in frequenza è data dalla funzione:

$$H(k,l) = \begin{cases} 1 & \text{se } k = 0 \cap l = 0 \\ 1 - (1 - 0.9e^{-\frac{k^2+l^2}{156.25}}) & \text{altrimenti} \end{cases} \quad (3.15)$$

Dove con $H(k,l)$ si intende la posizione del pixel nell'immagine. Il filtro porta ad una intensità uniformemente distribuita su tutta l'immagine, come mostrato in figura 3.16.(c).

Ridimensionamento e suddivisione dell'immagine

L'immagine viene ridimensionata a 256×256 pixel; questa operazione potrebbe comportare anche un ricampionamento, in figura 3.16.(d) si vede il risultato.

Blocks Subdivision

Il passo successivo al ridimensionamento consiste nel suddividere l'immagine in 16 sotto-parti equamente dimensionate, in blocchi di 64×64 pixel, come mostrato in figura 3.17.(a). La divisione viene fatta in modo da poter ottenere informazioni sui valori di ampiezza nel dominio della frequenza in certe regioni dell'immagine. Con questa operazione si conclude lo stadio di processing dell'immagine acquisita dalla telecamera mobile a bordo del veicolo.

3.6.4 FFT e campionamento

Un blocco essenziale per l'algoritmo è l'applicazione della trasformata veloce di Fourier, FFT (Fast Fourier Transform). Questa operazione viene effettuata a ciascuna

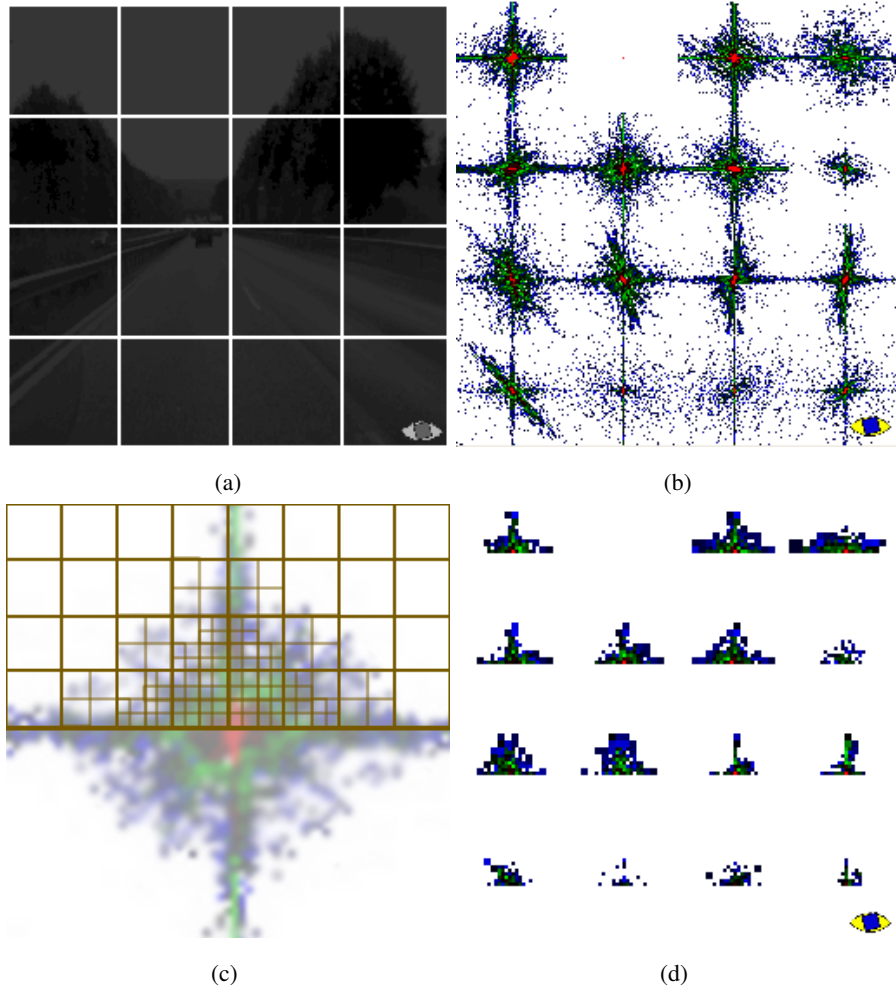


Figura 3.17: Elaborazione nel dominio della frequenza: (a) suddivisione in blocchi, (b) FFT per ogni blocco, (c) maschera per filtro di campionamento, (d) FFT campionata per ogni blocco.

delle 16 sottoparti e permette di descriverle come funzioni complesse, separate in ampiezza e fase come illustrato dall'equazione:

$$F_i(k, l) = |F_i(k, l)|e^{j\phi(k, l)} \quad (3.16)$$

La formula 3.16 rappresenta la trasformazione di un valore di coordinate k e l in ampiezza e fase. La figura 3.17.(b) mostra come gli spettri di ampiezza della FFT per l'immagine in input. Si nota che nelle aree in cui, grazie anche alla pre-elaborazione, non ci sono elementi inquadrati o cambiamenti di contrasto, come per esempio nel cielo, la FFT è nulla e l'immagine è bianca. Nelle regioni in cui si passa dalla cima degli alberi al cielo il modulo della trasformata prende una forma a croce più o meno marcata, per evidenziare che il maggior cambiamento avviene lungo i due assi.

Si useranno queste sedici trasformate per descrivere l'immagine. Per estrarre l'informazione rilevante, ciascuno spettro viene gestito in modo indipendente. Poiché l'obiettivo è l'implementazione in tempo reale, campionando ognuno dei 64×64 valori dello spettro si otterrebbe una descrizione di dimensioni troppo elevate ed inutilizzabile per una classificazione diretta, inoltre una così grande quantità di dati potrebbe causare un possibile *overfitting* del *training set*, che interferirebbe sulla ricerca di una buona generalizzazione della classificazione. In più la conoscenza del valore di ogni singola ampiezza ad una certa frequenza non è di interesse, poiché il cambiamento di posizione di un oggetto nell'immagine può portare a piccoli cambiamenti dei livelli di energia della spettro. Pertanto si valuterà la distribuzione di una certa area dello spettro invece di ogni singolo valore di ampiezza. Per fare questo ciascuno spettro è filtrato con un *array* di 128 maschere, che vanno a ricoprire solo la parte superiore dell'immagine, dato che la trasformata è simmetrica. Le maschere sono dimensionate in modo da avere bassa risoluzione ad alte frequenze ed alta risoluzione a basse frequenze, perchè, nel caso di ambienti stradali, è proprio intorno all'origine la più alta concentrazione di informazione, mentre ci sono poche variazioni ad alte frequenze. Si usa una maschera di 8×8 pixel per le parti più esterne, una 4×4 nella zona intermedia, e una 2×2 in quella centrale, come esemplificato in figura 3.17.(c). I valori che riempiono queste maschere vengono ordinati in modo crescente e come caratteristica dominante viene salvato il valore che si trova in posizione centrale all'array ordinato, nello stesso modo in cui si applicherebbe un filtro mediano. In figura 3.17.(d) viene mostrato come potrebbe apparire una immagine costituita dai risultati del campionamento.

I 128 campioni ottenuti andranno a costituire le *features* della sottoparte dell'im-

immagine da cui sono stati estratti: l'*output* della *features extraction* sono 16 insiemi di 128 campioni.

3.6.5 Scene Classifier

In questa sezione si analizza lo stadio dell'algoritmo responsabile della classificazione dell'immagine corrente. Per la classificazione si utilizzano tre differenti tecniche illustrate nelle sezioni successive. Il classificatore baserà la propria scelta confrontando le ipotesi ottenute dai classificatori.

3.6.6 Classificazione mediante PCA

Questa sezione descrive il metodo di classificazione della scena con un approccio che utilizza la PCA. Inizialmente, fuori linea, si effettua una specie di apprendimento, per capire come assegnare i voti in fase di esecuzione. Dal training set sono state estratte 16 matrici, ognuna indica ogni sottoblocco delle immagini dell'insieme, con numero di righe pari al numero di frame e 128 colonne, pari al numero delle feature. Su ciascuna di queste matrici sarà applicato lo stesso procedimento.

Tramite la SVD, una matrice M , $M \in R^{n_frame \times 128}$ viene scomposta in tre componenti, secondo la formula $M = U\Sigma V^*$. In Σ , $\Sigma \in R^{n_frame \times 128}$ sono contenuti gli autovalori della matrice in ordine decrescente, e ciascuno di questi autovalori è legato ad un autovettore che compone una colonna di V^* , $V^* \in R^{128 \times 128}$. Visualizzando gli autovalori si possono individuare quali sono i primi l dal valore più elevato e distintivi per l'insieme, e se ne deduce che i primi l autovettori contengono la maggior parte dell'informazione per quella matrice M . Si realizza, allora, una matrice che ha per colonne questi l autovettori e che sarà la base di trasformazione per la rotazione degli assi secondo la PCA. In ognuna delle 16 matrici M , si moltiplica ciascuna riga per gli autovettori scelti, ottenendo l scalari per riga. La trasformazione adottata è del tipo $R^{#FRAME \times 128} \rightarrow R^{#FRAME \times l}$. Gli l valori, risultato del prodotto riga per colonne, saranno le coordinate $[x, y, z, \dots]$ che mappano ciascun vettore nel nuovo spazio della PCA (o spazio delle *features*). Si ottiene così un insieme di punti in l dimensioni,

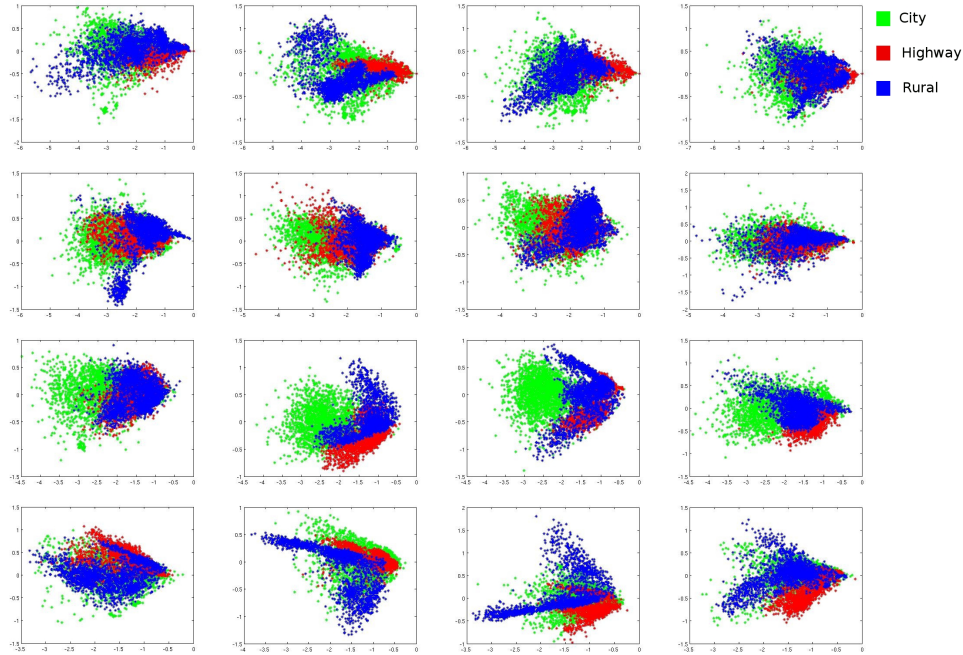


Figura 3.18: Trasformazione dei vettori di due classi tramite PCA

che caratterizza la classe di scenari per cui sono stati selezionati i frame e costruite le matrici M .

Operando tale trasformazione per classi di scenari differenti si generano tante nuvole di punti quante sono le classi scelte, ripetuto per le sedici sottoparti dell'immagine. In figura 3.18 viene mostrato il risultato di una trasformazione in 2D su insiemi di frame selezionati per le classi “città”, “extraurbano” e “autostrada”.

L'obiettivo della PCA è quello di ottenere delle regioni ad alta distribuzione di punti in modo da poterle dividere facilmente con una funzione di primo o secondo grado, salvare questa funzione e usarla come discriminante durante l'esecuzione della classificazione (predizione) di immagini sconosciute. Come funzione caratterizzante ciascuna nuvola di punti si sceglie di usare la distribuzione di probabilità (*pdf*) gaussiana in modo tale che, conoscendo media e varianza dei punti di una classe, si possa

calcolare durante la predizione quale sia la probabilità che il frame da classificare faccia parte di quella classe.

La formula della funzione di distribuzione gaussiana (o normale) l -dimensionale è riassunta in 3.17, e dipende da media (μ) e deviazione standard (σ) per ogni dimensione:

$$f(x, y, \dots, l) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{1}{2}[(\frac{x-\mu_x}{\sigma_x})^2]} \cdot \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{1}{2}[(\frac{y-\mu_y}{\sigma_y})^2]} \cdot \dots \cdot \frac{1}{\sqrt{2\pi}\sigma_l} e^{-\frac{1}{2}[(\frac{l-\mu_l}{\sigma_l})^2]}; \quad (3.17)$$

Terminata questa prima fase, che può essere paragonata all'addestramento di un classificatore, devono essere salvati i due autovettori scelti e media e varianza per ogni classe di scenario.

