Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica Applicata

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione

XXIX Ciclo

TESI DI DOTTORATO

Rilevamento e Conteggio Automatico di Persone da Telecamera in Posizione Zenitale

CANDIDATO: **ING. LUCA DEL PIZZO**

TUTOR: **PROF. GENNARO PERCANNELLA**

COORDINATORE: **PROF. ALFREDO DE SANTIS**

Anno Accademico 2016 – 2017

Dove c'è una grande volontà non possono esserci grandi difficoltà.

(Nicolò Machiavelli)

Alle persone più care...

Abstract

Negli ultimi anni, la crescente attenzione nei confronti della sicurezza in luoghi pubblici e privati ha portato la comunità scientifica ad esercitare un considerevole sforzo finalizzato a fornire efficaci sistemi di videosorveglianza volti a monitorare automaticamente lo scenario di interesse. Questo ambito è fortemente caratterizzato dal diffondersi di diverse applicazioni che si incentrano sull'osservazione delle persone presenti nella scena al fine di rilevarne i movimenti. Nel presente lavoro si propongono metodi di visione artificiale per il rilevamento ed il conteggio automatico di persone inquadrate da telecamera fissa installata in posizione zenitale. I metodi proposti sono specificamente progettati per raggiungere un elevato livello di accuratezza ed efficienza computazionale in scenari reali. Una valutazione estesa dei metodi viene effettuata su un *dataset* appositamente realizzato a partire dallo studio dei fattori principali che possono influire sulle prestazioni dei metodi, in particolare, la tecnologia di acquisizione (telecamera tradizionale e sensore di profondità), lo scenario di installazione (ambiente interno ed esterno) e la densità di persone presenti nella scena. Il *dataset* realizzato è pubblicamente disponibile per la comunità scientifica. I risultati sperimentali confermano l'efficacia dei metodi proposti anche in confronto a soluzioni presenti in letteratura e la possibilità di impiego sia su server di fascia alta che su architetture con ridotte risorse computazionali, come le *smart camera* presenti sul mercato.

Sommario

Indice delle figure

Indice delle tabelle

Acronimi

QoS	Quality of service
VCA	Video content analysis
WLAN	Wireless local area network
ASIC	Application specific integrated circuit
FPGA	Field Programmable Gate Array
SOC	System on a Chip
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
BS	Background Subtraction
ABS	Adaptive Background Subtraction
FSM	*Frame* Difference Method
SIMD	Single Instruction Multiple Data
IIR	Infinite Impulse Response
CUDA	Compute Unified Device Architecture
FPS	*Frame* per second
VMS	Video Management Software
DSP	Digital Signal Processor
SDRAM	Synchronous Dynamic Random Access Memory
GCC	GNU Compiler Collection
API	Application Programming Interface
FPU	Floating-point unit
VGA	Video Graphics Array
QVGA	Quarter Video Graphics Array
CGI	Common Gateway Interface
FTP	File Transfer Protocol

SDK	Software Development Kit
EAP	Executable Axis Packet
IoT	Internet of Things
TOF	Time of Flight
SVM	*Support Vector Machines*
MJPEG	Motion Joint Photographic Experts Group
RTSP	Real Time Streaming Protocol
MAC	Media Access Control
VIPER	Video Performance Evaluation Resource
OpenCV	Open Source Computer Vision Library
CMOS	Complementary Metal-Oxide-Semiconductor
HTML	HyperText Markup Language
SSI	Server Side Includes
PoE	Power over Ethernet
ISP	Internet Service Provider
JSON	JavaScript Object Notation
ML	Machine Learning
GUI	Graphical User Interface
PIR	*Passive Infra Red*

Introduzione

Il presente studio di tesi ha come obiettivo la progettazione, la realizzazione e la caratterizzazione sperimentale di metodi di visione artificiale per il **rilevamento ed il conteggio automatico di persone** inquadrate da telecamere, tradizionali e di profondità, installate in posizione fissa e zenitale. Rispetto all'approccio tradizionale, i metodi proposti, oltre a presentare un'elevata accuratezza in ambienti reali (al variare dello scenario e della densità di persone nella scena) possono operare anche su piattaforme con ridotte risorse computazionali (sistemi *embedded* presenti sul mercato). Inoltre, è stata realizzata e resa disponibile alla comunità scientifica una collezione di immagini (*dataset*) con annessa *ground truth,* per l'ambito di riferimento, con caratteristiche prossime agli scenari reali (varietà della tipologia di dispositivi di acquisizione, dello scenario di installazione e della densità di persone), con la quale è stata completata un'attività sperimentale per caratterizzare le prestazioni dei metodi proposti di rilevamento e conteggio delle persone.

1.1 Motivazione e descrizione del problema

In un sistema tradizionale, il controllo ed il monitoraggio video di un'area sono attività demandate ad uno o più operatori che hanno il compito di visionare un certo numero di monitor alla ricerca di eventi (Figura 1). Tale attività porta ad un calo fisiologico di attenzione con conseguente mancato rilevamento di situazioni di allarme. In questo scenario, risultano di fondamentale importanza i **sistemi di analisi video intelligente** poiché rappresentano un valido strumento di supporto al monitoraggio per le Amministrazioni Pubbliche, per i privati e per le Forze dell'Ordine in uno scenario sempre più complesso. Tali sistemi si basano su algoritmi di *computer vision* (CV), o visione artificiale, la disciplina che studia i modelli ed i

metodi per rendere le macchine abili alla comprensione ed all'interpretazione delle informazioni visuali presenti in immagini o sequenze video acquisite tramite telecamere analogiche o digitali. La visione artificiale vuole emulare la visione biologica non solo in termini di acquisizione di una immagine, ma soprattutto per quanto concerne l'interpretazione del contenuto, riproducendo sui calcolatori elettronici il percorso cognitivo compiuto dall'essere umano nell'interpretazione della realtà che lo circonda. Per cui, l'analisi video intelligente costituisce un complesso di tecniche di elaborazione di flussi video acquisiti da telecamera, con la finalità di rilevare in tempo reale eventi di interesse all'interno di una scena e di interpretarne il contenuto. In particolare, l'analisi video consente di riconoscere oggetti o individui in movimento (*object detection*), eventualmente di caratterizzarne dimensione, forma e colore, di calcolarne la traiettoria (*object tracking*) e di rilevare i potenziali eventi presenti nella scena (*behavioral analysis*). Un sistema di analisi video intelligente consente di segnalare ad un operatore il verificarsi di un evento di interesse (approccio *event-based*), con la possibilità di visualizzare i soli istanti di tempo associati ad esso.

Figura 1 - Sistema di videosorveglianza tradizionale

La CV è un campo in rapida crescita, favorito dall'evoluzione tecnologica a cui stiamo assistendo e dall'intensa attività di ricerca da parte della comunità scientifica, che dai primi anni Ottanta ad oggi ha portato ad una crescita esponenziale delle pubblicazioni in questo settore. Tra le principali difficoltà riscontrabili nella CV si registra la dipendenza della precisione dei risultati da diversi fattori, come la risoluzione della telecamera, lo scenario di riferimento, il *frame rate* di acquisizione, la configurazione dell'algoritmo, la tipologia di telecamera utilizzata.

Tra i numerosi ambiti di applicazione della CV, quello riguardante il riconoscimento ed il conteggio di persone all'interno della scena è oggetto della presente tesi. L'obiettivo è quello di localizzare gli individui e di conteggiare quanti attraversano un varco di ingresso-uscita (ricavando anche la direzione di attraversamento). Gli algoritmi di analisi video consentono di superare le limitazioni dei sistemi tradizionali (sensori ad infrarossi) ma devono affrontare diverse problematiche che si aggiungono a quelle generiche della CV definite in precedenza, come le occlusioni, la presenza di oggetti aggiuntivi di medie/grandi dimensioni, i differenti stili di abbigliamento e di postura delle persone. Un altro aspetto da valutare nella fase di progettazione, considerato secondario in diverse soluzioni presenti in letteratura, è la complessità computazionale del metodo, la quale, deve consentire l'elaborazione dell'algoritmo in *real-time* anche su piattaforme con risorse limitate.

1.2 Ambiti applicativi

Le informazioni su rilevamento, localizzazione e conteggio delle persone nella scena sono indispensabili in diverse applicazioni nell'ambito *security*. Si pensi ad un sistema il cui obiettivo è quello di:

- conoscere in ogni istante il numero di presenze in una struttura al fine di ottemperare alle prescrizioni dettate dal piano di emergenza e sicurezza (Figura 2);
- rilevare la presenza di persone in zone ad accesso limitato (anti-intrusione);
- identificare situazioni anomale, come la presenza di un individuo che si muove in direzione opposta all'andamento di una folla o che staziona per un lungo periodo in una specifica area senza apparente motivo (Figura 3).

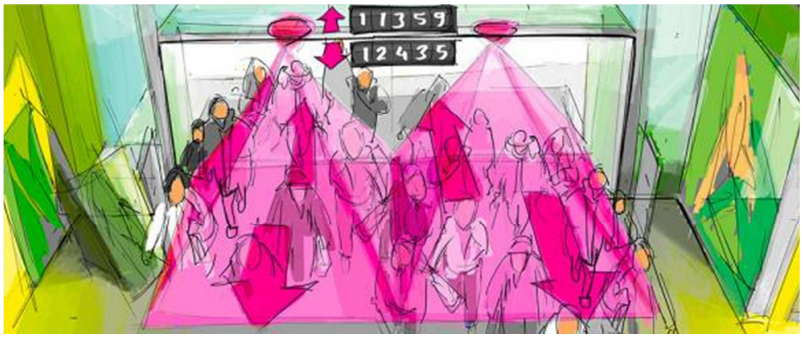


Figura 2 – Conteggio dei flussi presso un varco di ingresso e uscita

Figura 3 – Rilevamento e inseguimento delle persone (figura di sinistra) e rilevamento di intrusi (figura di destra)

Un altro ambito applicativo riguarda la *business intelligence* (BI): un sistema di analisi video raccoglie dati ed analizza informazioni strategiche, al fine di esaminare la storia passata e di stimare le

prestazioni future del *business*. Il *retail* è il settore della BI con il maggior utilizzo di algoritmi di analisi video intelligente. Essi ad esempio estraggono informazioni relative al comportamento degli utenti di un punto vendita al fine di conoscere, in ogni istante, gli individui presenti all'interno dell'area, di monitorare i varchi di accesso e di stimare il tempo di permanenza dei clienti (Figura 4). Questo metodo consente di ricavare molteplici statistiche utili ad ottimizzare la linea di *marketing*, l'utilizzo del personale e la *customer satisfaction.*

Figura 4 – Analisi video e Business Intelligence: in rosso le aree in cui si registra un maggior tempo di permanenza da parte delle persone

1.3 Trend tecnologico

Il quinquennio corrente registra costi sempre più contenuti delle telecamere ed un significativo miglioramento della risoluzione dei dispositivi con un'elevata qualità delle immagini; inoltre delinea il completo passaggio dai dispositivi analogici ai digitali, permettendo di digitalizzare l'intera infrastruttura del sistema video (Figura 5).

Figura 5 - Sistema video analogico e digitale

Ad oggi, la maggior parte dei sistemi di analisi video (analogici o digitali) opera in modalità **server-side** (Figura 6). Le telecamere sono connesse ad un centro di elaborazione (*server*) il quale, cattura i flussi video attraverso il sistema di trasmissione dedicato ed elabora le immagini (tramite algoritmi di analisi video intelligente), al fine di rilevare e segnalare eventi di interesse (ad esempio intrusioni e conteggio persone).

Figura 6 – Sistema server-side

Il principale vantaggio di tale approccio, risiede nella possibilità di aggiungere intelligenza ad un qualsiasi sistema video preesistente, indipendentemente dalla tipologia di telecamere installate. Le maggiori risorse di calcolo disponibili su un server garantiscono una maggiore flessibilità in termini di numero di funzioni di analisi attivabili contemporaneamente sullo stesso flusso video. Inoltre, vi è la possibilità di stabilire il miglior compromesso tra il numero di flussi video elaborati in parallelo e la coppia risoluzione-*frame rate* con cui sono elaborate le immagini. Il sistema appena descritto presenta i seguenti vincoli:

- il flusso video trasferito deve essere ad alta risoluzione e deve presentare un *frame* rate sufficiente per poter applicare in maniera efficace ed efficiente gli algoritmi di analisi video;
- la rete necessita di banda e latenza sufficienti a supportare i flussi di tutte le telecamere connesse;
- il server è il punto critico del sistema, quindi è necessario prevedere una ridondanza che comporta un aumento dei costi.

I principali problemi della suddetta architettura sono:

- necessità di maggiore banda trasmissiva all'aumentare dei dispositivi connessi;
- maggiori costi per la rete ed il server;
- problemi di privacy legati alla trasmissione e memorizzazione del video secondo la normativa vigente (D.Lgs. 196/2003).

In considerazione di ciò, diverse aziende (ad esempio la *Axis Communications*) stanno investendo sulla ***smart camera:*** un *sistema embedded* che consente di elaborare il flusso in modalità **edge-side**, ovvero direttamente sull'hardware integrato della telecamera (analisi intelligente a bordo camera). Tale sistema consente di trasmettere al server solo gli eventi rilevanti e se richiesto, le immagini che hanno determinato la generazione dell'evento (Figura 7). Questa tecnologia permette di avere un'occupazione di banda minima, di eliminare il *single point of failure* e di ridurre (o eliminare) il problema della privacy poiché non avviene la trasmissione e la memorizzazione dell'intero flusso video, ma della coppia evento-*frame* (o dei soli eventi di interesse). Con la diminuzione del carico, le reti *WLAN* possono supportare un numero elevato di sensori connessi, eliminando i problemi rilevati nei casi in cui la connettività cablata è assente, limitata o comunque insufficiente per l'estesa distribuzione dei punti di monitoraggio sul territorio. Con l'utilizzo dei sistemi *embedded*, il centro di elaborazione (server) diviene un gestore degli eventi inviati dalle singole telecamere, difatti non effettua nessuna elaborazione sul flusso video. In particolare, il server si limita a catturare tutti gli eventi ed a gestirli al fine di ottenere un'informazione di livello più alto, ad esempio inviando una segnalazione sul cellulare dell'interessato in seguito al rilevamento di un intruso da parte di una o più telecamere (Figura 7).