Durante la fase di esecuzione del classificatore i dati di ingresso sono i 16 vettori da 128 campioni di un frame. Per calcolare la probabilità che un vettore di quei 16 appartenga ad una categoria di scenari c , questo deve essere prima di tutto trasformato nello spazio delle *features*, moltiplicandolo per la matrice di trasformazione avente per colonne gli autovettori di quella classe c . Successivamente si applicherà la gaussiana multidimensionale con i valori di media e varianza di c al punto in cui è stato trasformato il vettore. Ripetendo questo procedimento che interessa un vettore, per tutte le tipologie di scenari fra cui si vuole discernere, si otterranno la probabilità con cui l'area a cui si riferisce il vettore appartiene a ciascuna classe.

Dopo aver eseguito questo passaggio per tutti i 16 vettori, si ottengono le n probabilità per le n classi. Come indicato in formula 3.18, sommando tra loro, e poi dividendo per 16, le 16 probabilità della prima classe, le 16 della seconda, le 16 della terza e via seguendo. si arriva a determinare quali sono le probabilità che l'intera immagine appartenga alla prima classe, alla seconda, alla terza, o alle seguenti. Naturalmente vincerà la categoria con probabilità, sull'intera immagine, più alta, seguendo l'indicazione della disuguaglianza 3.19.

$$Se P_{tot}[n] = \frac{\sum_{i=1}^{16} P_i}{16} \quad (3.18)$$

$$P_{tot}[a] > P_{tot}[b] > P_{tot}[c] \dots \rightarrow frame \in Classe[a]. \quad (3.19)$$

Si prenda come esempio di avere 3 classi di scenari, di cui precedentemente si siano calcolati autovettori, media e varianza. Si possono ora sostituire le loro medie e varianze nella funzione della pdf 3.17 per ottenere 3 funzioni che dipendono solo da $x, y, \dots l$. Si trasformi ciascun vettore di ingresso con gli autovettori della prima classe, ottenendo 16 punti di l -dimensioni. A ciascuno di questi punti va applicata la gaussiana relativa alla prima classe e si ottengono 16 probabilità, che sommate tra loro, e poi divise per 16, danno la probabilità totale che l'immagine appartenga alla prima classe. Ripetendo il procedimento per le altre due classi si ottengono anche le altre due probabilità totali. La più alta assegnerà l'etichetta all'immagine

3.6.7 Classificazione mediante Neural Networks

In questa sezione si illustra brevemente l'approccio di classificazione utilizzato per discriminare tra le classi di scenari basato su reti neurali. Per i classificatori basati su reti neurali è fondamentale una fase di addestramento che è basata sul training set. In questa fase viene individuato il numero di reti da utilizzare, la struttura per collegarle, e in un secondo passo, il numero di strati nascosti e di neuroni e l'algoritmo di apprendimento di ciascuna rete. Per realizzare il classificatore basato su reti neurali sono state utilizzate le librerie LIBFANN che permettono di utilizzare 4 diverse tipologie di addestramento:

- INCREMENTAL;
- BATCH;
- RPROP;
- QUICKPROP.

Per trovare il miglior algoritmo per il problema della classificazione di scenari si sono eseguiti, sul training set, ad uno ad uno, tutti gli algoritmi disponibili, per un numero medio di epoch uguale per tutti (2000), e si è valutato quello che portava ad un migliore errore in termini di MSE.

Le tabelle in 3.2 riassumono questi risultati ed è evidente come il migliore algoritmo sia l'incrementale.

epoch	<i>INCREMENTAL</i>		<i>BATCH</i>	
	mse	errori	mse	errori
1	0.000621	56	0.061464	6949
250	0.000497	43	0.022633	491
500	0.000498	42	0.019897	513
750	0.000476	42	0.022164	810
1000	0.000450	39	0.018188	587
1250	0.000451	39	0.017214	575
1500	0.000456	41	0.016268	555
1750	0.000454	39	0.015471	527
2000	0.000449	40	0.014926	520

epoch	<i>RPROP</i>		<i>QUICKPROP</i>	
	mse	errori	mse	errori
1	0.061464	6949	0.061464	6949
250	0.010574	467	0.020416	624
500	0.009272	402	0.017569	586
750	0.008780	386	0.015893	471
1000	0.008393	362	0.014807	444
1250	0.008180	359	0.014734	474
1500	0.008061	357	0.013888	447
1750	0.007978	360	0.013630	456
2000	0.007919	355	0.013301	447

Tabella 3.2: Addestramento con diverse funzioni di apprendimento

Con la stessa rete, stesso insieme di campioni e con il miglior metodo di addestramento trovato, si è proceduto a verificare anche quali fossero le migliori funzioni di attivazione per i neuroni degli strati nascosti e per i neuroni di output.

Per ogni prova sono state scelte una funzione per gli strati nascosti e una per gli output e si è eseguita la rete per un breve, ma sufficiente periodo di 2000 epoch. Alcune delle combinazioni sono state in grado di raggiungere ottimi obiettivi, arrivando fino ad un MSE di circa 0.0001, come quella sigmoide/sigmoide o sigmoide/sigmoide a gradino, mentre altre non sono riuscite in quel periodo di tempo ad andare sotto l'MSE di 0.004 come è successo per le lineari e le gaussiane simmetriche.

Considerando i risultati ottenuti si è optato per utilizzare come funzioni di attivazione quella sigmoide per gli strati nascosti e quella sigmoide a gradino per i neuroni di uscita, questo per tutte e 17 le reti.

3.6.8 Classificazione mediante SVM

In questa sezione verrà descritto il blocco relativo al classificatore basato su support vector machines.

Il principale problema nel costruire una SVM è decidere quale kernel utilizzare e con quali parametri. La procedura solitamente usata da chi per la prima volta utilizza questi strumenti è quella di:

- trasformare i dati nel formato richiesto dalle SVM;
- scelta di kernel e parametri;
- serie di test per verificare se i parametri scelti possono andare bene, e nel caso negativo ripetizione dal punto precedente.

Per realizzare il classificatore basato su SVM sono state utilizzate le librerie LIB-SVM [27] con kernel RBF. La procedura consigliata nel manuale delle librerie LIB-SVM è la seguente:

- trasformare i dati nel formato richiesto dalle SVM;
- normalizzare o scalare i dati;

- utilizzare un kernel RBF;
- cross-validation e grid search per determinare i parametri C e γ ;
- utilizzare i parametri C e γ per l'addestramento sul training set completo;
- test.

Per addestrare le SVM per la classificazione di scenari si sono eseguite una serie di grid-search con 10-fold cross-validation, sui campioni per le 16 aree delle immagini. I valori migliori trovati sono stati $C = 8$ e $\gamma = 0.5$ per una accuratezza fino al 95%.

3.6.9 Tracking

Il sistema opera su ogni frame per realizzare la classificazione, senza avere nessuna retroazione sulle decisioni effettuate sulle ipotesi. Per irrobustire e stabilizzare il comportamento del sistema di classificazione degli scenari si è utilizzata una tecnica base di tracking dei valori di classificazione ai frame precedenti.

Per evitare oscillazioni tra classi differenti si tiene traccia di un numero arbitrario di classificazioni precedenti, se si nota che nella coda la classificazione è cambiata in modo significativo allora viene cambiata la classificazione al frame attuale.

Con il test set utilizzato sono stati sufficienti 5 frame di isteresi per avere una classificazione stabile. In questo modo si possono avere delle classificazioni in ritardo, bisogna tener conto però che con frame rate di 12 FPS attendere 5 frame significa attendere 0.4 secondi per avere una decisione stabile.

3.6.10 Risultati ottenuti

Per l'analisi dei risultati si è fatto riferimento ad un test set composto da 19231 immagini.

La tabella 3.3 riassume il numero di frame per ogni classe e il frame rate di ogni sequenza. Le sequenze sono state annotate da un operatore umano per realizzare un ground truth preciso.

Classe	numero frame	frame rate(FPS)
Città	7239	10
Autostrada	8492	15
Extraurbano	3500	12

Tabella 3.3: Composizione del test set

Metodo	Classificazioni corrette		
	Città	Autostrada	Extraurbano
PCA	95.6%	96.9%	7.0%
NN	91.6%	100.0%	0.0%
SVM	100.0%	84.4%	95.0%

Tabella 3.4: *Correct Detection* dei classificatori su tutto il test set.

La tabella 3.4 illustra le correct detection in termini percentuali su tutto il test set. Il sistema di riconoscimento di scenari in ambiente automotive sviluppato nell'ambito di ricerca di questo dottorato ottiene risultati in classificazione molto buoni per quanto riguarda la discriminazione tra le classi Città e Autostrada. Relativamente alla classe di scenari Extraurbano i risultati sono molto buoni solamente nel caso in cui si utilizzino le SVM, con gli altri classificatori, infatti, i risultati risultano essere non soddisfacenti.

In questa sezione sono analizzate anche le matrici di confusione relative ai singoli classificatori implementati per analizzare a fondo le motivazioni nel caso di errore.

	Città	Autostrada	Extraurbano
Città	95.6%	4.0%	0.4%
Autostrada	2.4%	96.9%	0.7%
Extraurbano	64.0%	29.0%	7.0%

Tabella 3.5: Matrice di confusione per il classificatore con PCA.

Nella tabella 3.5 viene illustrata la matrice di confusione per il classificatore basato su PCA. Il classificatore realizzato tramite PCA, in ambiente cittadino ha una

3.6. Sistema per il riconoscimento e la classificazione di ambienti stradali 101

percentuale di predizioni corrette del 95.6%, sbagliando solo 318 frame su 6921, 290 dei quali etichettati come autostrada e 28 come extraurbano. In autostrada il rilevamento migliora arrivando a una percentuale di correttezza del 97%. Su 8492 immagini, 207 sono state etichettate come città e 55 come rurale.

Lo scenario in cui questa tecnica risulta più debole è l'extraurbano. Su 3500 frame solo 241 sono rilevati correttamente, per una percentuale del 6.9%.

La causa dell'assegnazione delle immagini extraurbane alla classe di città è da ricercarsi nella frequente presenza di alberi ad alto fusto che possono modificare drasticamente la FFT delle parti alte dell'immagine facendo credere al classificatore di trovarsi in presenza di palazzi, come accade nel caso in figura 3.22.(a). Il metodo con PCA e pdf gaussiana non è stato in grado di trovare un modo per separare ottimamente le *features* per riconoscere le caratteristiche degli scenari extraurbani; l'unico caso in cui la predizione è corretta è quando viene individuato sull'immagine un ostacolo isolato, nella parte sinistra o destra del frame, come l'albero che si vede in figura 3.22.(d). Questo è una particolarità che non avviene in città dove vi sono le infrastrutture a coprire tutto o in autostrada dove, se ci sono, gli alberi sono più lontani. Tutto sommato il classificatore è riuscito a rilevare una peculiarità di questo paesaggio da utilizzare per classificare correttamente.

	Città	Autostrada	Extraurbano
Città	91.6%	8.4%	0.0%
Autostrada	0.0%	100.0%	0.0%
Extraurbano	0.0%	100.0%	0.0%

Tabella 3.6: Matrice di confusione per il classificatore con Reti Neurali.

Le reti neurali hanno una precisione leggermente più scarsa in città, dove predicono correttamente il 91.6% dei frame, sbagliandone 604 (l'8.4%) in favore dell'autostrada. Questo accade presumibilmente, come per la PCA, a causa della larghezza di alcune strade. Le reti neurali hanno dimostrato di essere molto robuste in autostrada dove classificano correttamente il 100% delle immagini.

Tuttavia viene classificato come autostrada anche tutto l'ambiente extraurbano (100%

di errori), per lo stesso motivo della PCA: la predominanza di frame in cui più della metà del cielo è sgombro da ostacoli e in cui la strada è priva di traffico. I risultati delle reti neurali sono riassunti in tabella 3.6.

	Città	Autostrada	Extraurbano
Città	100.0%	0.0%	0.0%
Autostrada	15.6%	84.4%	0.0%
Extraurbano	5.0%	0.0%	95.0%

Tabella 3.7: Matrice di confusione per il classificatore con SVM.

Le SVM si dimostrano essere il classificatore più affidabile sui tragitti misti, riuscendo ad identificare correttamente il 100% dei frame di città e ben il 95% dei frame di extraurbano. Per l'autostrada i rilevamenti corretti sono stati dell' 84.4% , tutto sommato un buon risultato visti i precedenti problemi di distinzione fra questa classe e l'extraurbano. Le percentuali si trovano riassunte in tabella 3.7. Alcuni errori delle SVM sono dovuti all'errata classificazione di frame autostradali indicati come cittadini, in figura 3.22.(a) si nota come questo fatto avvenga nel caso in cui ci si trovi, ad esempio, a fianco ad un pullman. Nelle figure 3.19.(c) e 3.20 vengono riprodotte due



Figura 3.19: Esecuzione classificatori in città

esecuzioni, in città e autostrada. In questi ambienti tutti i classificatori dimostrano al massimo la loro accuratezza. Nella figura 3.21 viene mostrato il passaggio sotto a un cavalcavia in ambito autostradale, si nota come sia un punto critico per il classificatore con SVM. Questo problema potrebbe evitarsi agendo sulla coda di isteresi del classificatore per stabilizzare la scelta avuta in precedenza.