Figura 7 - Soluzione *edge-side*

Di *contro*, i sistemi *embedded* presentano una limitata potenza di calcolo, una ridotta memoria disponibile e dei tempi di sviluppo maggiori a causa dell'assenza di librerie di alto livello per l'*image processing*. Quanto detto, porta ad affermare che per utilizzare al meglio i sistemi *embedded* nell'ambito dell'analisi video, è richiesta una **approfondita ingegnerizzazione degli algoritmi presenti in letteratura**, riducendo l'onere computazionale e della memoria, con una gestione a basso livello dei *pixel*, sfruttando le caratteristiche delle architetture *embedded* moderne. Un ulteriore elemento da considerare è che le architetture dei processori dei diversi brand di telecamere che supportano l'esecuzione *embedded,* spesso hanno caratteristiche differenti, il che rende più complesso lo sviluppo di software per tali sistemi.

Negli ultimi anni, la progressiva miniaturizzazione dei componenti hardware, la diminuzione dei costi delle unità di calcolo ed il basso consumo energetico hanno consentito di allargare lo spettro di applicazione dei sistemi *embedded*. Questa tendenza conduce ai **sistemi di analisi video *embedded*** (ad es. la *smart camera* descritta in precedenza), i quali incorporano a bordo proprie capacità di elaborazione visiva, con algoritmi specializzati per l'interpretazione dei contenuti informativi di immagini e video, in grado di comprendere le caratteristiche dell'ambiente in cui operano (elaborazione *edge-side*).

1.4 Struttura della tesi

Il resto del documento è strutturato come segue.

Nel Capitolo 1 si presentano i risultati di un approfondito studio della letteratura scientifica riguardante gli algoritmi di *moving object detection* con l'utilizzo di telecamere fisse. Tali metodi rappresentano il primo importante passo che accomuna numerosi algoritmi di analisi video intelligente, il cui obiettivo è quello di individuare i *pixel* che appartengono agli oggetti di interesse, in movimento nella scena. L'analisi della letteratura si rivolge soprattutto agli algoritmi progettati per sistemi con ridotte capacità elaborative e di memoria, al fine di ridurre la complessità spaziale e/o temporale.

Nel Capitolo 2 si descrive un'attività progettuale e sperimentale finalizzata all'ingegnerizzazione di un algoritmo di *moving object detection,* ampiamente utilizzato in letteratura, al fine di consentire la sua esecuzione su dispositivi con risorse limitate. Per lo sviluppo di tale attività, sono state definite ed implementate diverse ottimizzazioni volte a ridurre l'onere computazionale dell'algoritmo. Le ottimizzazioni introdotte sono state oggetto di un'approfondita attività sperimentale destinata a caratterizzare le prestazioni del metodo su diverse famiglie di processori disponibili sulle *smart camera* commerciali. I risultati ottenuti dimostrano che le ottimizzazioni introdotte consentono la esecuzione del metodo sui sistemi reali presenti sul mercato.

Nel Capitolo 3 si delinea un metodo per il conteggio delle persone che attraversano una linea virtuale mediante la elaborazione del flusso video in tempo reale da una telecamera installata in posizione zenitale. Si riporta inoltre la caratterizzazione delle prestazioni sia dal punto di vista dell'accuratezza che dei requisiti computazionali: il metodo è progettato per raggiungere una elevata accuratezza (sia in scenari interni che esterni) con un utilizzo ridotto delle risorse computazionali e di memoria. La valutazione del metodo è stata effettuata tenendo in

considerazione diversi fattori che possono avere un significativo impatto sulle prestazioni: il tipo di sensore impiegato per l'acquisizione del flusso video, lo scenario di riferimento, la densità degli attraversamenti, il *frame* rate di acquisizione e la risoluzione del *frame*. Per sperimentare l'efficacia e l'efficienza della soluzione è stato realizzato un *dataset* relativo al problema in esame che prende in considerazione i predetti fattori. In particolare, utilizzando una telecamera ottica tradizionale ed un sensore di profondità, entrambi installati in posizione zenitale. Esso è caratterizzato da diversi flussi di attraversamento e da scenari interni ed esterni. La sperimentazione su differenti architetture hardware conferma non solo l'elevata accuratezza della soluzione proposta, ma anche la possibilità di utilizzare il metodo sia in modalità *server-side* che *edge-side*. Il *dataset* realizzato è pubblicamente disponibile per la comunità scientifica.

Nel Capitolo 4 si propone un metodo per il rilevamento e la relativa localizzazione delle persone presenti nella scena inquadrata da telecamera di profondità installata in posizione zenitale. Tale metodo è progettato per raggiungere una elevata accuratezza con un ridotto utilizzo delle risorse computazionali e una configurazione semplificata. La sperimentazione condotta sul *dataset* descritto in precedenza conferma il raggiungimento di una notevole precisione pur utilizzando operazioni non complesse, in confronto ad altri due metodi presenti in letteratura che affrontano lo stesso problema ma con approcci diversi e più articolati. Si presenta inoltre il processo di combinazione di tre metodi di rilevamento (il metodo proposto e i due metodi di confronto) attraverso la realizzazione di un sistema multi-esperto con regole di combinazione per il dominio in questione. La sperimentazione attesta l'efficacia del sistema multi-esperto, il quale raggiunge prestazioni più elevate rispetto ai singoli esperti al variare dello scenario e della densità di persone presenti nella scena.

Nel Capitolo 5 si descrivono le librerie utilizzate per l'implementazione delle soluzioni proposte e le caratteristiche tecniche delle telecamere impiegate nel corso delle diverse sperimentazioni.

Per concludere si riportano le considerazioni finali del lavoro realizzato.

Capitolo 1. Algoritmi di Moving Object Detection

Nell'ultima decade si evidenzia un massiccio incremento delle istallazione dei sistemi di video soprattutto in ambito videosorveglianza. In questi sistemi l'analisi video intelligente si utilizza per accrescere il livello di sicurezza e per scopi di business. Inoltre, al fine di superare le limitazioni di una soluzione *server-side,* sono sempre più richieste architetture *edge-side* tramite l'utilizzo delle *embedded smart camera*, le quali oltre a generare l'immagine includono un processore, una memoria e interfacce di comunicazione che permettono di elaborare il flusso video direttamente a bordo camera, in modo da memorizzare e/o inviare al server solamente gli eventi di interesse e i relativi *frame*. Tuttavia, i sistemi *embedded* sono progettati per avere risorse computazionali e di memoria limitate (ad esempio non presentano una *floating-point unit*). Anche le telecamere con alimentazione a batteria introducono nuove sfide, poiché nella progettazione del software si deve tener in considerazione il vincolo di un utilizzo limitato della CPU.

Detto ciò e considerando che gli algoritmi di analisi video sono computazionalmente onerosi, la sfida degli ultimi anni si indirizza nell'**ingegnerizzare** tali algoritmi al fine di consentirne l'**elaborazione in real-time sui sistemi *embedded***. Da evidenziare che, l'incremento della potenza di calcolo dei più recenti sistemi *embedded* non risolve il problema poiché è in costante aumento anche la risoluzione e la complessità delle applicazioni richieste.

L'analisi della scena dovrebbe, quindi, impiegare solo una certa percentuale della risorsa CPU disponibile e il risultato dovrebbe essere fruibile con un ritardo piccolo. Altri fattori da considerare durante la progettazione degli algoritmi sono la memoria richiesta e la portabilità della soluzione su diverse architetture.

I sistemi di analisi video, in generale, si compongono dei seguenti blocchi funzionali:

- *Moving Object Detection:* individua gli oggetti di interesse presenti nella scena, segmentando il *frame* in una regione di *foreground* (i *pixel* che appartengono agli oggetti di interesse) e una di *background*. Gli oggetti sono descritti tramite *bounding box* (il più piccolo rettangolo che contiene l'oggetto di interesse i cui lati sono paralleli a quelli del *frame*).
- *Object Tracking:* a partire dai *bounding box*, gestisce le identità degli oggetti che attraversano la scena, assegnandogli un identificativo univoco, al fine di ottenere le traiettorie di ogni oggetto.
- *Behavior Analysis:* a partire dalle traiettorie di ogni oggetto è possibile classificare il suo comportamento. Il sistema viene programmato per reagire a specifici comportamenti.

Le applicazioni di analisi video sono caratterizzate da operazione di basso, medio ed alto livello. Le prime due, a cui appartengono gli algoritmi di *moving object detection*, sono computazionalmente onerose ma teoricamente poco complesse, mentre quelle di alto livello sono meno onerose ma più complesse. Quanto detto è confermato dal lavoro di *Chen et al.* [1] il quale presenta un'analisi del carico di lavoro associato ad ogni modulo dell'analisi video (dai moduli con operazioni di basso livello fino a quelli con operazioni di alto livello) ed evidenzia come la fase di *moving object detection* sia quella più onerosa (circa il *95%*).

In accordo con la precedente analisi, in questo capitolo si riportano i risultati di un'approfondita analisi della letteratura al fine di evidenziare le ottimizzazioni proposte per ridurre la complessità spaziale e/o temporale degli algoritmi di *moving object detection* per applicazione in real-time e con *frame* provenienti da telecamera fissa.

I metodi presenti in letteratura che affrontano il problema del *moving object detection* con telecamera statica si possono classificare in due categorie:

1) *Frame Difference Methods* (FDM) [2][3]: alla differenza tra due (o tre) *frame* consecutivi viene applicata una soglia (parametro τ) per stabilire quali *pixel* appartengono ad oggetti in movimento. Anche se computazionalmente poco onerosa, tale soluzione genera errori nel caso di oggetti che si fermano rapidamente e non risulta efficace per l'estrazione della forma degli oggetti in movimento (*ghosting problem* e *foreground aperture)*

$$D_n(x,y) = \begin{cases} 1, \text{if } |F_n(x,y) - F_{n-1}(x,y)| > \tau_T \\ 0, \text{otherwise} \end{cases}$$

2) *Background Subtraction Method* (BSM) [4][5][6][7][8][9][9][11][12][13]: per stabilire quali *pixel* appartengono ad oggetti in movimento, il metodo effettua prima una classificazione, in cui il *frame* corrente viene sottratto con un predeterminato modello del background *B,* il quale rappresenta la parte statica dello scenario e deve include anche i *pixel* degli oggetti in movimento ma non di interesse (background dinamico, come ad esempio fontane e alberi). Al fine di accrescere l'efficacia del metodo, anche in considerazione di ombre, variazioni rapide di luminosità e varietà del background dinamico, è richiesta una fase di aggiornamento che riporti i cambiamenti dello scenario all'interno dello stesso modello.
   In generale, il modello del background può essere aggiornato utilizzando due tecniche: *non recursive* [42][43] e *recursive* [44][45][46]. Le tecniche *non recursive* aggiornano il background utilizzando *N frame* precedenti, mentre le tecniche *recursive* utilizzando il *frame* corrente e informazioni sulla storia del background. In *Heikkilä et al.* [46] il modello viene

aggiornato utilizzando un filtro IIR (*Infinite Impulse Response*), in particolare si parla di *Adaptive Background Subtraction Method (ABSM)* poiché il modello del background viene aggiornato come segue:

$$D_n(x,y) = \begin{cases} 1, |F_n(x,y) - B(x,y)| > \tau_T \\ 0, \text{otherwise} \end{cases}$$

$$B_{n+1}(x,y) = \alpha \cdot F_n(x,y) + (1-\alpha) \cdot B_n(x,y)$$

in cui il coefficiente α *(learning rate*) definisce la velocità di aggiornamento del modello alle variazione del scenario. La scelta del coefficiente α è cruciale per l'accuratezza della soluzione: un aggiornamento veloce induce il metodo ad includere nel modello anche oggetti statici mentre con un aggiornamento lento il metodo è particolarmente sensibile ai cambiamenti di luminosità. Una possibile soluzione per gestire i cambiamenti di luminosità è proposta in *Matsushita et al.* [47], in cui gli autori tramite l'utilizzo dei *illumination eigenspace* rendono il *frame* da analizzare indipendente dalle variazioni di illuminazione.

Nelle applicazioni di analisi video, i *BSM* sono gli algoritmi di *moving object detection* più utilizzati, quindi nel seguito si farà riferimento a questa tipologia.

1.1 Algoritmi di Background Subtraction

Gli algoritmi di background subtraction si possono sintetizzare nelle seguenti quattro fasi:

1) *Preprocessing*: il *frame* corrente viene elaborato per eliminare distorsioni indesiderate o migliorare alcune caratteristiche dell'immagine utili per le successive elaborazioni. Operazioni di *temporal* e *spatial smoothing* sono impiegate per ridurre il

rumore della telecamera (ad es. la pioggia e la nebbia delle riprese in scenario esterni). Altre operazioni incluse in questa fase sono *color conversion, region extraction* e *edge detection*.

2) *Foreground Detection:* in questa fase si confronta ogni *pixel* del *frame* corrente con il corrispondente *pixel* del modello del background. Il risultato è la classificazione di ogni *pixel* del *frame* corrente in *background pixel* o *foreground pixel*. In particolare si ottiene la cosiddetta *foreground mask* (un'immagine binaria nella quale i *foreground pixel* sono impostati a 1 mentre i *background pixel* a 0, o viceversa) la cui accuratezza dipende dalla tipologia di metodo utilizzato.
3) *Background Update*: il modello del background viene aggiornato in accordo al *frame* corrente e alla *foreground mask*. L'obiettivo di questa fase è ottenere un modello del background che sia robusto contro i cambiamenti della parte non di interesse presente nella scena, e sensibile per identificare al meglio gli oggetti di interesse in movimento.
4) *Postprocessing*: in questa fase la *foreground mask* ottenuta viene migliorata utilizzando algoritmi di *shadow detection,* per eliminare le ombre, e *morphological filtering,* per risaltare le caratteristiche degli oggetti di interesse.

In generale, le fasi precedenti lavorano su tutti i *pixel* del *frame*, per questo motivo, nel diagramma di flusso illustrato nella Figura 8 si riporta anche la fase di *blob detection* (o *connected component grouping*) che identifica le regioni (*blob*) all'interno dell'immagine mediante operazioni *pixel-based*. La fase di *blob detection* permette, inoltre, di filtrare le regioni la cui dimensione (o forma) è incompatibile con gli oggetti di interesse. Come evidenziato nella Figura 8, la fase di *background update* fornisce a quella di *foreground detection* il modello corrente, mentre in ingresso ottiene la *foreground mask* (nell'esempio i *pixel* scuri rappresenta la foreground, il sole non presente nel modello) e il *frame* corrente per aggiornare il modello da

utilizzare nel *frame* successivo. La fase di *postprocessing* affina la *foreground mask* da utilizzare nella fase di *blob detection* per la ricerca delle componenti connesse (evidenziate in figura con un quadrato rosso).