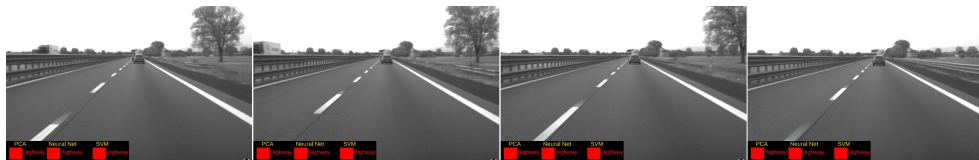


Figura 3.20: Esecuzione classificatori in autostrada



Figura 3.21: Passaggio sotto un ponte

3.6.11 Tempi di esecuzione

Il sistema è stato testato sul seguente hardware:

- cpu Intel(R) Mobile Core(TM)2 Duo P7450 @ 2.133GHz
- memoria ram SODIMM Synchronous 1600 MHz (0,9 ns) 2GiB x2
- Scheda video: ATI Mobility Radeon HD 4650 1GB@550MHz

In tabella 3.8 sono riassunti i tempi di elaborazione medi e minimi necessari per l'esecuzione del sistema sviluppato, in riferimento all'hardware utilizzato.

Le reti neurali richiedono un tempo medio di 1.27 ms, mentre le esecuzioni più veloci arrivano ad 1.01 ms, è sicuramente il metodo più veloce poiché gran parte del calcolo richiesto è stato fatto in fase di addestramento.

La classificazione con PCA ha un tempo medio di 2.01 ms ed uno minimo di 1.36 ms, leggermente maggiore a causa delle trasformazioni e dei molti passaggi che richiede. Ma la parte più "lenta" è sicuramente quella dedicata alle SVM, che arrivano in

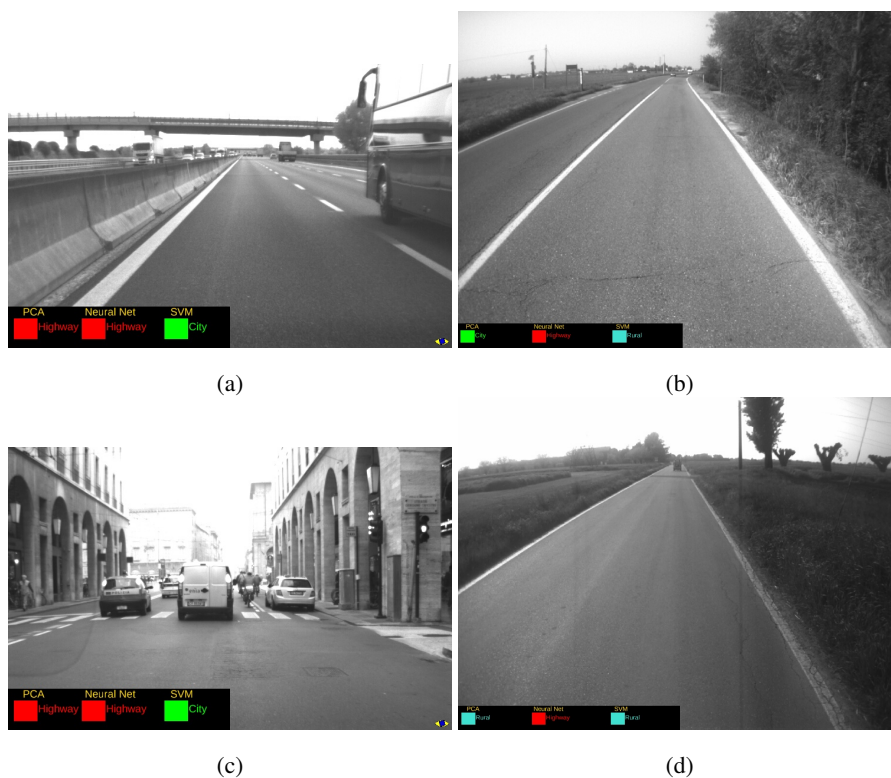


Figura 3.22: Risultati ottenuti: (a) Passaggio di fianco ad un pullman, scenario autostradale, (b) extraurbano, la PCA lo classifica città, (c) città, erroneamente classificato da PCA e NN come autostrada (d) extraurbano, correttamente classificato dalla PCA e dalle SVM Autostrada.

Funzione	minimo (ms)	medio (ms)
Normalizzazione istogramma	1.28	1.40
Filtro passa alto	4.81	5.0
Resize	0.82	0.897
Suddivisione	0.257	0.281
FFT	2.67	2.86
Campionamento della FFT	0.782	1.56
Esecuzione PCA	1.36	2.01
Esecuzione NN	1.01	1.27
Esecuzione SVM	17.9	18.5
Tracking	0.002	0.004
Algoritmo completo	35.2	37.4

Tabella 3.8: Tempi di esecuzione del sistema.

media a 18.5 ms. Colpa di questo ritardo è da addebitare ai molti prodotti scalari richiesti dal kernel che occorre fare quando si passa ad uno spazio di dimensioni molto più grande. Unendo tutte e quattro queste parti, il tempo medio richiesto dall'intero algoritmo è di 37.4 ms, con alcune esecuzioni che arrivano a 35.2 ms. Questo è un tempo decisamente ridotto che ben si adatta alle applicazioni automobilistiche.

3.7 Conclusioni

Le attività di ricerca connesse alla telecamera mobile hanno mostrato di produrre buoni risultati dal punto di vista delle prestazioni richieste e ottenute. Considerando lo stato dell'arte si può notare come alcuni sistemi si trovano in modo naturale al suo interno; mentre alcune applicazioni riescono a innovare alcuni aspetti e possono considerarsi idee che apportano un incremento significativo dal punto di vista della ricerca.

I sistemi realizzati nell'ambito di questa attività di ricerca sono divenuti dei prodotti commerciabili da parte dello spin-off VisLab s.r.l del Dipartimento di Ingegneria

dell'Informazione dell'Università degli Studi di Parma.

Alcuni sistemi sono stati oggetto di pubblicazione presso conferenze internazionali.

In generale, considerando le motivazioni che hanno spinto il lavoro di questa parte di tesi di dottorato bisogna ritenersi soddisfatti per i risultati ottenuti in termini di prestazioni, considerando che si possono in ogni caso sempre migliorare gli algoritmi considerando nuove tecniche o nuovi approcci.

Capitolo 4

Conclusioni

*I propose to consider the question, "Can machines think?"
This should begin with definitions of the meaning
of the terms "machine" and "think".*

Alan Turing
Computing machinery and intelligence

Le attività di ricerca relative a questa tesi di dottorato riguardano la visione artificiale ad ampio raggio. L'elaborazione di immagini viene considerata come parte fondamentale di questa disciplina, unendo le caratteristiche estrapolate dalle immagini a tecniche proprie dell'intelligenza artificiale che coinvolgono anche metodologie di apprendimento automatico, è stato possibile affrontare problemi complessi con soluzioni applicative innovative.

Lo sviluppo di sistemi commerciali o industriali o di ricerca applicata che a parole sembrano di facile soluzione per un essere umano, molte volte richiedono un percorso lungo di progettazione e sviluppo per poter essere implementate su di un calcolatore. Normalmente per progetti di visione artificiale le scelte iniziali hanno forti ricadute sulla riuscita del progetto e sulla conferma dei risultati attesi.

In questa attività di ricerca si è focalizzata l'attenzione su tecniche di elaborazione di immagini provenienti da telecamera fissa con applicazioni legate alla videosor-

veglanza e immagini provenienti da telecamera mobile con applicazioni nel campo automotive.

In entrambi i settori, che per molti versi paiono ortogonali, sono richieste conoscenze di base relative alla calibrazione delle telecamere, sia intrinseca, per conoscere la distorsione apportata dall'ottica che estrinseca, per associare correttamente i punti del mondo reale, della scena inquadrata, ai punti dell'immagine fornita. In alcuni casi, soprattutto nel campo automotive, i sensori presenti sul veicolo autonomo sono di tipologie differenti e in gran numero, per rafforzare la percezione grazie a tecniche di fusione sensoriale, per cui non si può prescindere dal considerare approcci che coinvolgano anche gli altri sensori.

Da un altro punto di vista si deve considerare la possibilità del nostro tempo di avere a disposizione con relativa facilità una quantità molto ampia di dati su cui impostare il lavoro di ricerca. Questo permette di cambiare visione dal punto di vista dell'approccio di ricerca passando da un approccio orientato al ragionamento per la risoluzione di problemi a un approccio orientato ai dati, e all'analisi degli stessi per cercare in essi la soluzione con tecniche di pattern recognition. Nei sistemi presentati in tutta la tesi di dottorato l'approccio orientato ai dati risulta evidente. Analizzando le immagini con alcune elaborazioni si possono ricercare caratteristiche salienti per poter effettuare ragionamenti e quindi classificare oggetti inquadrati, o addestrare tramite tecniche di apprendimento automatico il calcolatore a riconoscerli.

L'utilizzo di tecniche di intelligenza artificiale sempre più sta coinvolgendo vaste aree della scienza e tecnologia dell'informazione. Nel caso della visione artificiale questo connubio è in atto già da molti anni, si considera la percezione visiva come un vero e proprio senso per il calcolatore e in molte situazioni si utilizza questo legame anche in molte altre branche dell'ingegneria e dell'informatica. Il sistema per il riconoscimento di scenari in ambito automotive ha permesso di esplorare in parte sistemi e tecniche di apprendimento automatico per la realizzazione di un sistema che riuscisse a confrontarle per ottenere risultati soddisfacenti.

I sistemi sviluppati partendo da una forte connessione con lo stato dell'arte hanno mostrato come le tecniche utilizzate e gli algoritmi ideati abbiano incrementato l'ampiezza delle soluzioni in letteratura. Alcuni sistemi hanno risentito fortemente

dei suggerimenti ottenuti grazie all'analisi delle soluzioni prodotte nel mondo scientifico, in altre situazioni la ricerca è stata guidata da considerazioni legate alle limitazioni del tempo di esecuzione, ottenendo soluzioni più semplici e veloci anche se con prestazioni inferiori. Nel caso dei sistemi presentati nel capitolo 3, il riconoscitore di veicoli e il riconoscitore di pedoni, si nota come le prestazioni non siano eccellenti, però questo fatto ha permesso lo sviluppo di sistemi che soddisfacessero requisiti real-time.

Nel caso del sistema per il rilevamento dei tunnel presentato nel capitolo 3 la velocità in termini di tempo di esecuzione risultava un parametro fondamentale per il sistema, la soluzione trovata ha permesso di rispettare questo vincolo, ma anche di ottenere in modo pratico un approccio leggero e semplice con alte prestazioni.

In alcuni casi i sistemi sviluppati sono risultati totalmente innovativi, come nel caso del controllo di accesso ai varchi con telecamera fissa stereo. In questo caso in letteratura si trovano pochi lavori simili, con risultati dal punto di vista delle prestazioni inferiori. L'approccio con una tecnica basata su disparità, proveniente da applicazioni sviluppate nel laboratorio per il settore automotive, si è rivelata adatta allo scopo, realizzare un sistema modulare che potesse essere esteso a varie applicazioni.

Il lavoro svolto nell'attività di ricerca legata al dottorato ha permesso di ottenere dei risultati riconosciuti tramite pubblicazioni su riviste e conferenze internazionali. Il sistema per il rilevamento dei pedoni e il sistema per il rilevamento di tunnel sono stati presentati a conferenze internazionali, altri sistemi sono in fase di pubblicazione.

In conclusione si può affermare che l'obiettivo di sviluppare delle tecniche innovative di visione artificiale per l'elaborazione di immagini provenienti da telecamere fissa e mobile possa considerarsi raggiunto. Le prestazioni ottenute dai sistemi implementati sono risultate buone in termini di funzionamento e l'innovazione dei sistemi stessi risulta considerevolmente soddisfacente.

Appendice A

Classificazione e Apprendimento Supervisionato

L'intelligenza artificiale è una disciplina che riguarda la soluzione di problemi tramite l'utilizzo di funzioni sintetiche, astratte, di ragionamento e apprendimento tipiche dell'essere umano da parte di un elaboratore elettronico.

Questa disciplina riguarda un'area di ricerca molto vasta all'interno della scienza e tecnologia dell'informazione, si propone quindi di risolvere problemi complessi simulando il ragionamento e i processi cognitivi umani, un'importante lettura in merito è [28].

Una strategia per risolvere problemi complessi è quella di permettere al computer di apprendere la funzione che mappa input in output tramite degli esempi, nello stesso modo in cui i bambini imparano cosa siano le macchine sportive semplicemente indicandogli quali, tra un grande numero di macchine, sono sportive, piuttosto che dandogli una precisa definizione di sportività (Nello Cristianini, [29]). L'approccio che utilizza esempi per sintetizzare programmi è conosciuto come *metodologia di apprendimento*, e nei particolari casi in cui gli esempi sono coppie ingresso/uscita viene chiamato *apprendimento supervisionato*. Gli esempi della funzionalità input/output sono identificati come set di addestramento o *training set*.

Le coppie ingresso/uscita tipicamente riflettono una relazione funzionale che

mappa gli input con gli output, ma ciò può non essere sempre vero nel caso, per esempio, che gli output siano corrotti da rumore. Quando una evidente funzione tra input e output esiste, essa viene indicata col nome di funzione obbiettivo (*target function*). La stima della funzione obbiettivo che è appresa o indicata dall'algoritmo di apprendimento è conosciuta come *solution* del problema di apprendimento. Nel caso della classificazione questa funzione viene a volte indicata come funzione di decisione o *decision function*. La soluzione è scelta fra un insieme di funzioni candidate che mappano lo spazio degli ingressi nel dominio delle uscite. Di solito si sceglie un particolare insieme o classe di funzioni candidate conosciute come *ipotesi*, prima di provare ad individuare la funzione corretta. L'algoritmo che prende il *training set* come input e seleziona una ipotesi dallo spazio delle ipotesi viene indicato come *algoritmo di apprendimento*, mentre l'abilità di una ipotesi di classificare correttamente dei dati non appartenenti al training set è conosciuta come *generalizzazione*, ed è questa proprietà che si cerca di ottimizzare.

Particolare attenzione va posta nella scelta degli esempi che formano il training set, devono rappresentare caratteristiche o proprietà distintive di un certo oggetto (*feature*). Nell'elaborazione delle immagini le *feature* sono caratteristiche numeriche, ricavate misurando alcune proprietà significative. Devono essere scelte rispettando le seguenti proprietà:

- discriminanza: i valori delle feature sono simili per oggetti appartenenti alla stessa classe e sono molto diversi per oggetti appartenenti a classi diverse;
- indipendenza: i valori delle feature non devono essere correlati tra loro;
- minimalità: devono essere il minimo numero possibile di proprietà;
- disponibilità: facili e veloci da calcolare.

A.1 Reti Neurali

Come definita nel libro di Russell e Norvig, *Artificial Intelligence: A Modern Approach* [30], una rete neurale artificiale (indicata anche con ANN o NN) è un mo-

dello matematico che si ispira alla struttura e ad alcune funzioni delle reti neurali biologiche.

Una rete neurale artificiale è composta da un certo numero di nodi, o unità, connessi fra loro; ciascun collegamento ha un peso numerico ad esso associato. I pesi sono il principale mezzo di memorizzazione a lungo termine nelle reti neurali e l'apprendimento in genere ha luogo aggiornando i pesi. Alcune unità sono collegate con l'ambiente esterno e possono essere designate come unità di ingresso o di uscita. I pesi vengono modificati in modo tale da portare il comportamento della rete, in termini di associazione di uscite agli ingressi, ad essere più in linea con quanto richiesto dall'ambiente che fornisce gli input.

Ciascuna unità ha un insieme di collegamenti di ingresso che provengono da altre unità; l'idea è che ciascuna unità effettui un calcolo basato sugli ingressi provenienti dai suoi vicini senza il bisogno di alcun controllo esterno e globale sull'insieme dei nodi. Il calcolo è suddiviso tra due componenti. In primo luogo una componente lineare, chiamata funzione di ingresso, calcola la somma pesata dei valori d'ingresso delle unità. In secondo luogo una componente non lineare chiamata funzione di attivazione, trasforma la somma pesata nel valore finale che funge da valore di attivazione dell'unità. Solo se tale valore supera una determinata soglia l'unità darà il proprio contributo. In genere tutte le unità in una rete usano la stessa funzione di attivazione; tre scelte comuni di funzione di attivazione sono la funzione a gradino, la funzione di segno e la sigmoide, illustrate in figura A.1.

Per costruire una rete neurale che svolga un qualche compito si deve dapprima decidere quante unità occorre utilizzare, che tipo di unità è adatto e come tali unità devono essere connesse per formare una rete. Le reti che presentano una serie di nodi collegati fra quelli di ingresso e di uscita sono chiamate multistrato, perchè possiedono i cosiddetti *strati nascosti*, mentre le reti che non hanno strati nascosti vengono chiamate perceptron. In figura A.2 è illustrato lo schema di una rete neurale multistrato, in cui i nodi rappresentano le unità o neuroni, divisi in strati di ingresso, nascosti e uscita, e gli *edge* i loro possibili collegamenti.

Dopodichè si inizializzano i pesi della rete, di solito in modo casuale, e la si addestra impiegando un algoritmo che si serve di un insieme di esempi o *training set*.

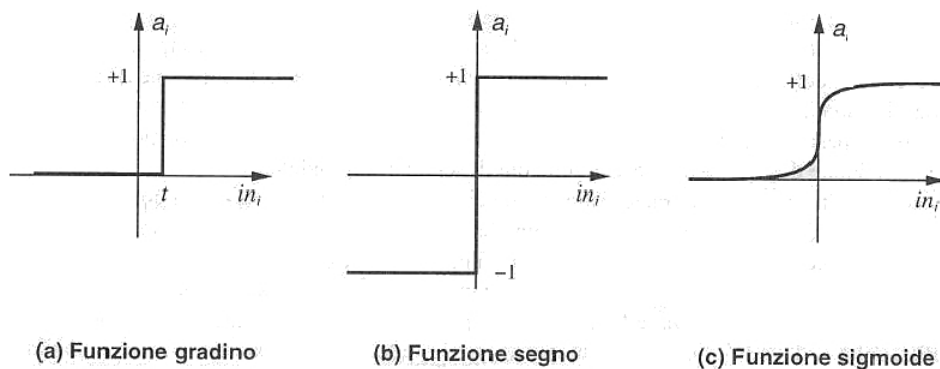


Figura A.1: Funzioni di attivazione per le reti neurali

L'insieme indica quali sono gli output che devono corrispondere a particolari valori di input. L'addestramento avviene con un metodo chiamato propagazione all'indietro: vengono sottoposti alla rete degli esempi di ingresso e se la rete calcola un vettore di uscita che corrisponde all'obiettivo non si fa nulla. Se invece c'è un errore (una differenza fra l'uscita e l'obiettivo) si procede a modificare i pesi in modo da ridurre l'errore. Il "trucco" consiste nel suddividere la colpa dell'errore tra i diversi pesi che vi contribuiscono.

A.1.1 Apprendimento on-line e off-line

In base a quando e come sono forniti al classificatore gli esempi di addestramento, esso può essere diviso in *on-line* e *off-line*.

Nell'apprendimento *off-line*, tutti i dati sono memorizzati e possono essere acceduti ripetitivamente. Nell'apprendimento *on-line* invece, ogni caso viene scartato dopo che è stato processato e i pesi aggiornati.

Con l'apprendimento *off-line*, si possono calcolare le funzioni obiettivo per ogni insieme di punti fissato, in tal modo si può vedere dove si è compiuto del progresso nell'apprendimento. Si calcola un minimo della funzione obiettivo per ogni preci-

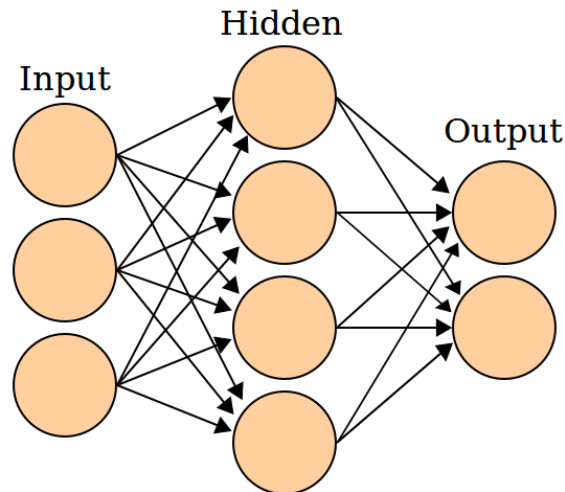


Figura A.2: Nodi interconnessi in una rete neurale

sione desiderata, utilizzando una varietà di algoritmi per evitare i minimi locali, come l'inizializzazione multipla *random* o gli algoritmi di ottimizzazione globale. Si calcola poi la funzione di errore su un *validation set* e si usa la fermata anticipata, o *early stopping*, oppure si sceglie fra differenti reti per migliorare la generalizzazione. Si usa la *cross-validation* e il *bootstrapping* per stimare l'errore di generalizzazione e si possono calcolare la predizione e gli intervalli di confidenza.

Con l'apprendimento *on-line* non si può fare nulla di queste cose perchè non si può calcolare la funzione obbiettivo sul *training set* o la funzione di errore sul *validation set* per un insieme fissato di punti, poichè questi dati non vengono salvati. Quindi l'apprendimento *on-line* è in genere molto più difficile ed inaffidabile dell'apprendimento off-line.

A.1.2 *Backpropagation*

L'algoritmo di *backpropagation* è utilizzato negli apprendimenti con supervisione. Esso permette di modificare i pesi delle connessioni in modo tale che si minimizzi una certa funzione errore E . Tale funzione dipende dal vettore i -esimo di output out_i

restituito dalla rete, dato il vettore i -esimo di ingresso x_i e dal vettore i -esimo di output y_i che si desidera (e che fa parte del training set). Il training set è dunque un insieme di n coppie di vettori (x_i, y_i) , con $h = 1, \dots, N$. La funzione errore che si deve minimizzare si può scrivere come:

$$E(w) = \frac{1}{2} \sum_h \sum_k (out_k^h - y_k^h)^2 \quad (\text{A.1})$$

dove l'indice k rappresenta il valore corrispondente al k -esimo nodo di uscita. $E(w)$ è una funzione dipendente dai pesi (che in generale variano nel tempo), per minimizzarla si può utilizzare l'algoritmo della discesa del gradiente (*gradient-descent*). L'algoritmo parte da un punto generico $x(0)$ e calcola il gradiente $\nabla f(x(0))$. Il gradiente dà la direzione verso cui muoversi lungo la quale si ha il massimo incremento (o decremento se si considera $-\nabla$). Definita la direzione ci si muove di una distanza η predefinita a priori e si trova un nuovo punto $x(1)$ sul quale è calcolato nuovamente il gradiente. Si continua iterativamente finché il gradiente non è nullo. L'algoritmo di *backpropagation* può essere diviso in due passi:

- *Forward pass*: l'input dato alla rete viene propagato al livello successivo e così via ai livelli successivi (il flusso di informazioni si sposta in avanti). Si calcola dunque $E(w)$, l'errore commesso.
- *Backward pass*: L'errore fatto dalla rete è propagato all'indietro (backward) e i pesi sono aggiornati in maniera appropriata.

I passi logici per addestrare una rete neurale con apprendimento supervisionato sono i seguenti:

- Costruire un insieme di *pattern* di *input* ed il relativo insieme di *pattern* di *output* desiderati;
- Inizializzare i pesi della rete neurale (le connessioni tra i neuroni) a dei valori casuali, piccoli rispetto ai valori futuri che assumeranno, ed a norma nulla;
- Ciclo di apprendimento, esce dal ciclo solo quando l'errore generale è minore di quanto si è deciso oppure dopo un determinato numero di iterazioni:

- Ciclo *feedforward*, dallo strato di *input* a quello di *output*:
 - * estrarre un pattern di input a caso tra quelli a disposizione;
 - * calcolare il valore di tutti i neuroni successivi (sommatorie di produttorie);
 - * detrarre dal risultato il valore di soglia di attivazione del neurone (se il valore di soglia non é già stato simulato con l'aggiunta di un neurone ad ingresso fisso a valore 1.0);
 - * filtrare l'uscita del neurone applicando una funzione logistica per far diventare tale valore input del neurone successivo;
 - * Confrontare il risultato ottenuto della rete con il pattern di output relativo all'input inserito e ricavare l'errore attuale della rete.
- Ciclo di *backpropagation*, dallo strato di *output* a quello di *input*:
 - * calcolare la correzione da apportare ai pesi secondo la regola di localizzazione del minimo scelta;
 - * applicare la correzione ai pesi dello strato.

Al fine di avere un buon addestramento, l'insieme dei pattern di input deve essere completo, quindi, dopo aver scelto in modo casuale un pattern, nell'estrazione del successivo il vecchio non deve partecipare al ballottaggio. Parteciperá cosí solo chi non è ancora stato estratto, per poi ripescare dall'insieme completo quando tutti sono stati estratti almeno una volta.