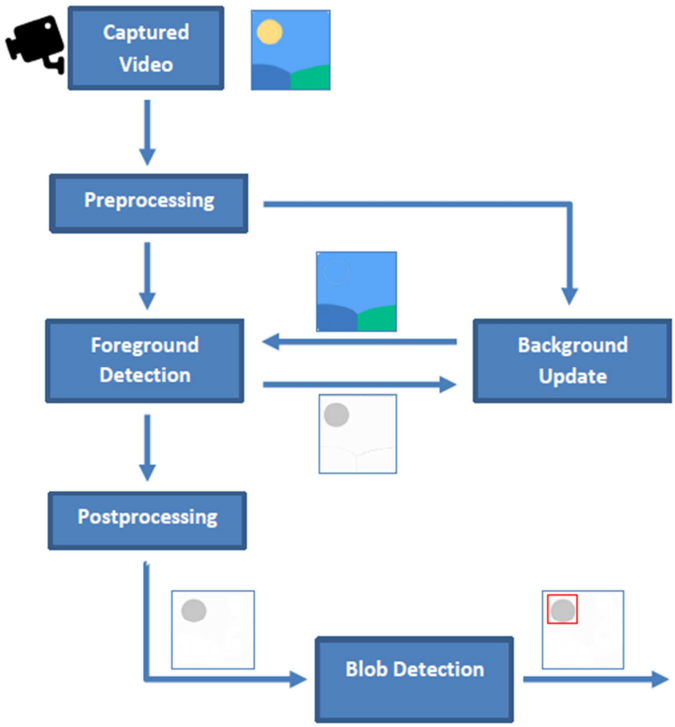


Figura 8 - Diagramma di flusso che rappresenta le fasi di un generico algoritmo di *background subtraction*

Nella progettazione di un algoritmo di *background subtraction* bisogna considerare le problematiche connesse al modello del background, in particolare quest'ultimo dovrebbe:

- essere settato senza includere oggetti in movimento;
- adattarsi ai cambiamenti di luminosità dell'ambiente (sia interno che esterno);

- includere background dinamico, come ad esempio alberi in movimento e fontane;
- eliminare le ombre;
- individuare gli oggetti in movimento che presentano caratteristiche cromatiche simili a quelle del background (*camouflage problem*);
- gestire correttamente i casi in cui gli oggetti si fermano rapidamente o ripartono dopo essere entrati a far parte del *background* (al fine di classificare l'area in cui la persona era presente un'istante prima come di background).

In letteratura sono presenti diversi metodi per risolvere il problema del *moving objects detection* con l'utilizzo degli algoritmi di *background subtraction*. I lavori [14][15][16][17][18] forniscono una panoramica dei principali metodi evidenziando vantaggi e svantaggi. Molti di questi metodi sono stati introdotti con l'obiettivo di incrementare l'accuratezza rispetto ad altri metodi presente in letteratura, senza considerare vincolanti la memoria richiesta, il costo computazionale e la portabilità della soluzione su altre architetture. Per cui, si ottengono soluzioni efficaci in termini di accuratezza, ma non utilizzabili in applicazioni dove si richiede un'elaborazione in *real-time*, soprattutto in considerazione delle piattaforme emergenti come la *smart camera*.

Li et al. [19] presentano un *statistical background modelling method* per rilevare gli oggetti in movimento presenti in uno scenario complesso, ma la quantità di memoria richiesta per poter memorizzare le tabelle delle statistiche è elevata. *Chiu et al.* [20] presentano un efficace e robusto sistema di segmentazione degli oggetti, ma richiede la memorizzazione di un numero elevato di informazioni per estrarre il background dal *frame* corrente. *Tsai e Chiu* [21] mostrano come un algoritmo di *Fourier reconstruction* possa estrarre i contorni degli oggetti in movimento, tuttavia tale soluzione ha un elevata complessità computazionale. Le tecniche *sliding window-based* (o *non*

recursive), come quella presentata da *Cheung e Kamath* [22], memorizzano gli ultimi *w frame* per accrescere l'accuratezza della soluzione, ma questo porta ad un incremento della memoria utilizzata e della complessità computazionale al fine di processare correttamente le *w* immagini. Altre soluzioni sono il *multimodal background model* come GMM (*Gaussian mixture model*), chiamato anche MoG (*Mixture of Gaussians),* proposto da *Stauffer and Grimson* [4], oppure il *texture model* proposto da *Heikkila and Pietikinen* [9]. Questi metodi lavorano bene anche in ambienti con background dinamico, ma presentano una complessità temporale e spaziale elevata. Quanto affermato è confermato anche dal lavoro di *Sobral and Vacavant* [16], in cui i risultati delle loro sperimentazioni mostrano come memoria e complessità computazionale sono elevate in particolar modo per i metodi che presentano le migliori *performance* in termini di accuratezza (*Hofmann et al.* [24], *Maddalena and Petrosino* [8], *Yao and Odobez* [25]).

1.2 Algoritmi di Background Subtraction per applicazioni real-time

Al fine di risolvere i problemi connessi alle soluzioni *server-side*, l'analisi video sta spostando l'attenzione verso soluzioni *edge-side*, con l'utilizzo della *smart camera.* Questa piattaforma consente di elaborare il flusso direttamente a bordo camera, ma presenta risorse computazionali e di memoria limitate, per cui si ha la necessità di completare un processo di ingegnerizzazione degli algoritmi presenti in letteratura al fine di permettere l'elaborazione in *real-time* su queste piattaforme.

Nel seguito si riportano le soluzioni presenti in letteratura il cui obiettivo è quello di ottimizzare un metodo di *background subtraction* per ridurne l'onere computazionale e la memoria richiesta. In particolare, l'attenzione è rivolta alle fasi di *foreground detection* e *background update* poiché le più onerose. Per completare la

trattazione si analizza la complessità computazionale e la memoria richiesta dalle diverse soluzioni in confronto ai seguenti due metodi di *background subtraction*:

1. *Stauffer and Grimson* [4] (**MoG**): ad oggi il metodo più conosciuto ed utilizzato. Memorizza per ogni *pixel* più distribuzioni gaussiane per rappresentare differenti condizioni di illuminazioni associate al background. Per il confronto si utilizzano quattro distribuzioni gaussiane per *pixel* ed elaborazione su canale *RGB*.
2. *Conte et al.* [36] (**ABSMConte**): modifica l'ABSM di base (tra i BSM quello con una richiesta di risorse tra le più limitate), al fine di accrescere l'accuratezza (in particolar modo per scenari esterni con significative variazioni di luminosità) mantenendo una semplice e intuitiva configurazione e una ridotta complessità computazionale. Di fatto, confrontandolo con il MoG si ha una complessità computazionale di circa il *20%,* con una richiesta di memoria pari a circa il *5%*.

Dall'analisi della letteratura, le ottimizzazioni proposte si possono raggruppare in tre categorie:

1) **Ottimizzazioni che riducono i dati elaborati utilizzando una forma compressa del modello del background**: le soluzioni che appartengono a questa categoria lavorano direttamente sul modello del background al fine di ottenerne una forma ridotta. Si ha una riduzione della memoria poiché vengono memorizzate meno informazioni per ogni *pixel*, e si riduce la complessità computazionale poiché si analizzano meno informazioni o si utilizzano operazioni computazionalmente meno onerose.
2) **Ottimizzazioni che sfruttano le caratteristiche dell'hardware su cui viene elaborato l'algoritmo**: ad es. con

l'utilizzo dei FPGA (*Field Programmable Gate Array*), circuiti integrati le cui funzionalità sono programmabili via software e che permettono di elaborare enormi quantità di dati in tempo reale, in modo coerente e con latenza ridotta; oppure con l'utilizzo della GPU (*Graphics Processing Unit*) adatta per le operazioni *data-intensity,* ovvero con un'architettura *multicore*, in cui si utilizzano tecniche di *parallelization* e *partitioning* al fine di sfruttare ogni *computing core*.

3) **Ottimizzazioni che utilizzano più tecniche per creare un nuovo metodo meno oneroso**: superano i limiti dei singoli metodi combinando due o più soluzioni. Questa ottimizzazione prevede l'utilizzo di un algoritmo meno oneroso per la maggior parte dei *pixel* del *frame* corrente oppure per un numero elevato di *frame*, con l'utilizzo di un algoritmo più complesso solo per particolari aree del *frame* o per un numero limitato di *frame*.

La Tabella 1 riporta i lavori selezionati in letteratura e divisi in categorie, il cui obiettivo è quello di ottimizzare i metodi di *background subtraction* al fine di ridurne l'onere computazionale e la memoria richiesta. Nel seguito si riporta una breve descrizione di ogni metodo evidenziando le ottimizzazioni introdotte e presentando un confronto in termini di complessità spaziale e temporale rispetto ai metodi MoG e ABSMConte. Per il confronto si prendono in considerazione due tipiche applicazioni dell'analisi video: il conteggio delle persone con telecamera posizionata in corrispondenza dei varchi di ingresso e uscita (con una variazione dei *pixel* di foreground tra il *10-60%*) e sistemi di rilevamento di intrusi all'interno di aree ad accesso limitato con telecamera posizionata in modo da coprire una area più vasta (con una variazione dei *pixel* di foreground tra il *2-10%*). Nel seguiti si indicherà con *w*h* la risoluzione del *frame* acquisito.

CATEGORIE	SOTTOCATEGORIE	LAVORI	RIFERIMENTI
Ottimizzazioni che utilizzano una forma compressa del modello del background		- RaFFD: Resource-aware Fast Foreground Detection in *Embedded* Smart Cameras	Wang et al. [27]
		- Efficient Approximate Foreground Detection for Low-Resource Devices	Tessens et al. [28]
		- Block-Based Major Color Method for Foreground Object Detection on *Embedded* SoC Platforms	Tsai et al. [21]
		- Light-weight Salient Foreground Detection for *Embedded* Smart Cameras	Casares et al. [29]
Ottimizzazione che sfruttano le caratteristiche dell'hardware	*GPU Architecture*	- Fast Background Modeling Using GMM on GPU	Ye et al. [31]
	FPGA Architecture	- Background Subtraction Algorithm for Moving Object Detection in FPGA	Sanchez-Ferreira et al. [32]
	Multicore Architecture	- Real-Time Adaptive Background Modeling for Multicore *Embedded* System	Apewokin et al. [48]
Ottimizzazioni che utilizzano più tecniche		- Hierarchical Bayer-Pattern Based Background Subtraction for Low Resource Devices	Shoaib et al. [34]

Tabella 1 - Lavori con ottimizzazioni per ridurre l'onere computazionale dei metodi di *background subtraction*

Wang et al. [27] propongono un metodo, chiamato RaFF (*Resource-aware Fast Foreground Detection*), il quale effettua un'analisi a livello di regione (non a livello di *pixel*) nella fase di *background update* e individua il background dinamico lavorando solamente sulla variazione delle singole regioni rispetto al *frame* precedente. Questo permette una riduzione della complessità computazionale della soluzione. Inoltre, l'algoritmo consente di estrarre le componenti connesse senza l'utilizzo di un algoritmo di *postelaboration*. A partire dalla *foreground mask* ottenuta tramite sottrazione tra il *frame* corrente e il modello del background, il metodo effettua una ricerca dei contorni e in seguito individua le regioni (insieme di punti che formano un'area chiusa). Le regioni vengono classificate (foreground, background o background dinamico) misurando il grado di variazione (posizione e dimensione) rispetto al *frame* precedente. Nella fase di aggiornamento del modello del background, il metodo scansiona e aggiorna solo i *pixel* appartenenti alle regioni classificate come background, ciò consente di evitare un aggiornamento dell'intera immagine, quindi la complessità del metodo dipenda dalla complessità dello scenario. In generale, si ottiene una complessità tra il *3-14%* rispetto al MoG e tra il *15-62%* rispetto ad ABSMConte, con un costo più elevato per i *pixel* di foreground.

Il metodo memorizza circa un byte per *pixel* (*gray-level*), quindi la memoria richiesta per il modello del background è circa l'*1%* rispetto a quella utilizzata dal MoG e circa il *20%* rispetto ad ABSMConte.

L'accuratezza dipende dalla soglia utilizzata in fase di rilevamento dei contorni, riducendo il suo valore si ha una precisione maggiore ma anche un incremento del costo computazionale. In fase di *foreground detection* il metodo permette di considerare anche il background dinamico analizzando il grado di variazione delle regioni, ma potrebbero presentarsi problemi in caso di rapidi cambiamenti di luminosità, di conseguenza il metodo dovrebbe operare con migliori risultati in scenari interni. Negli scenari utilizzati dagli autori nelle diverse sperimentazioni (con microprocessore basato su architettura

CITRIC), con persone che camminano lungo una strada e persone in bicicletta, l'accuratezza è prossima al *95%*.