A.1.3 Overfitting

La rete neurale deve avere capacità di comprensione del modello statistico dei dati, non memorizzare i soli dati del training set. Solo cosí può generalizzare, cioè rispondere esattamente a *input* non in *training set*. La tecnica migliore per evitare ciò è l'*early stopping*. Nell'apprendimento l'errore E tende a 0 perchè la rete sta imparando meglio, curva blu in figura A.3. Usando un *validation set* di coppie non usate nel *training set*, si misura la E e si crea la seconda curva che tende a un minimo e poi ricresce, come la rossa in figura A.3: da quel punto la rete sta imparando il *training*

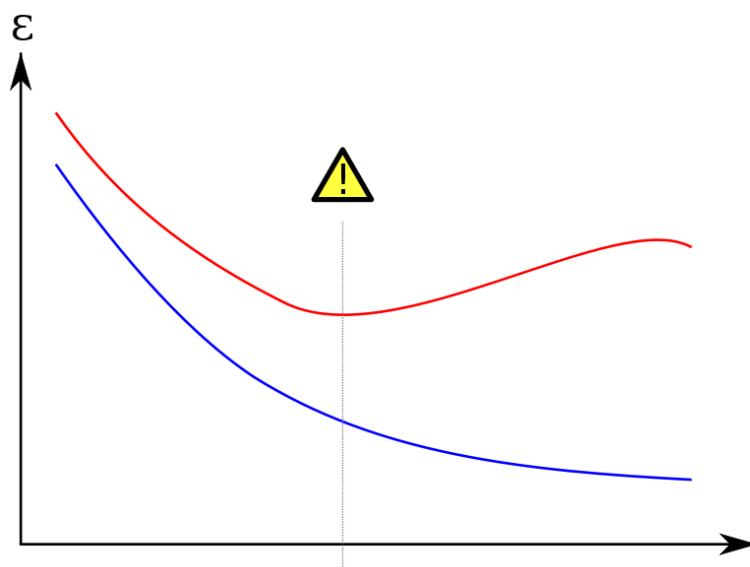


Figura A.3: Correzione dell'overfitting

set e non il suo modello statistico. Si ferma l'apprendimento al minimo di E rispetto al *validation set*.

A.1.4 Valutazione delle prestazioni

Per misurare le prestazioni di una rete neurale dopo l'apprendimento, si crea il *test set* formato da coppie non usate per il *training* e per il *validation set*. In genere il *test set* è un terzo del *training set* ed è composto da *input* critici su cui la risposta della rete deve essere buona, altrimenti si butta via la rete.

Per ogni coppia del test set:

- si calcola la risposta della rete all'input;
- si calcola l'errore dato dalla differenza tra output rete e output di coppia.

L'errore totale è dato dalla somma degli errori avuti per ogni coppia. Secondo il tipo di problema, si usano varie tecniche statistiche per decidere se utilizzare o meno la

rete creata; in genere si accetta una rete se sul test set ha mostrato un errore inferiore al 20-25% delle coppie.

Per un maggiore dettaglio sulle funzioni delle reti e sui metodi di addestramento si rimanda al libro [30].

A.2 Support Vector Machines

Le Support Vector Machines (SVM) [29] in statistica ed informatica rappresentano un insieme di metodi correlati di apprendimento automatico per analizzare dati e riconoscere pattern, usati per la classificazione e l'analisi di regressione. Sono stati inventati da Vladimir Vapnik e l'ultima versione standard è stata proposta da Vapnik e Corinna Cortes nel 1995.

Le SVM standard prendono come ingresso un insieme di dati e predicono, per ogni input, quale di due possibili classi comprende l'input, rendendo le SVM un classificatore binario lineare non probabilistico.

Dato un insieme di esempi di addestramento, ciascuno marcato come appartenente ad una delle due categorie, un algoritmo di addestramenti di SVM costruisce un modello che assegna nuovi esempi in una o nell'altra categoria. Un modello SVM è una rappresentazione di esempi come punti nello spazio, mappati in modo che gli esempi di categorie separate sono divisi da un netto divario il più ampio possibile. Nuovi esempi sono poi mappati nello stesso spazio e predetti di appartenere ad una categoria in base su quale lato sono caduti. In figura A.4 sono rappresentate due regioni indicate con colori diversi, separate mediante un iperpiano, i punti che cadono in esse vengono etichettate come appartenenti a quella classe. I punti in verde sono i cosiddetti *support vector*.

Più formalmente, una Support Vector Machines costruisce un iperpiano o un insieme di iperpiani in uno spazio a molte o infinite dimensioni, che può essere usato per la classificazione, regressione, o altro. Intuitivamente, una buona separazione viene raggiunta dall'iperpiano che ha la maggiore distanza dal più vicino punto dell'insieme dei dati fra tutte le classi (chiamato *marginale funzionale*), in modo che in genere più è largo il margine più è piccolo l'errore di generalizzazione del classificatore.

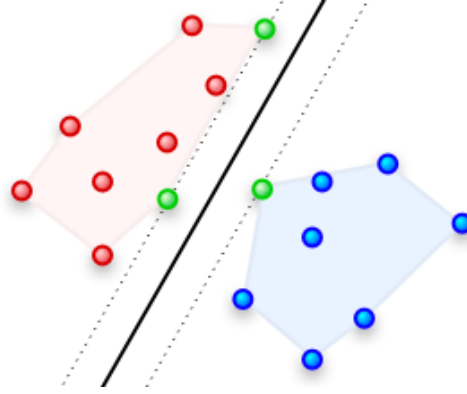


Figura A.4: Separazione tramite SVM lineare

In figura A.5 l'iperpiano H_3 (verde) non separa le due classi, H_1 (blu) riesce a separarle ma con poco margine, H_2 (rosso) separa col massimo del margine possibile.

Considerando che il problema originario può trovarsi in uno spazio di dimensioni finite, spesso capita che gli insiemi da discriminare non siano linearmente separabili in quello spazio. Per questa ragione, è stato proposto che lo spazio originale di dimensioni finite, chiamato *input space*, sia mappato in uno spazio di dimensioni molto superiori, *features space*, rendendo presumibilmente più semplice la separazione, come illustrato in figura A.6. Per rendere ragionevole il carico computazionale, il mapping usato dagli schemi SVM sono progettati in modo da garantire che i prodotti scalare siano calcolati facilmente nei termini delle variabili dello spazio originale, definendoli con una funzione *kernel* $K(x, y)$ selezionata in base al problema. Gli iperpiani nello spazio a più grande dimensionalità sono definiti come l'insieme dei punti il cui prodotto scalare con un vettore in quello spazio è costante.

I vettori che definiscono gli iperpiani possono essere scelti per essere combinazioni lineari con parametri α_i delle immagini dei vettori delle *features* presenti nella base di dati. Con questa scelta di un iperpiano, i punti x nel *feature space*, che sono mappati nell'iperpiano sono definiti dalla relazione A.2:

$$\sum_i \alpha_i K(x_i, x) = \text{costante} \quad (\text{A.2})$$

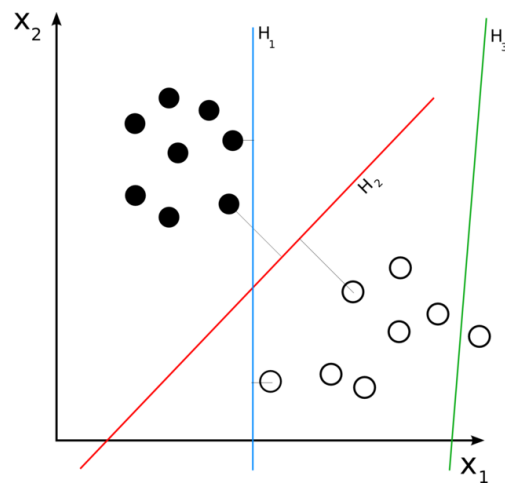


Figura A.5: Diversi iperpiani di separazione

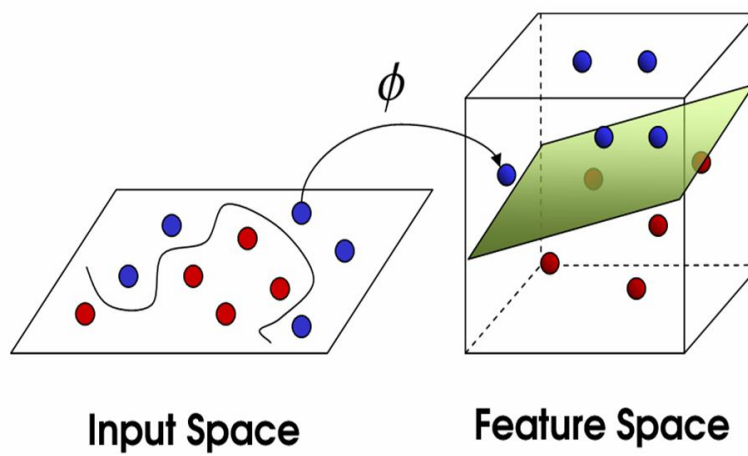


Figura A.6: Trasformazione dello spazio di input in uno spazio multidimensionale

Da notare che se $K(x, y)$ diventa molto piccolo, ogni elemento nella somma misura il grado di vicinanza del punto di test x con il corrispettivo punto x_i delle base di dati. Così la somma dei kernel di cui sopra può essere usata per misurare la vicinanza relativa di ciascun punto test dai punti dei dati originati in uno o nell'altro insieme da discriminare.

Si noti il fatto che l'insieme dei punti x mappati in qualsiasi iperpiano può permettere una discriminazione più complessa tra gli insiemi che non erano affatto convessi nello spazio originale.

A.2.1 SVM lineare

In questa sezione si introduce la prima versione sviluppata di SVM che utilizza vincoli e funzioni lineari.

Venga dato un insieme di dati D , un set di n punti nella forma:

$$D = \{(x_i, y_i) | x_i \in R^P, y_i \in \{-1, 1\}\}_{i=1}^n \quad (\text{A.3})$$

Dove y_i è 1 o -1, ed indica la classe a cui x_i appartiene. Ciascun x_i è un vettore reale p -dimensionale. Si vuole trovare l'iperpiano con il margine massimo che divide i punti aventi $y_i = 1$ da quelli che hanno $y_i = -1$. Ogni iperpiano può essere scritto come l'insieme di punti x che soddisfano:

$$w \cdot x - b = 0 \quad (\text{A.4})$$

dove $\cdot$ indica il prodotto scalare e w il vettore normale all'iperpiano. Il parametro $\frac{b}{\|w\|}$ determina l'offset dell'iperpiano dall'origine lungo il vettore normale w . Si devono scegliere w e b che massimizzino il margine, o la distanza fra gli iperpiano paralleli che siano il più lontano possibile tra loro, pur separando i dati. Questi iperpiano possono essere descritti dalle equazioni:

$$w \cdot x - b = 1 \quad (\text{A.5})$$

e

$$w \cdot x - b = -1 \quad (\text{A.6})$$

Si noti che se i dati di addestramento sono linearmente separabili, si possono selezionare i due iperpiani del margine in modo che non ci siano punti tra di loro e poi cercare di massimizzare la loro distanza. Utilizzando la geometria, è possibile trovare la distanza fra questi due iperpiani tramite $\frac{2}{\|w\|}$, in modo da minimizzare $\|w\|$. Per prevenire che i punti dei dati cadano nel margine, si aggiunge il vincolo:

Per ogni i :

$$w \cdot x_i - b \geq 1 \text{ per } x_i \text{ della prima classe} \quad (\text{A.7})$$

o

$$w \cdot x_i - b \leq -1 \text{ per } x_i \text{ della seconda classe} \quad (\text{A.8})$$

Il ché può essere riscritto così:

$$y_i(w \cdot x_i - b) \geq 1, \text{ per tutti } 1 \leq i \leq n \quad (\text{A.9})$$

È possibile mettere queste assieme per ottenere il problema di ottimizzazione:

Minimizzare (in w, b)

$$\|w\| \quad (\text{A.10})$$

soggetto di (per ogni $i = 1, \dots, n$)

$$y_i(w \cdot x_i - b) \geq 1 \quad (\text{A.11})$$

La figura A.7 mostra l'iperpiano con il margine massimo e i margini per una SVM lineare addestrata con campioni di due classi, i campioni sul margine sono detti *support vectors*.

A.2.2 SVM non lineare

L'algoritmo originale dell'iperpiano ottimo proposto da Vapnik nel 1963 era un classificatore lineare. Tuttavia, nel 1992, Bernhard Boser, Isabelle Guyon e Vapnik suggerirono un modo per creare un classificatore non lineare usando il "trucco" della

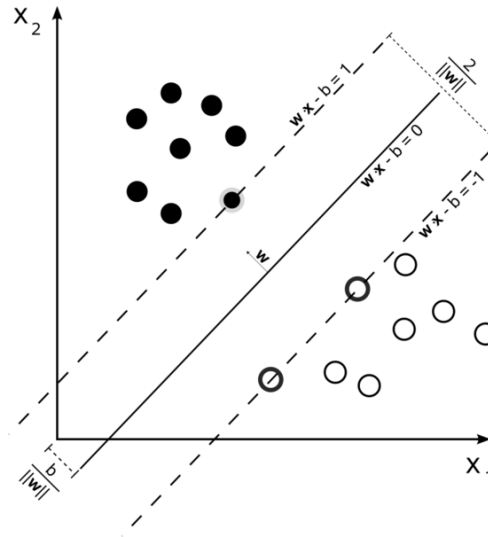


Figura A.7: Iperpiano con il margine massimo

funzione kernel, originariamente proposta da Aizerman e altri [31], per l'iperpiano del margine massimo [32]. L'algoritmo risultante è formalmente simile all'originario, eccetto che ogni prodotto scalare viene replicato da una funzione *kernel* non lineare. Questo permette all'algoritmo di adattare l'iperpiano in uno spazio delle *features* trasformato. La trasformazione può essere non lineare e lo spazio trasformato a grandissime dimensioni; così se il classificatore è un iperpiano nello spazio a molte dimensioni delle *features*, potrebbe essere non lineare nello spazio degli ingressi originale.

Se il *kernel* usato è una funzione Gaussiana dalla base radiale, il corrispondente spazio delle *features* è uno spazio di Hilbert di infinite dimensioni. I classificatori del margine massimo sono ben regolarizzati, così che le dimensioni infinite non rovinano i risultati.

Alcuni *kernel* comuni includono:

- Lineare: $k(x_i, x_j) = x_i^T \cdot x_j$
- Polinomiale (omogeneo): $k(x_i, x_j) = (x_i^T \cdot x_j)^d$

- Polinomiale (eterogeneo): $k(x_i, x_j) = (\gamma x_i^T \cdot x_j + r)^d, \gamma > 0$
- Funzione Gaussiana di base radiale(RBF): $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, per $\gamma > 0$
A volte parametrizzato usando $\gamma = 1/2\sigma^2$
- Tangente iperbolica: $k(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c)$, per alcuni (non tutti) $k \geq 0$ e $c \leq 0$

Il *kernel* è correlato alla trasformazione $\phi(\mathbf{x}_i)$ dall'equazione

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j). \quad (\text{A.12})$$

Anche il valore w viene mappato nello spazio trasformato, con

$$\mathbf{w} = \sum_i \alpha_i y_i \phi(\mathbf{x}_i). \quad (\text{A.13})$$

Il prodotto scalare con w per la classificazione può ancora essere calcolato con il "trucco" del kernel, ovvero

$$\mathbf{w} \cdot \phi(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) \quad (\text{A.14})$$

Tuttavia, non esiste in genere un valore w' che sia

$$\mathbf{w} \cdot \phi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x}) \quad (\text{A.15})$$

La figura A.8 mostra come si possano trasformare gli input per poter discriminare con una funzione lineare che nello spazio degli ingressi sarebbe non lineare.

A.2.3 Proprietà

Le SVM appartengono ad una famiglia di classificatori lineari generalizzati e possono essere interpretati come una estensione dei *perceptroni*. Esse possono anche essere considerate un caso speciale della *regolarizzazione di Tikhonov*. Una proprietà speciale rivela che esse, simultaneamente, minimizzano l'errore della classificazione

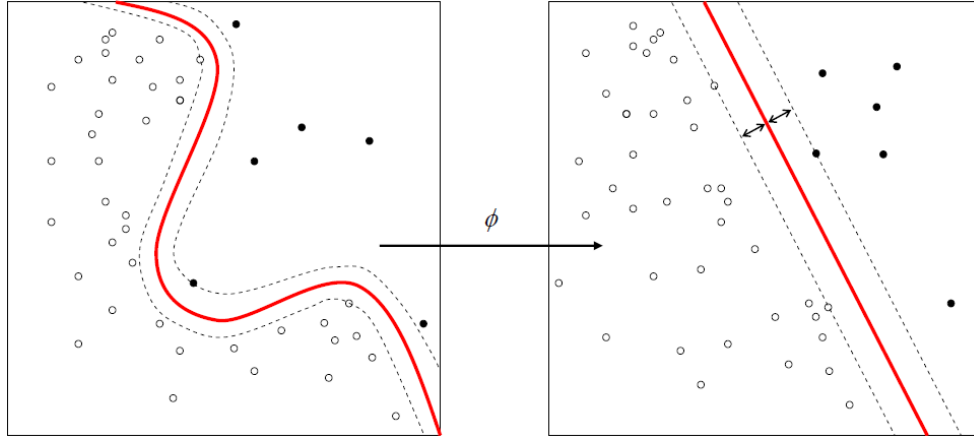


Figura A.8: Kernel Machine

empirica e massimizzano il margine geometrico; quindi sono anche conosciute come classificatori dal margine massimo (in engl. *maximum margin classifier*).

Un confronto tra SVM e altri classificatori è stato compiuto da Meyer, Leisch e Hornik [33].

Selezione dei parametri

L'efficacia delle SVM dipende dalla selezione del kernel, i parametri del kernel, e il parametro C detto di *soft margin*.

Una scelta comune è quella del kernel Gaussiano, che possiede un solo parametro γ . La miglior combinazione di C e γ è spesso selezionata da una ricerca *grid* con sequenze di crescita esponenziale di C e γ , per esempio, $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, \dots, 2^1, 2^3\}$. Tipicamente, ogni scelta di combinazione di parametri viene controllata usando una *cross-validation*, e vengono raccolti i parametri con la miglior accuratezza in fase di *cross-validation*.

Il modello finale, che è usato per testare e per classificare nuovi dati, viene addestrato sull'intero *training set* usando i parametri selezionati [34].

Problematiche

I potenziali svantaggi delle SVM seguono i seguenti due aspetti:

- Probabilità non calibrata di appartenenza alle classi;
- Le SVM sono solo direttamente applicabili per problemi a due classi. Tuttavia, come vedremo in seguito, possono essere applicati algoritmi che riducono il problema con molte classi in diversi problemi binari;

I parametri di un modello risolto possono essere di difficile interpretazione.

A.2.4 SVM Multiclasse

Le SVM multiclasse si propongono di assegnare etichette alle istanze usando le *support vector machines* dove le etichette sono tratte da un insieme finito di diversi elementi. L'approccio dominante è ridurre il singolo problema multiclasse in diversi problemi di classificazione binaria [35]. Metodi comuni per tale riduzione includono [35], [36]:

- La costruzione di classificatori binari che distinguono fra (I) una delle etichette dalle altre (*one-versus-all*) o (II) fra ogni coppia di classe (*one-versus-one*). La classificazione di nuove istanze per il caso *one-versus-all* è ottenuta con la strategia del *winner-takes-all*, in cui il classificatore con la funzione di output più alta assegna la classe (è importante che le funzioni di uscita siano calibrate per produrre punteggi comparabili). Per l'approccio *one-versus-one*, la classificazione è fatta con la strategia *max-wins*, in cui ogni classificatore assegna l'istanza ad una delle due classi, incrementando di uno il voto per la classe assegnata, e in fine la classe con più voti determina la classificazione dell'istanza.
- DAGSVM [37].
- *error-correcting output codes* [38].

Crammer e Singer proposero un metodo per SVM multiclasse che trasforma il problema della classificazione multiclasse in un singolo problema di ottimizzazione, piuttosto che decomporlo in problemi di classificazione binaria [39].

Appendice B

Algoritmi di boosting

B.1 Introduzione agli algoritmi di boosting

Gli algoritmi di classificazione tradizionale prevedono uno schema rigido in cui un'istanza viene introdotta come input ed un classificatore decide come classificarla producendo in uscita l'etichetta adeguata.

In alcuni casi è possibile considerare una funzione che legghi più classificatori, ognuno diverso dall'altro, che, parallelamente, analizzano l'istanza producendo un valore di output, un voto. Queste tipologie di classificatori producono una funzione d'insieme, o di *ensemble*.

$$ensemble(x) = \sum_{i=1}^n w_i C_i(x) \quad (B.1)$$

Nell'equazione B.1 è rappresentata la formula per un classificatore che consideri un numero di classificatori indipendenti, in numero n indicati con C_i . In base al valore di uscita si può decidere come suddividere in classi le istanze x . Il valore w_i rappresenta il peso assegnato ad ogni classificatore.

Con questo tipo di formula si possono realizzare due tipologie di assegnamento dei pesi, chiamato *voting*:

- voting semplice: se $w_k = w_i$ per ogni k, j ;

- voting pesato: assegnando dei pesi che influenzano il voto finale.

Questa tipologia di classificatori possono essere utili poichè permettono di specificare più sottoclassificatori, considerando delle parametrizzazioni personali. Normalmente hanno prestazioni migliori rispetto ad un classificatore singolo e sono più resistenti al rumore.

Di contro hanno due svantaggi principali: richiedono più tempo in termini di esecuzione, anche se parallelizzandoli si potrebbe risolvere in parte questo problema, e sono sensibili all'*overfitting* dei dati.

Un'alternativa a questo modello è fornito dagli algoritmi di classificazione di tipo *boosting*. Gli algoritmi di *boosting* hanno il pregio di poter formare un classificatore forte partendo da un insieme di classificatori deboli.

In questo modello i classificatori vengono eseguiti in sequenza. Ciascun classificatore dipende dal precedente, cercando di migliorarne gli errori. Gli esempi che sono stati classificati in modo scorretto dai classificatori precedenti sono scelti più spesso, oppure pesati maggiormente. L'ipotesi finale si ottiene grazie alla formula .

$$H(x) = \sum_{k=1}^T \alpha_k C_k(x) \quad (\text{B.2})$$

Il valore del peso α dipende dagli errori commessi dai classificatori.

B.2 Introduzione ad AdaBoost

In un articolo del 1997 Yoav Freund e Robert E. Schapire introducono per la prima volta il meta-algoritmo AdaBoost. Il nome AdaBoost deriva dalla contrazione di *adaptive boost*, è un algoritmo per la classificazione che ha come base un algoritmo di *boosting*, rafforzato da una componente adattativa legata all'errore commesso ad ogni iterazione.

Questo algoritmo, al suo interno, prevede l'utilizzo di un qualunque classificatore, chiamato classificatore debole, in inglese, *weak learn*. L'idea di fondo può, in effetti, sembrare molto banale: al posto di considerare un unico classificatore molto

raffinato e specializzato, se ne considerano un numero qualunque di tipo molto semplice, che sono appunto i classificatori deboli. Questi, uniti, riusciranno ad arrivare ad una classificazione corretta. Quindi, invece di considerare tutte le caratteristiche che dovrebbe possedere un oggetto per essere inserito in una classe, si pone l'attenzione solamente su alcune di esse in modo indipendente, pesandole in seguito.

Grazie a questo algoritmo, i due autori dell'articolo, ricevettero il premio Gödel nel 2003. L'algoritmo così come viene presentato in [40], [41] e [42] presenta due versioni chiamate AdaBoost.M1 ed AdaBoost.M2.

B.3 AdaBoost versione M1

La prima versione dell'algoritmo è quella più semplice ed è illustrata nell'algoritmo sottostante.

Algoritmo AdaBoost.M1

Input: una sequenza di m campioni $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$
 con etichette $y_i \in Y = \{1, \dots, k\}$
 un algoritmo di apprendimento debole, **WeakLearn**
 un intero T , che rappresenta il numero di iterazioni da eseguire.

Inizializza $D_1(i) = 1/m$ per ogni i .

Per ogni iterazione $t = 1, 2, \dots, T$

1. Esegui l'algoritmo **WeakLearn**, utilizzando la distribuzione D_t .
2. Ritorna un'ipotesi $h_t : X \rightarrow Y$.
3. Calcola l'errore di h_t :

$$\varepsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

Se $\varepsilon_t > 1/2$ allora setta $T = t - 1$ e interrompi il ciclo.

4. Setta $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$

5. Aggiorna la distribuzione D_t :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{se } h_t(x_i) = y_i \\ 1 & \text{altrimenti} \end{cases}$$

dove Z_t è una costante di normalizzazione scelta in modo che D_{t+1} sia una distribuzione.

Output: l'ipotesi finale:

$$h_{fin} = \arg \max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}$$

L'algoritmo prende in ingresso un insieme di addestramento, *training set*, di m campioni $S = \langle (x_1, y_1), \dots, (x_m, y_m) \rangle$. x_i è un'istanza sorteggiata da uno spazio X rappresentata in un modo appropriato, ad esempio, con un vettore di attributi; mentre $y_i \in Y$ è l'etichetta legata alla classe di appartenenza associata ad x_i , di cardinalità finita k .

L'algoritmo AdaBoost utilizza un sotto algoritmo chiamato *weakLearner*, che è un classificatore debole realizzato in un qualunque modo. L'algoritmo richiama questo classificatore in una serie di iterazioni, nel nostro caso, pari a T . Ad ogni iterazione t , l'algoritmo AdaBoost esegue il *weakLearner* con una distribuzione D_t sul training set S .

In risposta, il *weakLearner*, calcola l'ipotesi di classificazione $h_t : X \rightarrow Y$, che dovrebbe classificare una parte non banale dell'insieme di addestramento. L'obiettivo del *weakLearner* è di trovare l'ipotesi h_t che minimizza l'errore (di addestramento) $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$. È da notare che l'errore è misurato rispetto alla distribuzione D_t che viene fornita al *weakLearner*.