Tessen et al. [28] propongono di lavorare con *horizontal* e *vertical scan lines* (una versione ridotta del *frame* corrente) al fine di ridurre le il numero di operazioni e la memoria utilizzata. Il metodo parte dall'idea descritta in *Tessen et al.* [37]: in alcune applicazione di analisi video l'informazione a livello di *horizontal line* è sufficiente rilevare gli oggetti di interesse in movimento. Per ottenere le *scan lines*, il metodo utilizza la *Radon transform* dell'immagine lungo le due direzioni con un numero di operazioni pari a *w*h* per ogni *scan line*. Le fasi di *foreground detection* e *background update* utilizzano due modelli del background come descritto in *Gruenwedel et al.* [35] ma lavorando a livello di *scan lines* e quindi senza analizzare i *pixel* dell'intera immagine. Il metodo richiede *9*h+9*w* operazioni contro le *9*w*h* operazioni del metodo [37] applicato all'intera immagine. In generale, il costo computazionale del metodo è circa il *2%* confrontato con il MoG e compreso tra *9-15%* rispetto al metodo ABSMConte. Inoltre il metodo memorizza come modello del background solo le *horizontal* e *vertical scan lines* con un utilizzo ridotto della memoria, pari a circa lo *0.01%* confrontato con il MoG e dello *0.2%* in confronto con ABSMConte. La sperimentazione condotta dagli autori mostra come l'accuratezza della soluzione dipenda dalla distribuzione degli oggetti presenti nella scena (posizionamento della telecamera rispetto al piano): il metodo presenza prestazioni ridotte nel caso di telecamera installata in posizione zenitale poiché in tal caso non è possibile distinguere la presenza di due o più persone presenti lungo lo stesso asse. Inoltre, l'accuratezza dipende dalla dimensione degli oggetti: dovrebbero essere abbastanza grandi per l'*horizontal scan line* e piuttosto piccoli per il *vertical scan line.*

In *Tsai et al. [21]* gli autori utilizzano un metodo che modella il background a livello di blocco, questo consente di ridurre la richiesta di memoria poiché non si ha la necessità di memorizzare informazioni

per ogni *pixel* del modello. Il metodo parte dall'idea che i valori dei *pixel* nell'intorno di un dato *pixel* sono solitamente simili, per cui memorizza per ogni blocco uno o più *major color* (valori *RGB*) che rappresentano la distribuzione tonale di ogni blocco. Nelle sperimentazioni degli autori, il metodo utilizza in media tre *major color* per ogni blocco. Nella fase di *foreground detection,* l'algoritmo calcola per ogni blocco la distanza tra i valori dei *pixel* del *frame* corrente e i *major color* associati al blocco, se questa è inferiore a una data soglia il blocco è classificato come background, altrimenti come foreground, quindi la complessità in questa fase è lineare rispetto alla risoluzione del *frame*. Nella fase di *background update*, il metodo aggiorna solo i blocchi classificati come background, evitando un aggiornamento a livello dei *pixel*, con una riduzione dell'onere computazionale. Infine, per incrementare l'accuratezza della soluzione, il metodo analizza i blocchi di confine classificati come foreground effettuando un'analisi a livello dei *pixel*. In generale, si ottiene una complessità tra il *14-18%* rispetto al MoG e tra il *75-109%* rispetto a ABSMConte, con un costo più elevato per i *pixel* di foreground. Questo permette di affermare che, rispetto a ABSMConte, in scenari dove il numero di *pixel* di foreground è maggiore (ad. esempio in applicazioni di conteggio persone), la complessità è compatibile tra i due metodi, e in particolare il metodo in esame raggiunge una complessità inferiore del *25%* nel caso in cui il *50%* del *frame* corrente sia occupato da *pixel* di foreground. In media il metodo richiede *6 byte* per ogni blocco del modello del background, quindi circa l'*1%* rispetto al MoG e circa l'*11%* rispetto a ABSMConte. L'accuratezza dipende dalla dimensione del blocco, riducendo le dimensioni si ha un incremento delle *performance* con un aumento dell'onere computazionale. Altro parametro di interesse è la soglia utilizzata per la scelta dei *major color*, infatti una soglia bassa incrementa il numero dei *major color* per ogni blocco con un aumento dell'accuratezza (risposta rispetto ai cambiamenti di luminosità) ma anche della memoria richiesta e della complessità computazionale. Le

sperimentazioni effettuate dagli autori (con piattaforma *VIA VB8001*, *1.6 GHz* con unità *floating-point* e con la piattaforma *TI DaVinci* con processore *fixed-point*), con video in cui sono presenti alberi in movimento, giornate di pioggia e utilizzando il *Dataset* 3 di PETS 2001, mostrano risultati migliori rispetto ai metodi Codebook [38], LBP [9] e ALW [39], con un accuratezza intorno al *95%*.

Casares et al. [29] propongono un metodo, chiamato Adaptive Light-Weight algorithm (ALW), il quale riduce il numero di informazioni da memorizzare per ogni *pixel* appartenente al modello del background (rispetto ai metodi MoG e Codebook [7]) mantenendo soltanto un *gray-value* e un contatore che rappresenta il numero di variazioni dello stato del *pixel* negli ultimi 100 *frame*. Un valore ridotto del contatore fa riferimento ad un *pixel* stabile e affidabile, e viceversa. Questo metodo, rispetto ai due precedenti lavori [40][41], presenta una diminuzione degli accessi in memoria, del numero di istruzioni e del tempo di elaborazione (la decisione su un *pixel* di foreground viene effettuata in un modo differente e più efficiente). Nella fase di *foreground detection* viene utilizzato un *adaptive background model* e il *pixel* viene classificato analizzando anche il valore del contatore del *pixel*. Nella fase di *background update* il metodo considera il contatore associato a ciascun *pixel* al fine di incrementare la robustezza contro variazioni di luminosità e background dinamico. La differenza rispetto ai lavori precedenti è presente in fase di classificazione: quando un *pixel* è affidabile il metodo non calcola e verifica il contatore, con una diminuzione, in media, del tempo di elaborazione di circa *4.5ms*. Inoltre, l'algoritmo lavora con un unico canale e il numero di accessi in memoria e delle istruzioni dipendono dallo scenario (un numero inferiore al crescere dei *pixel* di background stabile). In generale, il costo computazionale è tra *4-11%* rispetto al MoG e il *29-45%* rispetto a ABSMConte, con un costo più elevato per i *pixel* di foreground. La memoria richiesta è di circa *6.25byte* per *pixel*, quindi il *6%* rispetto a MoG e del *125%* confrontata con il

metodo ABSMConte. Le sperimentazioni realizzate su una *embedded smart camera* non specificata, evidenziano un buon compromesso tra complessità spaziale, temporale e accuratezza della soluzione (con particolare riferimento a variazioni di luminosità e background dinamico), ottenendo prestazioni migliori rispetto ai metodi MoG [4][23] e Codebook [7].

Ye et al. [31] descrive un'implementazione parallela su GPU del metodo MoG, al fine di sfruttare le caratteristiche dell'architettura. Difatti, la fase di *background update* si adatta pienamente ad un'implementazione su GPU utilizzando CUDA (Compute Unified Device Architecture): le principali operazioni in *floating-point* del metodo sono presenti in questa fase e l'aggiornamento è un processo *pixel-based* che avviene senza alcuna dipendenza tra i *pixel* (*no data correlation*). Parti del metodo vengono eseguite come *kernel* e CUDA permette di elaborarli più volte su più *thread* simultaneamente. In particolare, gli autori descrivono due tipi di implementazioni: una di base in cui ad ogni *GPU thread* è assegnato l'elaborazione di un *pixel*, e una implementazione *GPU asynchronous* con diverse ottimizzazioni (*memory coalescing, pinned memory, asynchronous execution*). Le ottimizzazione effettuate dagli autori sono in accordo alla guida CUDA delle *best practices [57]*, come quelle che riguardano le strutture dati che rappresentano le distribuzioni gaussiane per migliorare le prestazioni tramite la *coalesced access* (in cui *thread* consecutivi accedono a *global memory locations* consecutive). L'architettura consente al codice di essere elaborato contemporaneamente alle operazioni di copia della memoria (*exeStream* and *copyStream* sono due *CUDA stream*). L'implementazione GPU *asynchronous* reduce gli effetti del ritardo del trasferimento dei dati in confronto alla soluzione CPU e *basic-GPU*, infatti, considerando la velocità di elaborazione si ha *CPU process* > *basic GPU process* > *asynchronous GPU process*. Utilizzando un video in formato HD, l'implementazione *basic-GPU*

incrementa la velocità di *7x* in confronto con la soluzione CPU, mentre l'implementazione *asynchronous-GPU* fornisce un incremento di *3x*. Per quanto riguarda l'accuratezza, gli autori non evidenziano significative differenze tra la foreground ottenuta con esecuzione su CPU e quella ottenuta con le due implementazioni su GPU.

Sanchez-Ferreira et al. [32] implementano un *Gaussian background modeling* su piattaforma FPGA al fine di ridurre il tempo di esecuzione tramite un elaborazione in parallelo della fase di *background update* senza ridurre l'accuratezza in confronto all'implementazione su CPU. La soluzione proposta dagli autori elabora un *pixel* per ogni ciclo di clock dell'FPGA e incrementa la velocità di elaborazione della soluzione su CPU di un fattore pari a *32*. Il metodo memorizza il modello del background (*gray-level*) in una memoria SRAM esterna, applica un *low-pass filter*, esegue la differenza tra il *frame* corrente e il modello e applica una sogliatura per ottenere la *foreground mask*. In seguito, tramite un *morphological filter* riduce il rumore e infine ricerca i contorni degli oggetti derivati. In definitiva, il sistema utilizza i seguenti moduli: un modulo per la conversione da *3-color channel* a *8-bit channel (gray-scale)* che lavora in un ciclo di clock, un modulo *convolution filter (low-pass filter)* che opera in un ciclo di clock, un modulo per la sottrazione del modello del background che per ogni *pixel* impiega un ciclo di clock, un modulo per la segmentazione che lavora in un ciclo di clock e un modulo per il *morphological filter* che opera in un ciclo di clock. Il sistema è implementato su una piattaforma *Altera* basata su *Cyclone II EP2C35 device*. Il tool *EDA Quartus II* è utilizzato dagli autori per progettare il sistema. L'implementazione su FPGA minimizza la *latency memory* ed elabora un *pixel* per ogni ciclo di clock dell'FPGA, raggiungendo i *60 FPS* (un *pixel* ogni *43.43 ns*), circa *32* volte più veloce della soluzione su CPU.

Apewokin et al. [48] mostrano come un'implementazione *multicore* (diverse configurazioni di elaborazione e di parallelizzazione) possa

aumentare, con un incremento del *25%*, le prestazioni di un *multimodal method,* in particolare del metodo chiamato MMean (Multimodal Mean) proposto in [48]. Il metodo MMean combina le caratteristiche delle *mixture of Gaussians* con una elaborazione dei *pixel* basata solo su somme e medie (complessità inferiore). Più in dettaglio, MMean presenta un'accuratezza paragonabile a quella del metodo MoG con un riduzione pari a *6x* del tempo di elaborazione e di un *18%* della memoria richiesta. Ogni *pixel* del modello del background è un insieme di *K* celle, dove ognuna è rappresentata da tre sommatorie e tre medie per ogni canale (*RGB*) e un contatore di quante volte un *pixel* è stato associato a una delle corrispondenti celle negli ultimi *t frame*. Nella fase di *foreground detection*, un *pixel* è confrontato con le *K* celle, quando il metodo rileva una corrispondenza con almeno una delle celle (soglia su tutti i canali *RGB*), il *pixel* viene classificato come background, altrimenti come foreground e una nuova cella viene creata (sostituendo una esistente nel caso si sia raggiunto il limite di *K* celle per il *pixel* in esame). Per i *pixel* classificati come background effettua il *background update* aggiornando la cella con cui si è rilevata una corrispondenza. Ogni *d frame*, effettua una decimazione di tutte le celle per adattare il modello ai cambiamenti di luminosità e limitare i contatori evitando l'*overflowing* della memoria in particolar modo per i sistemi con risorse limitate. Il metodo proposto dagli autori utilizza un *tiled partitioning* nel quale immagini consecutive sono elaborate in *stacks* di blocchi di immagine più piccoli assegnati ad ogni *core*. Questo approccio minimizza il trasferimento del modello del background tra la memoria centrale e ogni *processing core*. La sperimentazione (su *eBox-2300 Thin Client VESA PC* con *Windows Embedded CE 6.0* e su *Sony Playstation 3*) mostra un incremento del *25%* delle *performance* della soluzione *multicore* rispetto a quella su CPU.

Shoaib et al. [34] propongono un *hierarchical method* nel quale si ottiene una riduzione della complessità computazionale nelle fasi di

foreground detection e *background update* lavorando, per la maggior parte del *frame*, con un primo metodo *block-based* (con tempi di elaborazione), e utilizzando un metodo *pixel-based* (più accurato ma con maggiori tempi di elaborazione) solo per i blocchi classificati come foreground. Il primo algoritmo lavora in *single channel Bayer-pattern domain* in modo da sfruttare l'elevata correlazione esistente nell'intorno di ogni *pixel* (visto che le telecamere utilizzano un singolo sensore con un *bayer-pattern* per ricostruire l'immagine). Il primo metodo presenta una limitata memoria e complessità computazionale poiché memorizza ed elabora le informazioni a livello di blocco e non di *pixel*. Ogni blocco è rappresentato da due distribuzioni gaussiane, la *frchet distance [49]* viene utilizzata per verificare la corrispondenza tra i blocchi del *frame* corrente e quelli del modello. Una soglia determina la classificazione del blocco. Solo i blocchi classificati come *background* sono aggiornati utilizzando due coefficienti di aggiornamento diversi. Per incrementare l'accuratezza della soluzione, i blocchi classificati come foreground sono analizzati dal secondo metodo (analisi *pixel-level*). Quest'ultimo memorizza per ogni *pixel* del modello tre distribuzioni gaussiane e lavora su *interpolated RGB domain*. In particolare, gli autori seguono l'approccio presentato in *Suhr et al.* [50] in cui la modellazione del *background* avviene in *Bayer-pattern domain* (media e varianza sono valori scalari) mentre la fase di *foreground detection* è elaborata in *interpolated RGB domain*: le altre due componenti del colore sono interpolate guardando l'intorno del *pixel* corrente (utilizzando la tecnica *bilinear demosaicing* [51]). La fase di *background update* è simile a quella del primo metodo ma avviene a livello di *pixel* e non di blocco. Infine, per aumentare l'accuratezza della *foreground mask*, gli autori utilizzando il *color model* proposto in *Kim et al.* [7] e *Guo et al.* [52] per classificare il *pixel* di foreground come *highlight, shadow* o *foreground* (analizzando la *color distortion* tra i *pixel* del *frame* corrente e la *background distribution*). Per quanto riguarda la complessità computazionale, il metodo utilizza un algoritmo di

complessità non elevata per la maggior parte del *frame* (in particolar modo per gli scenari in cui è presente principalmente il *background*), questo porta ad un costo computazionale tra il *2-22%* in confronto al MoG e tra il *12-90%* in confronto a ABSMConte, con un costo più elevato per i *pixel* di foreground. Nelle sperimentazioni degli autori, il metodo raggiunge, in media, i *218 FPS* con blocchi di dimensione pari a *9 pixel*, mentre utilizzando unicamente il metodo *block-based* si raggiungono i *410 FPS*. La memoria richiesta dal modello *block-based* è *w*h*3*Kb/N* dove *N* è la dimensione del blocco (*9* è quella utilizzata dagli autori), *Kb* corrisponde al numero di distribuzioni per ogni blocco (pari a *2*), mentre *w*h*4*Kp* è la memoria richiesta dal modello *pixel-based* con *Kp=3* distribuzioni per *pixel*. Quindi, la memoria richiesta è circa l'*11%* in confronto a quella del MoG e *240%* in confronto al metodo ABSMConte. In particolare, anche se l'algoritmo richiede che i modelli di entrambi gli algoritmi siano in memoria, gli accessi avvengono in media meno frequentemente poiché utilizza per le zone di *background* solo il primo metodo che richiede l'accesso a due distribuzioni per ogni blocco. In termini di accuratezza, le sperimentazioni degli autori (su *3.0 GHz Intel Core 2 Quad CPU*) forniscono una percentuale di *true positive* pari a *97%* e una percentuale di *false positive* pari al *6%*.

Capitolo 2. Ingegnerizzazione di un Algoritmo di Background Subtraction

Le ottimizzazioni analizzate in precedenza, hanno l'obiettivo di ridurre la complessità spaziale e/o temporale di un particolare metodo appartenente alla tipologia dei *background subtraction*. Dall'analisi della letteratura, le principali ottimizzazioni introdotte si riferiscono ad un particolare metodo e in alcuni casi ad una specifica architettura, questo ne limita la generalità e quindi la possibilità di replicarle su altri metodi e architetture.

In questo capitolo si descrive il **processo di ingegnerizzazione di un algoritmo di background subtraction** per le fasi di *foreground detection* e *background update*, utilizzando ottimizzazioni che consentono l'elaborazione *real-time* su diverse architetture sia *embedded* che tradizionali, e applicabili a svariati algoritmi di analisi video intelligente. Tali ottimizzazioni derivano da un'esperienza maturata utilizzando diverse *smart camera* presenti sul mercato. Successivamente si riportano i dettagli di un'approfondita sperimentazione per caratterizzare le prestazioni del metodo su diverse famiglie di processori disponibili su *smart camera* commerciali.

2.1 Ottimizzazioni di Alto e Basso Livello

Le ottimizzazioni utilizzate per completare il processo di ingegnerizzazione si possono raggruppare in due tipologie:

- **Ottimizzazioni di alto livello**: strutturare il codice al fine di ridurre le *miss cache penalty*, utilizzare le opzioni del compilatore, convertire le operazioni di moltiplicazione e divisione con operatori *left shift* e *right shift*, convertire le operazioni *floating-point* in *fixed-point,* ridurre le allocazioni dinamiche della memoria, utilizzare le *inline function*;

- **Ottimizzazioni di basso livello**: sfruttare le istruzioni SIMD (Single Instruction Multiple Data)

Al fine di ridurre il *gap* tra la velocità della CPU e le *performance* della memoria, nei calcolatori viene implementata una memoria gerarchica. In questo modo, piccole, costose e performanti memorie sono integrate all'interno del *processor chip* al fine di fornire dati con bassa *latency* ed elevata *bandwidth* (registri della CPU e memorie *cache*). La memoria *cache* contiene copie dei blocchi contenuti nella memoria principale per incrementare la velocità degli accessi a quei blocchi che sono richiesti più frequentemente. La *cache* contiene *byte* che sono localizzati consecutivamente in memoria centrale, in base al principio della località spaziale: quando si accede ad un elemento della memoria si farà riferimento entro breve tempo, con probabilità prossima a 1, ad altri elementi che hanno indirizzo vicino a quello dell'elemento corrente (ad es. sequenzialità delle istruzioni e accesso a dati organizzati in vettori o matrici). Se i dati richiesti dal processore si trovano nella *cache* si parla di *cache hit*, altrimenti di *cache miss* (in questo caso bisogna recuperare i dati dalla RAM). Per ridurre il numero di *cache miss* dell'algoritmo si modifica l'ordine di scansione delle immagini, procedendo per colonna invece che per riga, questo perché all'interno della memoria le immagini sono memorizzate per colonna in un array a 1 dimensione. Un altro tipo di ottimizzazione è quella di unificare due cicli che presentano lo stesso spazio iterativo in un singolo ciclo, questo riduce l'*overhead* e aumenta il parallelismo. Altra ottimizzazione riguarda il *merging* di più array in un array multidimensionale oppure l'utilizzo delle strutture, al fine di rendere le variabili continue in memoria, come riportato in Figura 9.

```
1: double a[1024]                                  ▹ Original data structure
2: double b[1024]
3:
4: double ab[1024][2]      ▹ Array merging using multidimensional array
5:
6: struct{                                         ▹ Array merging using structures
7:     double a
8:     double b
9: }ab[1024]
```

Figura 9 – Esempio di *array merging* per ridurre i casi di *cache miss*

Un'altra tipologia di ottimizzazioni di alto livello sono quelle che si riferiscono al compilatore. Lavorando sulle opzioni di quest'ultimo è possibile migliorare le *performance* e ridurre la dimensione del codice al costo di un aumento del tempo di compilazione. Analizzando la Tabella 2, con l'utilizzo del *flag –O3* si ottiene una compilazione più rapida con un aumento dei tempi di elaborazione; ad esempio se la dimensione dell'immagine supera la dimensione disponibile dell'*instruction cache*, si ottiene una riduzione delle prestazioni. Quindi, al fine di bilanciare tempi di elaborazione e tempi di compilazione, si consiglia l'utilizzo dell'opzione *–O2*. Si sottolinea che alcune dei suddetti *flags* rendono impossibile il *debbugging*, quindi durante la fase di *debug* dell'applicazione si raccomanda di disabilitarli. Per maggiori dettagli si rimanda alla guida GCC (*GNU Compiler Collection*) [56].

Flag	Ottimizzazioni	Tempo di elaborazione	Dimesione del codice	Memoria	Tempo compilazione
-O0	per il tempo di compilazione (default)	+	+	-	-
-O1	per il codice e per il tempo di elaborazione	-	-	+	+
-O2	per il codice e per il tempo di elaborazione	--		+	++
-O3	per il codice e per il tempo di elaborazione	---		+	+++
-Os	per il codice		--		++

Tabella 2 - Opzioni del compilatore, con il sinbolo "+" si indica un incremento, con il simbolo "-" un decremento

Nel corso del processo di ingegnerizzazione si utilizzano gli operatori di *bitwise* che agiscono sui singoli bit. In particolare, questi operatori ("<<" *shift* a sinistra, ">>" *shift* a destra) eseguono un appropriato spostamento dall'operatore indicato a destra a quello indicato a sinistra. I *bit* liberati vengono riempiti con zero. Ad esempio, se *c=00000010* (binario) o 2 (decimale), allora *c>>2* effettua uno *shift* dei bit in c di due posti verso destra => *c=00000000* (binario) o 0 (decimale), inoltre, *c<<2* effettua uno *shift* dei bit in c di due posti verso sinistra => *c=00001000* (binario) o *8* (decimale). Quindi, uno *shift* a sinistra equivale ad una moltiplicazione per *2* e uno *shift* a destra ad una divisione per *2*. L'operazione di *shift* è molto più veloce della moltiplicazione (*) o divisione (/), quindi al fine di ottenere moltiplicazioni e divisioni per 2 veloci si utilizzano i predetti operatori.

In molti casi, i sistemi *embedded* presenti sul mercato non sono dotati di FPU (*Floating-point Unit*), di conseguenza queste operazioni devono essere simulate con una perdita in efficienza. Una possibile ottimizzazione, quindi, riguarda la conversione delle operazioni

floating-point in *fixed-point*. Per di più, anche se il processore è dotato della *floating-point unit*, l'utilizzo delle operazioni *fixed-point* su processori SIMD porta ad una elaborazione proporzionalmente più rapida. Questa ottimizzazione consente anche una riduzione delle richiesta di memoria, proporzionale al numero di variabili *floating-point* presenti nel metodo di base. L'accuratezza della soluzione può essere conosciuta comparando le *performance* al variare del livello di precisione. In genere, la perdita di accuratezza è trascurabile.

Nelle applicazioni real-time su sistemi *embedded*, l'utilizzo delle API (*Application Programming Interface*) *malloc* e *free* sono considerate una violazione delle *best practices*. Difatti, queste funzioni hanno prestazioni impreviste (il tempo necessario per allocare memoria è estremamente variabile), e potrebbe introdurre nel codice dell'applicazione perdite di memoria o frammentazione. Questi comportamenti indesiderati tendono a causare una progressiva riduzione delle prestazioni ed eventuali errori. Detto ciò, il numero di allocazione dinamiche si riduce al minimo.

Un'altra raccomandazione da rispettare nel corso dell'ottimizzazione di un algoritmo, riguarda l'utilizzo delle funzioni *inline* nei cicli. Questo elimina l'*overhead* associato alle chiamata della funzione, poiché suggerisce al compilatore di sostituire il corpo della funzione eseguendo l'espansione *inline*, senza dover passare ad un'altra posizione per eseguire la funzione, per poi ritornare al codice della funzione chiamata. Nel riportare questa ottimizzazione bisogna considerare che l'utilizzo delle funzioni *inline* aumenta la dimensione del codice.

Come tipologia di ottimizzazione di basso livello, si sceglie l'utilizzo delle istruzioni SIMD fornite dalla CPU. Le istruzioni SIMD permettono di processare almeno quattro *pixel* in un ciclo di clock, il che riduce significativamente il tempo di elaborazione delle immagini. Ad esempio, per modificare la luminosità dei *pixel* di un'immagine un

microprocessore dovrebbe caricare i valori dei *pixel* nei registri, effettuare le dovute modifiche e salvare il risultato in memoria. Un processore SIMD effettua le operazioni a blocchi (caricando un certo numero di *pixel* in un'unica operazione) con una riduzione significativa del tempi. Come mostrato nel seguito, utilizzando le istruzioni SIMD, il numero di *pixel* elaborati al secondo aumenta di un ordine di grandezza, mentre le operazioni *integer-point* sono processate fino a due ordini di grandezza più rapidamente rispetto alle operazioni in *floating-point*.

2.2 Ingegnerizzazione di un Adaptive Background Subtraction Method e Valutazione delle Prestazioni

Le ottimizzazioni viste in precedenza rappresentano una base di partenza per l'avvio del processo di ingegnerizzazione di un algoritmo di analisi video. Considerando che, in generale, gli algoritmi di *moving object detection* (con particolare riferimento a quelli di *background subtraction*) presentano una complessità tra le più elevate nel contesto dell'analisi video (come confermato dal lavoro di Chen et al. [1]) e rappresentano il primo *step* in diverse applicazione, si è scelto di focalizzata l'attenzione su questa tipologia di algoritmi.

Il paragrafo descrive l'attività progettuale e sperimentale finalizzata all'ingegnerizzazione di un algoritmo di *background subtraction* ampiamente usato in letteratura per consentire la sua esecuzione su piattaforme con ridotte capacità di calcolo e di memoria. Le ottimizzazioni introdotte sono oggetto di un'ampia attività sperimentale per caratterizzare le prestazioni del metodo su diverse famiglie di processori disponibili su *smart camera* commerciali.

Come metodo di base, si è adottato l'ABSM proposto in [36] per le fasi di *foreground detection* e *background update.* La scelta è ricaduta su questo metodo poiché presenta una configurazione semplice ed intuitiva, una buona accuratezza e pur essendo una soluzione con una

complessità spaziale e temporale non particolarmente elevata (in confronto con i metodi più utilizzati in letteratura), le sperimentazioni su *smart camera* hanno evidenziato prestazioni non idonee per far parte di un modulo per applicazioni *real-time*.

L'algoritmo implementato è composto da tre *step* principali (illustrati in Figura 10):

- ✓ *Absolute distance evaluation*: calcola la *distance map* dove ogni *pixel* rappresenta la L_1 *distance* tra il *pixel* del *frame* corrente e quello del modello del *background*,
  $$F_n(x,y) = |I_n(x,y) - B_n(x,y)|$$
  dove F_n è la *distance map* calcolata, I_n è il *frame* corrente e B_n è il modello del *background* (*current reference image*);
- ✓ *Distance image thresholding*: alla *distance map* ottenuta in precedenza applica una soglia per ottenere la *foreground mask*
  $$M_n(x,y) = \begin{cases} 1, & \text{if } F_n(x,y) \geq Th \\ 0, & \text{if } F_n(x,y) < Th \end{cases}$$
  dove M_n è la *foreground mask* e T_h è la soglia predefinita;
- ✓ *Reference image update (IIR filter)*: aggiorna il modello del *background* utilizzando la seguente equazione
  $$B_{n+1}(x,y) = (1-\alpha) * B_n(x,y) + \alpha * I_n(x,y)$$
  dove B_n è il modello del *background*, I_n il *frame* corrente e α il coefficiente che determina la velocità di aggiornamento.
  Il più delle volte, il *frame* corrente e il modello del *background* sono immagini *RGB* o *YUV*, di conseguenza le precedenti operazioni si applicano separatamente sui tre canali. Inoltre, poiché nella seconda fase si ottiene una *foreground mask* per ogni canale, si impiega un operatore *OR* per ottenere un'unica maschera (*binary OR*).