Questo processo continua per T cicli ed, alla fine, il *booster* combina le ipotesi deboli $h_1, h_2, \dots, h_T$ in una singola ipotesi finale h_{fin} .

Rimangono non specificate due cose: come viene calcolato D_t ad ogni turno e come viene calcolata h_{fin} . Differenti schemi di boosting rispondono a questi in-

terrogativi in modi diversi; AdaBoost.M1 risponde con la semplice regola mostrata nell'algoritmo.

La distribuzione iniziale è distribuita uniformemente su S , quindi $D_1(i) = 1/m$ per ogni i . Per calcolare la distribuzione D_{t+1} dalla D_t , e l'ultima ipotesi debole h_t , moltiplichiamo il peso del campione i per un numero $\beta_t \in [0, 1)$ se h_t classifica x_i correttamente, altrimenti il peso rimane inalterato. Il peso viene in seguito rinormalizzato dividendolo per una costante di normalizzazione Z_t .

Effettivamente, campioni semplici, che vengono classificati in modo corretto da molte delle precedenti ipotesi deboli hanno peso basso; mentre i campioni complicati, che tendono ad essere classificati in modo errato, acquistano peso maggiore. Così, AdaBoost, concentra il peso maggiore sui campioni che sembrano più difficilmente classificabili per il weakLearn.

Il numero β_t è calcolato, come mostrato nell'algoritmo, come funzione di ϵ_t . L'ipotesi finale h_{fin} è un voto pesato ottenuto dalle ipotesi deboli.

Per questo, per una data istanza x , h_{fin} restituisce un'etichetta y che massimizza la somma dei pesi delle ipotesi deboli che avevano predetto quell'etichetta. Il peso dell'ipotesi h_t è definito come $\ln(1/\beta_t)$ cosicchè i pesi maggiori sono dati dalle ipotesi con gli errori più bassi.

La più importante proprietà teorica di AdaBoost è data dal teorema 2.1. Esso mostra che, anche se le ipotesi finali hanno costantemente errore leggermente migliore di $1/2$, l'errore dell'ipotesi finale h_{fin} va a zero con velocità esponenziale. Per i problemi di classificazione binaria, questo vuol dire che, le ipotesi deboli, devono essere leggermente migliori di un valore di tipo casuale.

Teorema B.1. *Supponiamo di avere un algoritmo di apprendimento debole weakLearn, che, chiamato dall'algoritmo AdaBoost.M1, genera delle ipotesi con errore $\epsilon_1, \dots, \epsilon_T$, dove ϵ_t è definito come nell'algoritmo illustrato in precedenza.*

Assumiamo che ogni $\epsilon_t \leq 1/2$ e prendiamo $\gamma_t = 1/2 - \epsilon_t$.

Allora, il limite superiore sull'errore dell'ipotesi finale h_{fin} , è

$$\frac{1}{m} |\{i : h_{fin}(x_i) \neq y_i\}| \leq \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2 \sum_{t=1}^T \gamma_t^2\right)$$

Uno svantaggio di AdaBoost.M1 è che non è capace di trattare ipotesi deboli con errore maggiore di $1/2$. L'errore atteso di un'ipotesi che assegni un'etichetta a caso è $1 - 1/k$, dove k è il numero delle possibili etichette. Così il requisito per AdaBoost.M1 per $k = 2$ è che la predizione sia leggermente migliore della scelta casuale. Tuttavia, quando $k > 2$, il requisito di AdaBoost.M1 è molto più forte e difficile da trovare.

B.4 AdaBoost versione M2

Gli autori di AdaBoost presentano a corredo dell'algoritmo AdaBoost.M1 una seconda versione, chiamata AdaBoost.M2. La seconda versione di AdaBoost cerca di superare le difficoltà riscontrate dalla prima, aumentando la comunicazione tra l'algoritmo di *boosting* e i classificatori deboli. Possiamo esaminare l'algoritmo, però non dopo averlo descritto.

Algoritmo AdaBoost.M2

Input: sequenza di m campioni $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$
 con etichette $y_i \in Y = \{1, \dots, k\}$
 un algoritmo debole di apprendimento, **WeakLearn**
 un intero T , che rappresenta il numero di iterazioni.

Considera $B = \{(i, y) : i \in \{1, \dots, m\}, y \neq y_i\}$

Inizializza $D_1(i, y) = 1/|B|$ per $(i, y) \in B$.

Per ogni iterazione $t = 1, 2, \dots, T$

1. esegui l'algoritmo **WeakLearn**, utilizzando la distribuzione D_t .
2. Ritorna un'ipotesi $h_t : X \times Y \rightarrow [0, 1]$.
3. Calcola la pseudo perdita di h_t :

$$\varepsilon_t = \frac{1}{2} \sum_{(i, y) \in B} D_t(i, y) (1 - h_t(x_i, y_i) + h_t(x_i, y))$$

4. Setta $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
5. Aggiorna la distribuzione D_t :

$$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \cdot \beta_t^{(1/2)(1+h_t(x_i, y_i)-h(x_i, y))}$$

dove Z_t è una costante di normalizzazione scelta in modo che D_{t+1} sia una distribuzione.

Output: l'ipotesi finale:

$$h_{fin}(x) = \arg \max_{y \in Y} \sum_{t=1}^T \left(\log \frac{1}{\beta_t} \right) h_t(x, y)$$

Per prima cosa, permettiamo al weakLearner di generare ipotesi molto più espresive che restituiscono un vettore in $[0, 1]^k$ al posto di una singola etichetta in Y . Intuitivamente la y -esima componente di questo vettore rappresenta un grado di confidenza (*degree of credence*) che l'etichetta corretta sia y . Le componenti con valore vicino ad 1 oppure a 0 corrispondono alle etichette considerate rispettivamente plausibili o non plausibili.

Conseguendo l'obiettivo di una maggiore espressività per il weak learner, ci ritroviamo con un requisito sulle prestazioni delle ipotesi deboli più complesso. Non utilizziamo lo stesso tipo di errore, ma introduciamo un errore più sofisticato chiamato pseudo perdita (*pseudo-loss*). Contrariamente all'errore ordinario, calcolato rispetto alla distribuzione sui campioni, la pseudo perdita è calcolata rispetto alla distribuzione sull'insieme di tutte le coppie di campioni ed etichette errate.

Manipolando questa distribuzione, l'algoritmo di *boosting* può concentrare i *weak learner* non solamente su una classificazione di tipo *boost* dei campioni, ma più specificatamente sulle etichette errate che risultano più difficili da discriminare. Notiamo che l'algoritmo di boosting AdaBoost.M2, che è basato su quest'idea, realizza il *boosting* se ogni ipotesi debole ha una pseudo perdita migliore della scelta casuale.

In modo più formale, un'etichettatura scorretta, *mislabel*, è una coppia (i, y) dove i è l'etichetta del campione di training, ed y è una etichetta errata associata con il campione i . Prendiamo B come l'insieme di tutte le *mislabel*:

$$B = \{(i, y) : i \in \{1, \dots, m\}, y \neq y_i\}$$

Una *mislabel distribution* è una distribuzione che è definita nell'insieme B di tutte le *mislabel*. Per ogni ciclo t dell'algoritmo, AdaBoost.M2, fornisce al weakLearn una *mislabel distribution*. D_t In risposta, il weakLearn, calcola un'ipotesi h_t della forma $h_t : X \times Y \in [0, 1]$. Intuitivamente interpretiamo ogni *mislabel* (i, y) come se rappresenti la domanda: "Hai predetto l'etichetta associata con il campione x_i come y_i (etichetta corretta) o y (una delle etichette errate)? "

Con questa interpretazione il peso $D_t(i, y)$ assegnato a questa *mislabel* rappresenta l'importanza di distinguere l'etichetta y errata sul campione x_i .

Un'ipotesi debole h_t viene in seguito interpretata nel modo seguente.

Se $h_t(x_i, y_i) = 1$ e $h_t(x_i, y) = 0$ allora h_t ha predetto correttamente che questa x_i -esima è y_i , non y , poichè da h_t riteniamo che y_i può essere plausibile ed y non plausibile. Similmente se $h_t(x_i, y_i) = 0$ e $h_t(x_i, y) = 1$ allora h_t ha fatto, in modo non corretto, la previsione opposta. Se $h_t(x_i, y_i) = h_t(x_i, y)$, allora la h_t -esima previsione è presa con decisione casuale. I valori per h_t presi nell'intervallo $(0, 1)$ sono interpretati probabilisticamente.

Questa interpretazione ci porta a definire la pseudo perdita dell'ipotesi h_t rispetto alla *mislabel distribution* D_t con la formula

$$\epsilon_t = \frac{1}{2} \sum_{(i, y) \in B} D_t(i, y) (1 - h_t(x_i, y_i) + h_t(x_i, y)) \quad (\text{B.3})$$

La derivazione completa di questa formula è riportata in [41]. Risulta chiaro che la pseudo perdita è minimizzata quando l'etichetta corretta y_i acquista valori vicini ad 1 e le etichette errate $y \neq y_i$ hanno valore vicino a 0. Inoltre notiamo che la pseudo perdita $1/2$ è banalmente acquisita da qualunque valore costante dell'ipotesi h_t , se si ha un'ipotesi h_t con valore più alto di $1/2$ è possibile rimpiazzarla con l'ipotesi $1 - h_t$

in cui la pseudo perdita è minore di $1/2$. L'obiettivo dei weakLearn è di trovare un ipotesi debole h_t con una bassa pseudo perdita.

Dopo aver ricevuto h_t la mislabel distribution viene aggiornata con una regola simile a quella utilizzata in AdaBoost.M1. L'output dell'ipotesi finale h_{fin} per una data istanza x è l'etichetta y che massimizza una media pesata dei valori delle ipotesi deboli $h_t(x, y)$.

Il teorema 2.2 assegna un limite all'errore di apprendimento, *training error*, dell'ipotesi finale. È da notare che, questo teorema, richiede solo che le ipotesi deboli abbiano pseudo perdita minore di $1/2$, per esempio, solo leggermente meglio di un'ipotesi banale con valore costante. che non badi al numero delle classi. Inoltre, sebbene le ipotesi deboli sono valutate rispetto la pseudo perdita, valutiamo l'ipotesi finale usando la misura dell'errore ordinario.

Teorema B.2. *Supponiamo di avere un algoritmo di apprendimento debole weakLearn, che, chiamato dall'algoritmo AdaBoost.M1, genera delle ipotesi con errore $\epsilon_1, \dots, \epsilon_T$, dove ϵ_t è definito come nell'algoritmo illustrato in precedenza. Prendiamo $\gamma_t = 1/2 - \epsilon_t$. Allora il seguente upper bound sull'errore dell'ipotesi finale h_{fin} è*

$$\frac{1}{m} |\{i : h_{fin}(x_i) \neq y_i\}| \leq (k-1) \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2} \leq (k-1) \exp\left(-2 \sum_{t=1}^T \gamma_t^2\right)$$

dove k è il numero delle classi.

Appendice C

Principal Component Analysis

C.1 Introduzione

La *Principal Component Analysis* (PCA) è una procedura matematica che usa una trasformazione ortogonale per convertire un insieme di osservazioni di variabili possibilmente correlate in un insieme di valori indipendenti fra loro chiamati *principal components*. Il numero di *principal components* è minore o uguale al numero di variabili originarie. Questa trasformazione è definita in modo che la prima *principal component* abbia la più alta varianza possibile e ciascuna delle *components* successive abbia una varianza via a via minore, mantenendo il vincolo che sia ortogonale (incorrelata) con quella che la precede. È garantito l'indipendenza delle *principal components* solo se il *data set* è caratterizzato da distribuzione gaussiana e indipendenza delle variabili. La PCA è sensibile alla riduzione relativa delle variabili. A seconda del campo di applicazione viene anche chiamata Trasformata discreta di Karhunen-Loève (KLT), trasformata di Hotelling o *proper orthogonal decomposition* (POD). Uno dei maggiori studi su questa tecnica è stato presentato da I. T. Joliffe in [43].

C.2 Singular Value Decomposition

Chiamiamo M una matrice $m \times n$, i cui valori derivano dall'insieme K , dei numeri reali o dei complessi. Esiste una fattorizzazione di forma:

$$M = U\Sigma V^* \quad (\text{C.1})$$

Dove U è una matrice unitaria $m \times m$ su K , la matrice Σ è una matrice diagonale $m \times n$ di numeri reali non negativi sulla diagonale e V^* , matrice unitaria $n \times n$ su K , rappresenta la trasposta coniugata di V . Questa fattorizzazione è chiamata *Singular Value Decomposition* (SVD) di M .

I valori sulla diagonale $\Sigma_{i,i}$ di Σ sono conosciuti come “singular values” di M . È convenzione comune elencare i singular values in ordine decrescente. In questo caso, la matrice diagonale Σ è unicamente determinata da M (mentre U e V non lo sono).