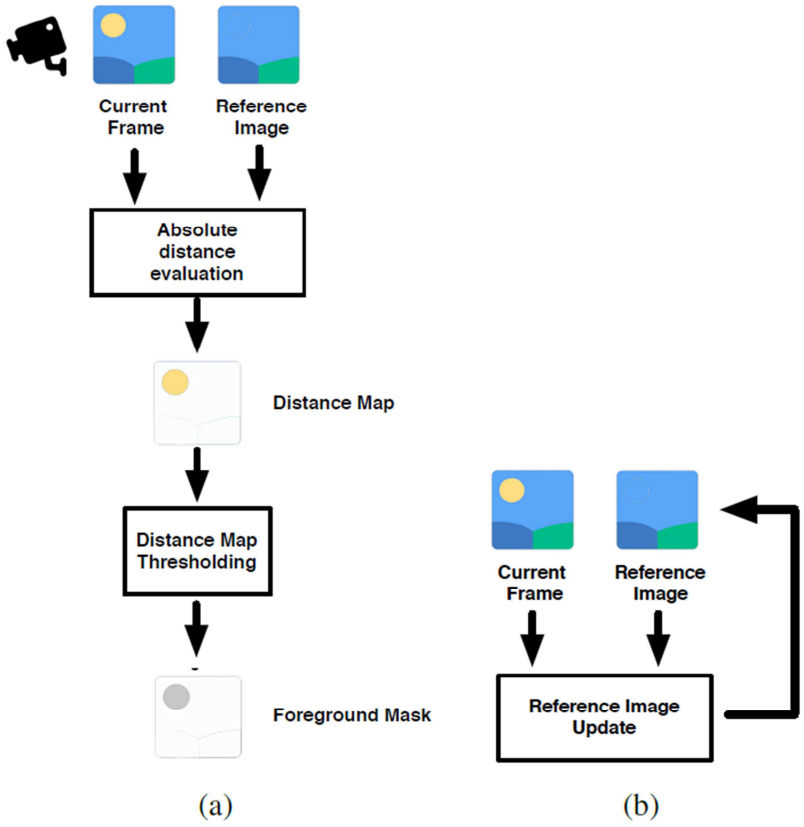


Figura 10 - Algoritmo di *background subtraction* di riferimento. In (a) la fase di *foreground detection*, in (b) la fase di *background update*

Ad oggi, diversi produttori di telecamere commerciali (ad es. Axis e Samsung) rendono disponibili piattaforme programmabili che permettono di sviluppare software per estendere le funzionalità del *device*. Generalmente, queste piattaforme si basano su architetture SoC (*System on a Chip*) composte da CPU *general purpose,* DSP (*Digital Signal Processor*), porta Ethernet e USB, *flash memory*, SDRAM (*Synchronous Dynamic Random Access Memory*), *I/O processors*. Dall'analisi delle caratteristiche delle piattaforme SoC, si passa ad identificare un insieme di ottimizzazioni *cross-platform*. In particolare, le CPU forniscono istruzioni SIMD, questa è una

caratteristiche rilevante per gli algoritmi di analisi video poiché consente di elaborare più di un *pixel per cycle*. Inoltre, le CPU in diversi casi non sono dotate di *floating-point unit*, poiché i costruttori preferiscono riservare le risorse hardware limitate ad altre caratteristiche (come la possibilità di applicare la crittografia al flusso, in accordo ai vincoli normativi sulla privacy, o per fornire codifiche diverse). Inoltre la presenza in molte telecamere del *kernel Linux* rappresenta un vantaggio in termini di portabilità del software su diverse telecamere programmabili. Per la fase di sperimentazione si selezionano due architetture CPU presenti su telecamere della Axis Communications: CRIS e MIPS (per dettagli sulle telecamere utilizzate si rimanda al Capitolo 5).

A partire dal metodo appena descritto, si applicano le ottimizzazioni di alto e basso livello descritte in precedenza, tra cui:

1. **Utilizzo delle istruzioni SIMD** nelle fasi di *absolute distance evaluation, distance image thresholding* e *reference image update* per poter processare almeno quattro *pixel* in un ciclo di *clock*, il che permette di ridurre significativamente il tempo di elaborazione delle immagini.
   Al fine di ottenere una soluzione portabile, si utilizza la libreria RAPP (Raster Processing Primitives) [55], libreria ANSI C open-source e portabile su un ampio *range* di piattaforme che consente allo sviluppatore l'utilizzo di funzioni ottimizzate e affidabili per la manipolazione a basso livello delle immagini (per dettagli si rimanda al Capitolo 4).
   La libreria è progettata per l'elaborazione delle consuete operazioni di analisi video, ad esempio, *spatial filtering*, *segmentation* e *labeling*, sfruttando le istruzioni SIMD. Con le funzioni fornite dalla libreria si sostituiscono tutte le *pixelwise operations* implementate tramite cicli nidificati. Analizzando i valori riportati in Tabella 3, si apprezzano i miglioramenti ottenuti sulle principali funzioni utilizzate nell'algoritmo di

riferimento. In particolare, si noti come nella maggior parte dei casi, si ha un miglioramento pari a un ordine di grandezza nel numero di *pixel* elaborati al secondo.

2. **Conversione delle operazioni floating-point** in *fixed-point* nella fase di *reference image update* al fine di ridurre i tempi di elaborazione. Il vantaggio di tale processo si presenta in Tabella 4 che riporta il numero di operazioni al secondo (*integer* e *floating-point*) su architetture CRIS e MIPS. I valori ottenuti evidenziano un incremento prestazionale delle operazioni su interi fino a due ordini di grandezza rispetto alle stesse operazioni in *floating-point.* Tale ottimizzazione è richiesta per l'utilizzo delle istruzioni SIMD, difatti le operazioni *floating-point* non ne permettono l'utilizzo. In accordo al metodo di riferimento, poiché le operazioni *floating-point* sono presenti solo durante la fase di *reference image update* (con due moltiplicazioni *floating-point* dovute al coefficiente α), si passa a convertire il coefficiente dal *range [0,1]* (*floating-point*) al *range* intero su *8 bit [0, 255]*. In accordo alla Tabella 4 e considerando una risoluzione dell'immagine pari a *320*240* è possibile completare l'aggiornamento del *background* su architettura CRIS in circa *230 millisecondi* con operazioni *floating-point* e circa *3 millisecondi* con interi, quindi sull'intera immagine si ottiene un'elaborazione in circa *0.2 secondi* (*5 FPS*) nel primo caso e di circa *35 secondi* (*0.02 FPS*) nel secondo. Mentre l'architettura MIPS richiede circa *1.85 secondi* (*0.55 FPS*) con operazioni *floating-point* e circa *0.025 second*i (*40 FPS*) con interi.

Operazione	CRIS (*pixel*/sec)	MIPS (*pixel*/sec)	CRIS SIMD (*pixel*/sec)	MIPS SIMD (*pixel*/sec)
Subtraction and Absolute	3.4M	6.2M	20.97M	96.99M
Threshold	6.9M	20.4M	28.18M	247.07M
Binary OR	6.6M	23.3M	689.44M	2.38G
IIR filter	6.6M	23.3M	17.04M	87.16M

Tabella 3 - Numero di *pixel* elaborati al secondo per ogni operazione dell'algoritmo di background subtraction su architetture CRIS e MIPS

Operazione	CRIS (op./sec)				MIPS (op./sec)			
	SUM	**SUB**	**MUL**	**DIV**	**SUM**	**SUB**	**MUL**	**DIV**
Integer	4.1K	5.1 K	4.2 K	0.4 K	62.0K	73.0K	18.0 K	8.0 K
Floating-point	0.05K	0.04 K	0.08 K	0.06 K	0.6K	0.5 K	0.4 K	0.4 K

Tabella 4 - Numero di operazioni (*integer* e *floating-point*) al secondo su architetture CRIS e MIPS

Dal punto di vista implementativo, l'algoritmo prende in input un flusso video in formato *I420* nello spazio di colore YUV con profondità di *8 bit* (per canale). La scelta del formato video planare è motivata dalla possibilità di poter utilizzare le librerie RAPP (come approfondito nel Capitolo 5).
Per la fase di *absolute distance evaluation* si effettua il calcolo delle distanze ottenendo tre immagini separate una per ogni canale, questo poiché i tre canali sono planari. Per ogni immagine si implementa la fase di *distance image thresholding* ed infine si ottiene la *foreground mask* attraverso un OR logico sui risultati delle tre immagini.

La sperimentazione dell'algoritmo si effettua su due architetture SoC di Axis equipaggiate rispettivamente con CPU CRIS e MIPS (per dettagli si rimanda al Capitolo 5). Le *smart camera* utilizzate presentano un sistema operativo che riserva una percentuale della CPU al fine di garantire la qualità e l'FPS richiesti dall'utente per lo *streaming* del video sulla rete (funzionalità di base della telecamera). Di conseguenza, durante i test si ha un FPS massimo pari a *25* (un

valore conforme per molte applicazioni di analisi video) e un utilizzo massimo della CPU pari a circa il *40%* per entrambe le architetture.

Durante la sperimentazione, si considerano le seguenti risoluzioni associate alle immagini acquisite: *QVGA/4 (160*120), QVGA (320*240)* e *VGA (640*480)*; e i seguenti indici prestazionali: FPS, utilizzo della CPU e tempo di elaborazione. Tali indici si calcolano per ogni fase dell'algoritmo al fine di caratterizzare in maniera puntuale le prestazioni dell'algoritmo.

Si noti come la memoria richiesta dall'algoritmo è costante durante l'elaborazione e si compone di tre immagini: la *reference image* (modello del *background*), il *frame* corrente e la *foreground mask*.

Per ogni piattaforma *hardware* e per ogni risoluzione si considerano tre *test* indipendenti, ognuno con un tempo pari a cinque minuti, al termine dei quali si calcola il valor medio associato ad ognuno degli indici prestazionali selezionati. In Figura 11 e Figura 12 si mostrano i valori medi di FPS e la percentuale di utilizzo della CPU al variare della risoluzione e dell'architettura. Nel *worst case* (risoluzione VGA) l'algoritmo richiede il massimo della risorsa CPU messa a disposizione dalla *smart camera* (circa il *40%* per entrambe le architetture), e quanto riguarda l'FPS l'architettura MIPS raggiunge un valore (circa *10*) pari al doppio rispetto al CRIS. Con risoluzione QVGA/4 e QVGA, le ottimizzazioni permettono di elaborare l'algoritmo al massimo del *frame rate* su MIPS e quasi al valore massimo su CRIS. In Figura 13 e Figura 14 si riportano le risorse utilizzate (millisecondi) in ogni fase dell'algoritmo sulle CPU selezionate. In generale, le *performance* raggiunte dal MIPS sono migliori rispetto a quelle del CRIS, il tempo di esecuzione medio richiesto dal MIPS è circa un ordine di grandezza inferiore rispetto a quello del CRIS. Il divario tra le due architetture aumenta all'aumentare della risoluzione, difatti con risoluzione massima (VGA) il tempo di elaborazione nel caso del MIPS è in media pari a

35ms mentre su CRIS è di circa *150ms*. L'accuratezza della *foreground mask* è compatibile con quella fornita dal metodo di base senza l'introduzione delle predette ottimizzazione. Quanto visto è riportato in lavoro pubblicato nel 2014:

Carletti, V., Del Pizzo, L., Percannella, G., & Vento, M. (2014, November). Foreground detection optimization for SoCs embedded on smart cameras. In Proceedings of the International Conference on Distributed Smart Cameras (p. 31). ACM.

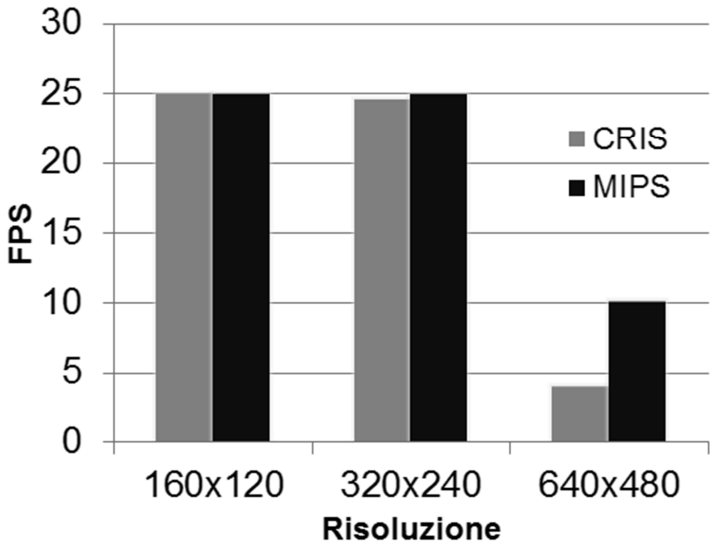


Figura 11 - FPS medio al variare della risoluzione e dell'architettura

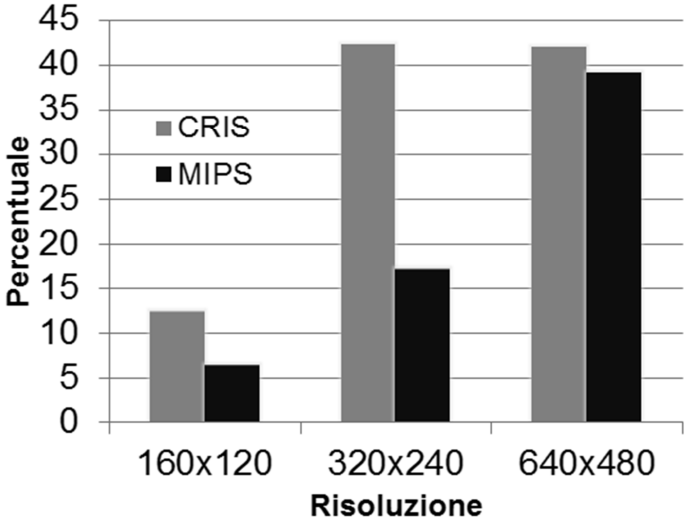


Figura 12 - Percentuale media di utilizzo della CPU al variare della risoluzione e dell'architettura

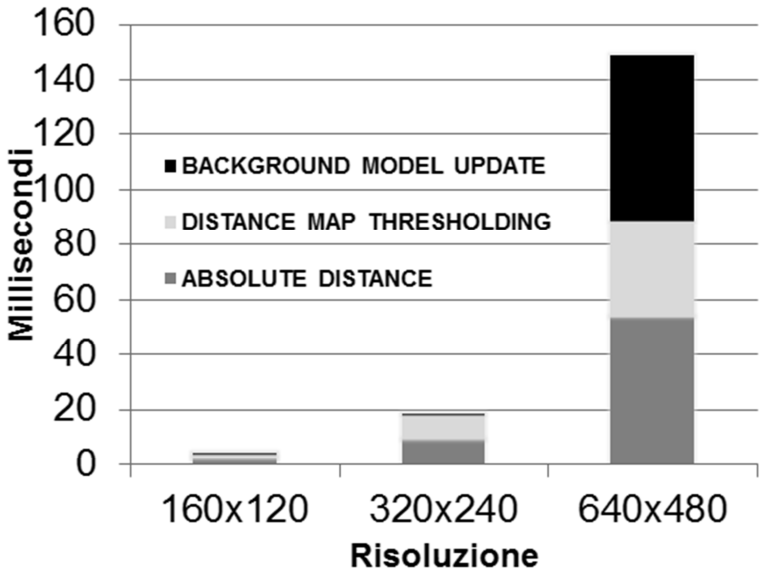


Figura 13 - Tempo medio di elaborazione (millisecondi) per ogni fase dell'algoritmo al variare della risoluzione con architettura CRIS

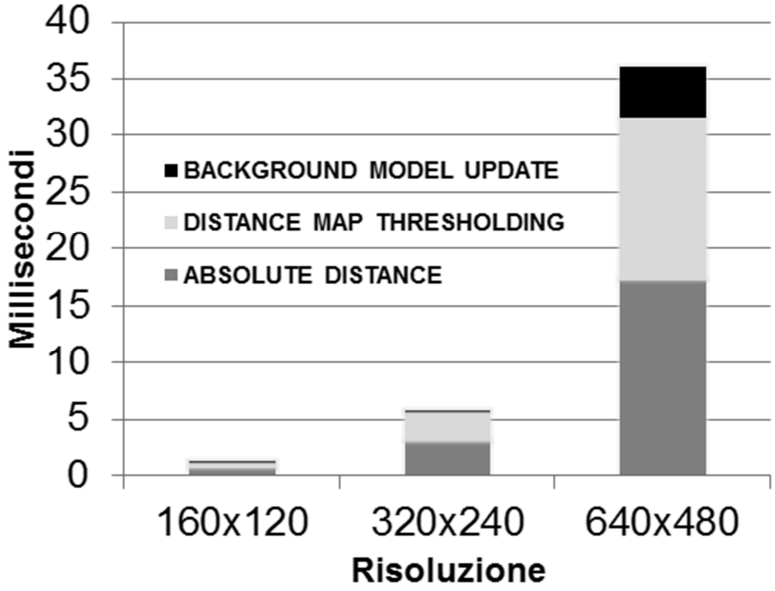


Figura 14 - Tempo medio di elaborazione (millisecondi) per ogni fase dell'algoritmo al variare della risoluzione con architettura MIPS

.