Le m colonne di U e le n colonne di V sono chiamate rispettivamente “left singular vectors” e “right singular vectors” di M .

SVD e auto-decomposizione sono strettamente correlate in quanto:

- I “left singular vectors” di M sono autovettori di MM^* .
- I “right singular vectors” di M sono autovettori di M^*M .
- I “singular values” non nulli di Σ sono le radici quadrate degli autovalori non negativi di M^*M o MM^* .

Per ulteriori dettagli si faccia riferimento al libro [44] e agli articoli [45, 46, 47].

C.3 Principal Component Analysis

La PCA è matematicamente definita come una trasformazione lineare ortogonale che trasforma i dati di input in un nuovo sistema di coordinate in modo che la maggior varianza di ciascuna proiezione dei dati venga a trovarsi sulla prima coordinata (chiamata *first principal component*), la seconda maggiore varianza sulla seconda coordinata e così via.

Definendo una matrice di dati, X , $X \in \mathbb{R}^{n \times m}$, con media nulla (la media empirica della distribuzione viene sottratta dal data set), dove ciascuna delle n righe rappresenta una diversa ripetizione dell'esperimento, e ognuna delle m colonne è un particolare campionamento in punto preciso.

La *singular value decomposition* di X è $X = W\Sigma V^T$, dove la matrice $m \times m$ W è la matrice degli autovettori di XX^T , la matrice Σ è una matrice $m \times n$ diagonale e rettangolare composta da numeri reali non negativi sulla diagonale, e la matrice V $n \times n$ è la matrice degli autovettori di $X^T X$. La notazione X^T si riferisce alla trasposta della matrice X . La trasformazione PCA che preserva le dimensioni, ovvero restituisce lo stesso numeri di *principal components* quante sono le variabili originarie, è data da:

$$\begin{aligned} Y^T &= X^T W \\ &= V \Sigma^T W^T W \\ &= V \Sigma \Sigma^T \end{aligned} \quad (C.2)$$

V non è unicamente definito solo per il caso in cui $m < n - 1$, ma nella maggior parte dei casi lo sarà. Fino a che W , come definito dalla SVD di una matrice reale, è una matrice ortogonale, ciascuna riga di Y^T è semplicemente una rotazione delle corrispondenti righe di X^T . La prima colonna di Y^T è costituita dai "punteggi" che rispecchiano la prima delle principal components, quella successiva dai punteggi della seconda principal component, e così via.

Se si vuole una rappresentazione di dimensioni ridotte, si può proiettare X nello spazio ridotto definito solo dai primi L vettori singolari, W_L :

$$Y = W_L^T X = \Sigma_L V^T \quad (C.3)$$

dove $\Sigma_L = I_{L \times m} \Sigma$

con $I_{L \times m}$ la matrice identità $L \times m$ rettangolare.

La matrice W dei vettori singolari di X è equivalente alla matrice degli autovettori della matrice della covarianza $C = XX^T$ come espresso in C.4 :

$$XX^T = W \Sigma \Sigma^T W^T \quad (C.4)$$

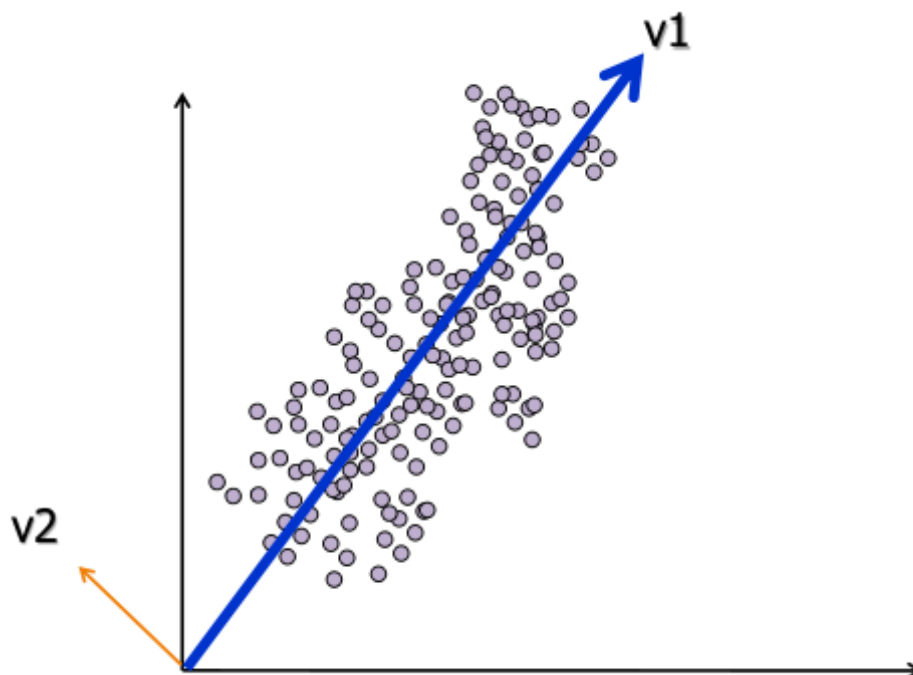


Figura C.1: PCA: allineamento degli assi con le *principal components*

Dato un set di punti nello spazio euclideo, la prima *principal component*, corrispondente alla linea blu nell'immagine C.1, corrisponde ad una linea che passa attraverso la media multidimensionale e minimizza la somma dei quadrati delle distanze dei punti dalla linea. La seconda *principal component*, indicata come una freccia rossa in figura C.1 corrisponde allo stesso concetto dopo che tutta la correlazione con la prima principal component è stata sottratta dai punti.

I singular value (in Σ) sono la radice quadrata degli autovalori della matrice XX^T . Ciascun autovalore è proporzionale alla porzione di “varianza”, in modo più corretto, alla somma dei quadrati delle distanze dei punti dalla loro media multidimensionale, che è correlata con ciascun autovettore. La somma di tutti gli autovalori è uguale alla

somma dei quadrati delle distanze dei punti dalla loro media multidimensionale. La PCA, essenzialmente, ruota l'insieme di punti attorno alla sua media in modo da allinearli con la *principal component* e si muove tanto quanto è la varianza, usando una trasformazione ortogonale, nelle prime dimensioni. I valori nelle rimanenti dimensioni, quindi, tendono ad essere piccoli e possono essere eliminati con una minima perdita di informazione. La PCA è spesso usata in tal modo per ottenere una riduzione di dimensioni del problema; ha la distinzione di diventare la trasformazione ortogonale ottima per mantenere il sottospazio che ha più grande “varianza” (come definito sopra). Questo vantaggio, però, richiede maggiori requisiti computazionali se comparato, per esempio quando è applicabile, alla trasformata discreta del coseno. Le tecniche di riduzione non lineari, invece, tendono ad essere più computazionalmente esigenti della PCA.

La PCA è sensibile allo “scaling” delle variabili. Se si hanno solo due variabili e queste hanno la stessa varianza campione e sono correlate positivamente, la PCA comporterà una rotazione di $\pi/4$ e il peso delle due variabili rispetto alla *principal component* sarà uguale. Ma se si moltiplicano tutti i valori della prima variabile per 100, la *principal component* sarà uguale alla direzione di quella variabile, con un minimo contributo delle altre variabili, mentre la seconda componente sarà per lo più allineata con la seconda variabile originale. Significa che ogni volta che le differenti variabili hanno diverse unità di misura (come temperatura e massa), la PCA è un metodo di analisi piuttosto arbitrario. Un modo per avere una PCA meno arbitraria è utilizzare variabili scalari in modo da avere varianza unitaria.

Bibliografia

- [1] Jerry. Ratcliffe. *Video surveillance of public places [electronic resource] / by Jerry Ratcliffe*. U.S. Dept. of Justice, Office of Community Oriented Policing Services, Washington, D.C. :, 2006.
- [2] M. Gill and A. Spriggs. *Assessing the impact of CCTV*. Home Office research studies. Home Office Research, Development and Statistics Directorate, 2005.
- [3] M. Bertozzi, A. Broggi, E. Cardarelli, R.I. Fedriga, L. Mazzei, and P.P. Porta. Viac expedition toward autonomous mobility [from the field]. *Robotics Automation Magazine, IEEE*, 18(3):120 –124, sept. 2011.
- [4] Tao Zhao, R. Nevatia, and Fengjun Lv. Segmentation and tracking of multiple humans in complex situations. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–194 – II–201 vol.2, 2001.
- [5] Tao Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II – 459–66 vol.2, june 2003.
- [6] B. Antic, D. Letic, D. Culibrk, and V. Crnojevic. K-means based segmentation for real-time zenithal people counting. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2565 –2568, nov. 2009.

- [7] Antonio Albiol, Inmaculada Mora, and Valery Naranjo. Real-time high density people counter using morphological tools. *IEEE Transactions on Intelligent Transportation Systems (ISSN 1524- 9050)*, 2(4):204–217, 2001.
- [8] H. Nanda and K. Fujimura. A robust elliptical head tracker. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 469 – 474, may 2004.
- [9] Tim van Oosterhout, Sander Bakkes, and Ben J. A. Kröse. Head detection in stereo data for people counting and segmentation. In *VISAPP*, pages 620–625, 2011.
- [10] D. Beymer. Person counting using stereo. In *Human Motion, 2000. Proceedings. Workshop on*, pages 127 –133, 2000.
- [11] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for easyliving. pages 3–10, 2000.
- [12] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision*, 37:175–185, 2000. 10.1023/A:1008103604354.
- [13] Philip Kelly, Noel E. O’Connor, and Alan F. Smeaton. Robust pedestrian detection and tracking in crowded scenes. *Image Vision Comput.*, 27:1445–1458, September 2009.
- [14] William K. Pratt and James E. Adams Jr. Digital image processing, 4th edition. *J. Electronic Imaging*, 16(2):29901, 2007.
- [15] Liang Zhao and Chuck Thorpe. Recursive context reasoning for human detection and parts identification. In *In IEEE Workshop on Human Modeling, Analysis and Synthesis*, 2000.

- [16] Mirko Felisa. *Studio e applicazione di algoritmi stereoscopici per la ricostruzione tridimensionale in ambiente automotive*. PhD thesis, Facoltà di Ingegneria, Università di Parma, April 2010.
- [17] Mirko Felisa and Paolo Zani. Incremental Disparity Space Image computation for automotive applications. In *Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, St.Louis, Missouri, USA, October 2009. submitted.
- [18] T. Gandhi and M.M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *Intelligent Transportation Systems, IEEE Transactions on*, 8(3):413–430, sept. 2007.
- [19] T. Gandhi and M.M. Trivedi. Image based estimation of pedestrian orientation for improving path prediction. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 506–511, june 2008.
- [20] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri, and Ho Gi Jung. A new approach to urban pedestrian detection for automatic braking. *Intelligent Transportation Systems, IEEE Transactions on*, 10(4):594–605, dec. 2009.
- [21] L. Leyrit, T. Chateau, C. Tournayre, and J.-T. Lapreste. Association of adaboost and kernel based machine learning methods for visual pedestrian recognition. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 67–72, june 2008.
- [22] M. Bertozzi, A. Broggi, M. Del Rose, M. Felisa, A. Rakotomamonjy, and F. Suard. A pedestrian detector using histograms of oriented gradients and a support vector machine classifier. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 143–148, 30 2007-oct. 3 2007.
- [23] M. Enzweiler and D.M. Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, dec. 2009.
- [24] D. Geronimo, A.M. Lopez, A.D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1239–1258, july 2010.

- [25] Zehang Sun, G. Bebis, and R. Miller. Monocular precrash vehicle detection: features and classifiers. *Image Processing, IEEE Transactions on*, 15(7):2019–2034, july 2006.
- [26] Zehang Sun, G. Bebis, and R. Miller. On-road vehicle detection: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711, may 2006.
- [27] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [28] Alan M. Turing. Computing Machinery and Intelligence. *Mind*, LIX:433–460, 1950.
- [29] Nello Cristianini and John Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University press, 2003.
- [30] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall / Upper Saddle River, 2002.
- [31] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
- [32] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.
- [33] D. Meyer. The support vector machine under test. *Neurocomputing*, 55(1-2):169–186, September 2003.

- [34] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification, 2000.
- [35] Kai-Bo Duan and Sathya S. Keerthi. Which Is the Best Multiclass SVM Method? An Empirical Study. In *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 278–285. Springer, 2005.
- [36] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines, 2002.
- [37] John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.
- [38] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *CoRR*, cs.AI/9501101, 1995.
- [39] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002.
- [40] Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, 14(5):771–780, 1999.
- [41] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.
- [42] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
- [43] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

-
- [44] Trefethen and N. Lloyd. *Numerical linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics., 1997.
 - [45] Jam Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. SCI. STAT. COMPUT*, 11(5):873–912, 1990.
 - [46] GOLUB G. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2:205–224, 1965.
 - [47] Michael Wall, Andreas Rechtsteiner, and Luis Rocha. Singular value decomposition and principal component analysis. In Daniel P. Berrar, Werner Dubitzky, and Martin Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 91–109. Springer US, 2003. 10.1007/0-306-47815-3.5.