Capitolo 3. Conteggio Automatico delle Persone

Esistono numerose tecnologie di mercato che consentono il conteggio delle persone in ingresso e uscita, a partire dalle barriere all'infrarosso fino ad arrivare a telecamere termiche capaci di rilevare oggetti caldi in movimento, sensori biometrici, ecc. Negli ultimi anni si assiste ad un incremento dell'interesse nei confronti delle applicazione di conteggio delle persone con algoritmi di analisi video intelligente e telecamere installate in posizione zenitale. Queste soluzioni presentano una precisione maggiore rispetto ai sensori a infrarossi (Figura 15), ad esempio, in scenari in cui due o più persone attraversano insieme il varco, il sistema a infrarossi riconosce un unico attraversamento, mentre in diverse condizioni il sistema di analisi video riporta entrambi i passaggi (Figura 16).

Figura 15 - Sensori a infrarossi

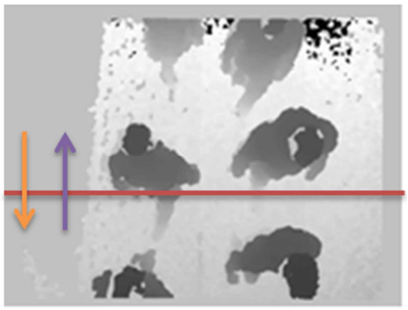


Figura 16 - Sensore linea virtuale con telecama di profondità

I sistemi di analisi video intelligente per il conteggio automatico delle persone che attraversano una linea virtuale, forniscono importanti informazioni in diversi scenari reali. In generale, tali sistemi si installano all'ingresso e all'uscita di aree chiuse e permettono di ottenere informazioni riguardanti l'affluenza. Alcuni esempi di installazione sono: centri commerciali, negozi, musei, fiere, metropolitane, treni e autobus. In tutti questi luoghi, i sistemi di conteggio sono estremamente importanti per ragioni di *business intelligence* e *security*. Difatti, tali sistemi permettono di valutare, per ragioni di sicurezza, il raggiungimento del massimo numero di individui presenti in una struttura (indice di affollamento) e di fornire statistiche sull'affluenza utili per la gestione delle vendite. L'ambito *retail* rappresenta uno dei tipici scenari dove tali sistemi sono ampiamente utilizzati. Le statistiche sul numero di persone presenti in un negozio o in un centro commerciale sono rilevanti per diversi scopi: permette di allocare al meglio il personale, di correlare l'affluenza dei clienti con specifiche campagne di *marketing* o con le *performance* di vendita di negozi simili nella stessa zona, e così via.

Nel seguito si descrive la realizzazione e la caratterizzazione delle prestazioni, sia dal punto di vista dell'accuratezza che dei requisiti computazionali, di un **metodo per il conteggio delle persone che attraversano una linea virtuale mediante la elaborazione del flusso video in tempo reale da una telecamera fissa installata in posizione zenitale** (come illustrato in Figura 17). L'utilizzo dei sensori in posizione zenitale consente di ridurre le occlusioni, quando un soggetto è parzialmente o completamente oscurato da altre persone o elementi presenti nella scena (Figura 18) e di aumentare il livello di privacy della soluzione (i volti delle persone sono difficilmente riconoscibili).

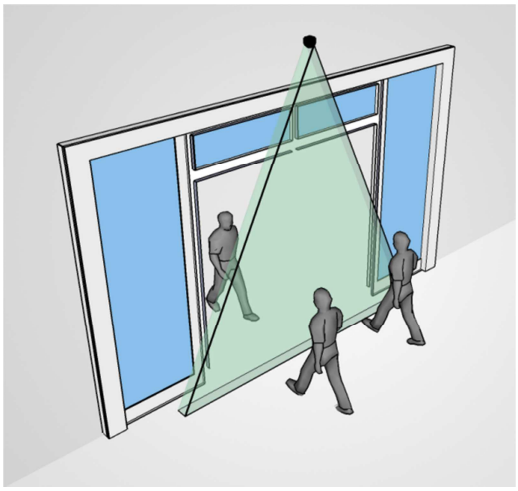


Figura 17 - Conteggio persone da telecamera zenitale

Figura 18 - Casi di occlusione con telecamera non in posizione zenitale

Il metodo è progettato per raggiungere una elevata accuratezza (sia in scenari *indoor* che *outdoor*) con un utilizzo ridotto di risorse

computazionali e di memoria. Si riportano i risultati di una approfondita valutazione del metodo tenendo in considerazione diversi fattori che possono avere un significativo impatto sulle prestazioni: la tecnologia utilizzata per l'acquisizione del flusso (due tipologie di telecamere, un sensore tradizionale *RGB* e un sensore di profondità), lo scenario (*indoor* e *outdoor*), la tipologia degli attraversamenti (singoli o di gruppo), il *frame rate* di acquisizione (da *5* a *30 FPS*) e la risoluzione del *frame* (*160*120, 320*240, 640*480*). Il confronto con altri approcci presenti in letteratura conferma l'efficacia del metodo proposto. Inoltre la sperimentazione su tre differenti CPU attesta la possibilità di utilizzo del metodo sia lato *server-side* che *edge-side (*su *embedded smart camera* con ridotte risorse computazionali e di memoria). In particolare la sperimentazione si effettua su un *dataset* appositamente realizzato che considera i diversi fattori precedentemente elencati.

3.1 Analisi della Letteratura

In letteratura sono presenti diversi metodi per il conteggio delle persone con telecamera zenitale. Una generica architettura condivisa da numerosi metodi si articola nelle seguenti fasi:

1. *Moving object detection,* in cui si utilizzano algoritmi di *frame differencing* o *background subtraction* per separare i *pixel* in movimento dalla parte statica dell'immagine;
2. *People detection,* dove si rilevano le persone presenti nella scena a partire dalla *foreground mask* ottenuta nella fase precedente;
3. *Tracking,* in cui si utilizzano algoritmi di *object tracking* per determinare le traiettorie di ogni persona individuata e aggiornare il relativo contatore nel caso di attraversamento della linea virtuale.

Le principali differenze risiedono negli approcci utilizzati in ognuna delle fasi precedenti e dal tipo di sensore impiegato (il sensore *RGB* tradizionale o i sensori di profondità).

I primi approcci per il *people counting* proposti in letteratura si riferiscono ad algoritmi di analisi video con l'utilizzo delle telecamere *RGB* [58][59][60][61][62]. Diverse soluzioni utilizzano metodi per il *moving object detection* (*frame differencing* [58][59], *background subtraction* [60][61]) e per l'*object tracking* (*overlap tracking* [58][60][61], *optical flow* [59]), con differenze riguardanti l'approccio con il quale si migliora l'accuratezza della *foreground mask.* Nel metodi proposti in [60][58][59] l'algoritmo di *tracking* viene applicato dopo l'utilizzo di un algoritmo per la ricerca delle componenti connesse, questo approccio tende ad avere prestazioni basse nel caso di passaggi di gruppi di persone. Di conseguenza, i ricercatori hanno proposto negli anni algoritmi più complessi per rafforzare la fase di rilevamento delle persone. In [61] si adotta un algoritmo per la segmentazione delle persone basato su *k-means*, al fine di ridurre gli errori durante la fase di *detection.* Tuttavia, questa tecnica, seppur con un aumento dell'accuratezza rispetto alle soluzioni precedenti, non risolve i problemi in scene affollate. Di recente, Mukherjee et al. [62] propongono un metodo per il rilevamento delle teste tramite rilevamento dei bordi di ogni regione (*blob detection*) e ricerca di possibili cerchi (trasformata di *Hough*) all'interno del *frame*. In seguito, il conteggio degli attraversamenti viene completato applicando un algoritmo di *tracking* (*optical flow)*. Le principali limitazioni di questo metodo riguardano l'utilizzo della trasformata di *Hough* in scene affollate con cambiamenti di luminosità, ombre, riflessi e oggetti in movimento che presentano caratteristiche cromatiche simili a quelle del *background* (*camouflage problem*). Per limitare queste problematiche, sono stati proposti metodi per lavorare con telecamere di profondità (ad ogni *pixel* è associato un valore che corrisponde alla distanza, generalmente in millimetri, dell'oggetto

rispetto al sensore). I primi approcci si basavano su l'utilizzo delle telecamere *stereo* [63][64][65]. In seguito, il sensore di profondità Microsoft *Kinect* (*structured light*), per dettagli si rimanda al paragrafo 5.2, ha permesso ai ricercatori di proporre nuovi approcci che utilizzano questo tipo di *device*. In particolare, Zhang et al. [66] definiscono un metodo per il rilevamento delle teste delle persone ricercando i minimi locali all'interno dell'immagine di profondità e successivamente il conteggio tramite algoritmo di tracking; mentre in Vera et al. [67] si propone un metodo che utilizza un classificatore SVM *(Support Vector Machines)* e un *histograms of oriented gradients descriptor* per il rilevamento delle persone e il successivo conteggio.

Tra i lavori presentati in letteratura nell'ultimo biennio sul *people counting*, di seguito si riportano quelli che sfruttano le caratteristiche di un'installazione della telecamera in posizione zenitale. In Li et al. [84] gli autori affrontano il problema sviluppando una soluzione su una piattaforma *embedded*, utile ad esempio per applicazioni che effettuano il conteggio automatico dei passeggeri su un autobus o un treno. La soluzione proposta utilizza il sensore *Microsoft Kinect v2* e la piattaforma embedded *NVIDIA Jetson TK1* con quattro *core 32-bit ARM Cortex-A15 general purpose* a *2.3GHz* e una *GPU (192 Kepler a 852MHz)*. Il sensore Kinect v2 è una telecamera con tecnologia ToF (*Time of Flight*) che presenta un'accuratezza dell'immagine di profondità maggiore rispetto al Kinect v1. Per quanto riguarda il conteggio delle persone, gli autori utilizzano in prima analisi l'algoritmo di *background subtraction* di Wateosot et al. [85], in seguito applicano l'algoritmo di Zhang et al. [66] ed infine l'algoritmo di *tracking* per effettuare il successivo conteggio. Le sperimentazioni condotte riguardano altre piattaforme hardware [86] e un altro algoritmo [87] al fine di validare l'accuratezza del metodo proposto, che si registra intorno al *97%*. Ugualmente Kuo et al. [88] utilizzano il sensore Kinect v2 per lavorare sulle immagini di profondità. In

particolare, l'algoritmo proposto utilizza le informazioni di profondità nella fase di *preprocessing* per ottenere gli oggetti di *foreground*. In accordo ai *pixel* di *foreground* viene trasformata l'immagine a colori in livelli di grigio (con una riduzione dell'onere computazionale per le fasi successive). Nella fase di *detection* la *Hough Circle Transform* [89] viene utilizzata per rilevare i candidati sia nell'immagine di profondità che in quella a livelli di grigio. In seguito, i candidati sono verificati da un algoritmo di rilevamento e di *tracking* delle teste (*elliptical* e Ω*shape*). Con l'operazione di rilevamento delle spalle il metodo decide se un candidato è una "persona" o "non-persona". Infine, mediante una linea virtuale realizza il conteggio degli individui che attraversano la scena. La sperimentazione, effettuata con un proprio *dataset*, con circa *100* rilevamenti, mostra una precisione complessiva di circa il *98%*. In Kocak et al. [90] gli autori trattano il problema utilizzando una GPU (*parallel computing*), in particolare la *GeForce GT 630 2 GB* e la *GeForce GTX 550 Ti*, al fine di permettere l'elaborazione *real-time* anche agli algoritmi onerosi dal punto di vista computazionale. L'algoritmo trasforma in *gray-scale* l'immagine acquisita, applica un *mean filter* per ridurre il rumore e un algoritmo di *frame difference* (differenza tra il *frame* corrente ed il precedente) seguito da un'operazione di sogliatura per ottenere la *foreground mask*. Alla *foregrouns mask* si applica un operatore morfologico di apertura (operazione di erosione a cui segue un'operazione di dilatazione) per ridurne il rumore. Nella fase successiva gli autori impiegano un algoritmo di ricerca delle componenti connesse con una sogliatura sulle dimensioni del blob per filtrare gli oggetti in movimento di tipo "non-persona". Infine, per il conteggio degli individui in transito nella scena gli autori sperimentano due soluzioni, la prima con un'unica linea virtuale mentre la seconda con due linee virtuali. I risultati sperimentali (su un totale di circa *100* attraversamenti) evidenziano come l'accuratezza di entrambe le soluzioni dipenda dalla distanza tra le persone presenti nella scena e dalla risoluzione di acquisizione del flusso video. In Vera et al. [91]

gli autori propongono un metodo basato sulla cooperazione di più telecamere di profondità installate in posizione zenitale. In particolare, il metodo rileva singolarmente per ogni telecamera le persone presenti nella scena e successivamente combina i risultati (considerando la distanza tra le telecamere e il *timestamp* di rilevamento) per ottenere un'unica soluzione. La *detection* di ogni individuo avviene ricercando i minimi locali (*head detection*) all'interno dell'immagine di profondità, filtrati successivamente all'applicazione di un algoritmo di *background subtraction* [23]. Le regioni risultati sono considerate appartenenti alla classe "persona" se soddisfano criteri basati sulla dimensione e la forma. Infine gli autori applicano un algoritmo di *tracking* combinando i risultati delle diverse telecamere per effettuare il conteggio degli individui che transitano all'interno della scena. Per la sperimentazione gli autori impiegano sensori *Kinect* e analizzano l'algoritmo su un totale di circa *300* attraversamenti con un'accuratezza pari a circa il *98%* ed un significativo aumento della *recall* rispetto alla *detection* delle singole telecamere. I falsi negativi ed i falsi positivi sono presenti nel caso di persone ravvicinate, con presenza di oggetti in mano o con cappelli e berretti. In Zhu et al. [172] gli autori propongono un metodo per il rilevamento di comportamenti anomali all'interno di un ascensore. In primo luogo, il metodo rileva la presenza di individui nella scena con il supporto di un algoritmo di *background subtraction*. Successivamente gli autori presentano alcuni algoritmi per rilevare diversi comportamenti anomali da parte degli individui (ad es. una persona che cade o con comportamenti violenti). Le sperimentazioni condotte mostrano buone prestazioni dell'algoritmo. In Stahlschmidt et al. [92] gli autori propongono un metodo per la *detection* ed il *tracking* delle persone utilizzando le immagini di profondità acquisite da una telecamera ToF installata in posizione zenitale. Ad una prima fase di *preprocessing* per ridurre il rumore introdotto dal sensore, segue una fase di *people detection* basata sul *matched filter* [93] e una fase di *tracking* tramite filtro di *Karman* lineare. Per la sperimentazione gli autori realizzano

diverse applicazioni tra le quali è presente quella di conteggio persone che attraversano una linea virtuale. In Perng et al. [94] gli autori propongono un metodo per il conteggio dei passeggeri in entrata ed in uscita ad un autobus, utilizzando una telecamera tradizionale (RGB) installata in posizione zenitale. Il metodo proposto si basa sull'algoritmo di *frame difference* per rilevare gli oggetti in movimento. In seguito i singoli blob, individuati tramite un algoritmo per la ricerca delle componenti connesse, sono analizzati da un punto di vista geometrico per filtrare solo quelli con le caratteristiche desiderate. Infine, gli autori applicano un algoritmo di *tracking* al fine di contare le persone che attraversano una linea virtuale. La sperimentazione condotta dagli autori, con flussi video acquisiti da un autobus, attesta un'accuratezza della soluzione pari a circa l'*87%*.

3.2 Metodo Proposto

Nel seguito si descrive un **nuovo metodo per il conteggio automatico delle persone con l'obiettivo di ottenere un'implementazione efficiente e con un'accuratezza elevata**. Il metodo è presentato nei seguenti lavori:

- *Del Pizzo, L., Foggia, P., Greco, A., Percannella, G., & Vento, M. (2015, June). A versatile and effective method for counting people on either RGB or depth overhead cameras. In Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on (pp. 1-6). IEEE.*
- *Del Pizzo, L., Foggia, P., Greco, A., Percannella, G., & Vento, M. (2016). Counting people by RGB or depth overhead cameras. Pattern Recognition Letters, 81, 41-50.*

Il metodo proposto utilizza in prima analisi un algoritmo di *background subtraction* per determinare la *foreground mask*. In seguito, diversamente dai principali metodi presenti in letteratura, non si effettua la *detection* delle persone e non si applica alcun algoritmo

di *tracking*. In particolare, il conteggio si basa su un sensore virtuale appositamente progettato per garantire una buona accuratezza contro gli errori presenti nella *foreground mask* e per superare i limiti degli algoritmi di *people counting* nel caso di attraversamenti di due o più persone.

Un applicazione di analisi video per il conteggio degli individui che attraversano una linea virtuale, dovrebbe essere in grado di determinare il numero esatto e la relativa direzione di attraversamento. Per la realizzazione del metodo proposto, si è escluso l'utilizzo di un approccio basato sul rilevamento delle caratteristiche di un persona all'interno della scena (come ad esempio il riconoscimento della forma della testa o delle spalle) poiché sono, in genere, soluzioni computazionalmente onerose. Inoltre si è scartato l'utilizzo di un algoritmo di *tracking,* poiché presenta problemi dovuti alla precisione della *foreground mask* (*splits* o *merges* delle componenti connesse), e dell'algoritmo di rilevamento delle componenti connesse poiché, come indicato nel capitolo precedente, lavorando su tutti i *pixel* dell'immagine risulta computazionalmente oneroso.

Il metodo proposto si compone di tre moduli principali: un modulo di *preprocessing* per la riduzione del rumore, un modulo di *background subtraction* per la ricerca degli oggetti in movimento all'interno della scena e un modulo di *counting* che, a partire dalla *foreground mask*, effettua il conteggio delle persone presenti nella scena. La Figura 19 illustra i predetti passi, indicando con *I* il *frame* in uscita dal modulo di *preprocessing*, con *B* il modello del background, con *F* la *foreground mask,* con *t* l'istante corrente e con *t-1* l'istante precedente.

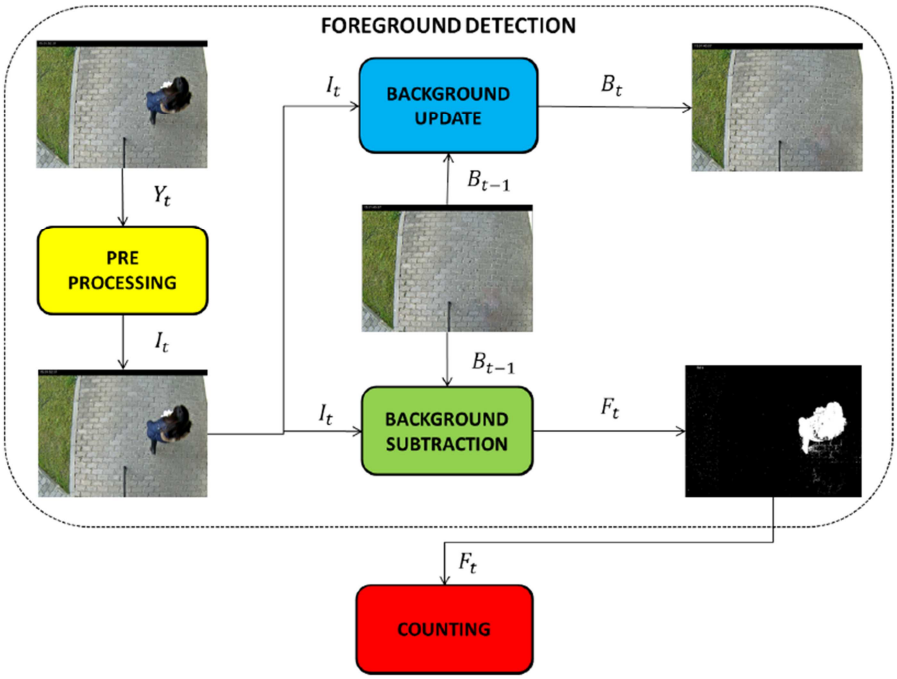


Figura 19 - Architettura del metodo proposto per il *people counting*

Le immagini catturate in scenari *indoor* o *outdoor* presentano rumore causato da vibrazioni o interferenze. Al fine di ridurre tale rumore si utilizza un filtro gaussiano nel modulo di *preprocessing*.

Il modulo di *background subtraction* corrisponde al metodo analizzato nel Capitolo 2, incluse le ottimizzazioni presentate.

Nel modulo di *counting*, si propone un nuovo sensore virtuale robusto contro gli errori presenti nella *foreground mask* e con un ridotto onere computazionale (con operazioni solo su interi). Inoltre, per stimare la direzione degli oggetti in movimento si usa una tecnica che produce meno informazioni rispetto a quelle derivabili da un algoritmo di *tracking* (per esempio, non vi è alcuna indicazione sulla velocità degli oggetti) ma si presenta più robusta ed efficiente per il conteggio. Il sensore presentato è caratterizzato da un'area rettangolare e una direzione di attraversamento, come ripotato in Figura 20.

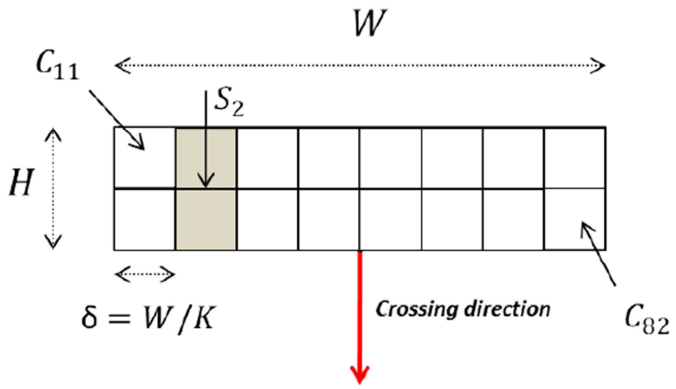


Figura 20 - Un sensore per il conteggio delle persone con 8 strisce

Con *W* si definisce la larghezza del sensore (lato perpendicolare alla direzione di attraversamento), mentre con *H* l'altezza del sensore (lato parallelo alla direzione di attraversamento). Il sensore è diviso in lunghezza in *K* strisce con una larghezza di $\delta=W/K$. Ogni striscia è divisa in altezza in due celle (*H/2*). Si indica con S_i $(i = 1,\ldots,K)$ una generica striscia del sensore e con C_{ij} $(j = 1,2)$ le due celle che appartengono alla striscia S_i. Al tempo *t* una cella C_{ij} è attiva se il numero di *pixel* di *foreground* all'interno della cella supera il prodotto tra il numero totale di *pixel* della cella, detto $Area(C_{ij})$ e una soglia $\theta_c \in [0, 1]$, quindi una percentuale sul totale dei *pixel* della cella, come di seguito riportato:

$$A_{ij} = \begin{cases} 1, & \text{if } F_t(C_{ij}) \geq \theta_C \cdot Area(C_{ij}). \\ 0, & \text{otherwise.} \end{cases}$$

L'algoritmo memorizza per ogni cella il valore attuale di attivazione A_{ij} e il precedente valore A'_{ij}, mentre per ogni striscia si memorizza un valore di attivazione B_i, inizializzato a *0*. Il valore di B_i varia da *0* a *1* in accordo alla seguente condizione:

$$B_i \leftarrow 1 \text{ if } A_{i2} = 1 \wedge A'_{i1} = 1 \wedge A'_{i2} = 0$$

Tale condizione consente di separare, con una buona affidabilità, casi in cui una persona attraversa la scena con un oggetto di medie/grandi dimensioni (ad esempio un carrello o una valigia) da attraversamenti di due o più persone una dietro l'altra. Questa condizione da sola non è però sufficiente a risolvere il problema del conteggio di persone ravvicinate. Per questa ragione, l'algoritmo ricerca continuamente sequenze di celle che hanno un valore di attivazione uguale a *1*, cioè tutte le coppie di indici p, q con $p \leq q$, tali che $B_p = B_{p+1} = \cdots = B_q = 1$ e $B_{p-1} = B_{q+1} = 0$. L'algoritmo disabilita una striscia solo quando anche le strisce adiacenti sono disabilitate e conta un numero di persone per ogni gruppo di strisce con lunghezza $L = (q - p + 1)$ maggiore di una soglia θ_k, pari al rapporto L/θ_k. Quindi θ_k rappresenta il numero minimo di strisce consecutive per l'attivazione del sensore (pari al numero di strisce che occupa mediamente una persona). In questo modo, l'algoritmo è in grado di risolvere i casi in cui gli attraversamenti avvengono tra persone vicine tra di loro.

```
IF A_{p,1} = A_{p+1,1} = ... = A_{q,1} = 0 THEN
    B_i ← 0 FOR i = p, ..., q
    IF q − p + 1 ≥ θ_K THEN
        L = q − p + 1
        Increase counting by floor(L/θ_K)
    END IF
END IF
```

I parametri richiesti in fase di configurazione dell'algoritmo, sono *W*, *H*, θ_c, θ_k, mentre i parametri *K* e δ si ottengono dai precedenti. In generale, *W* corrisponde alla larghezza del varco quindi è un parametro esterno che dipende dallo scenario di riferimento. *H* deve essere configurato considerando il miglior compromesso tra due richieste contrastanti: ad una altezza maggiore corrisponde una richiesta inferiore di *frame rate* del flusso video (un requisito importate soprattutto per le elaborazioni su sistemi a ridotte risorse computazionali o quando si ha la necessità di elaborare il maggior

numero di flussi possibile su un singolo server), mentre ad un'altezza inferiore corrisponde una maggiore capacità del sensore di rilevare correttamente passaggi con persone ravvicinate. Dalle sperimentazioni effettuate su sequenze di *frame* non incluse nel *dataset* di test, si raggiunge un buon compromesso impostando l'altezza pari al doppio della profondità media di una persona e settando $\theta_c=0.2$ e $\theta_k=3$. In Figura 21 si mostrano alcuni esempi di comportamento del sensore in situazioni tipiche con attraversamenti di una o più persone. Il sensore si compone di *8* strisce orizzontali e il numero minimo di strisce consecutive per l'attivazione del sensore è pari a 3. Ogni esempio si compone di una coppia di immagini, quella di destra mostra l'attivazione del sensore al tempo *t-1* mentre l'immagine di sinistra l'attivazione al tempo *t*, le celle attivate al tempo *t-1* sono evidenziate in giallo mentre le celle attivate al tempo *t* in rosso. Nell'esempio 1, una persona attiva tre strisce adiacenti, mentre nell'esempio 2, due persone attivano due gruppi differenti di tre strisce adiacenti. Nel terzo esempio, due persone camminano ravvicinate e quindi attivano un unico gruppo di sei strisce consecutive. Infine, nell'esempio 4, le persone camminano in direzione opposta e attivano al tempo *t-1* due gruppi di tre celle consecutive, al tempo *t* il sensore valuta la sequenza di attivazione delle strisce ed è capace di rilevare la direzione di attraversamento delle persone. In tutti i casi precedentemente esposti, il metodo è in grado di contare correttamente il numero di persone che attraversano il sensore.

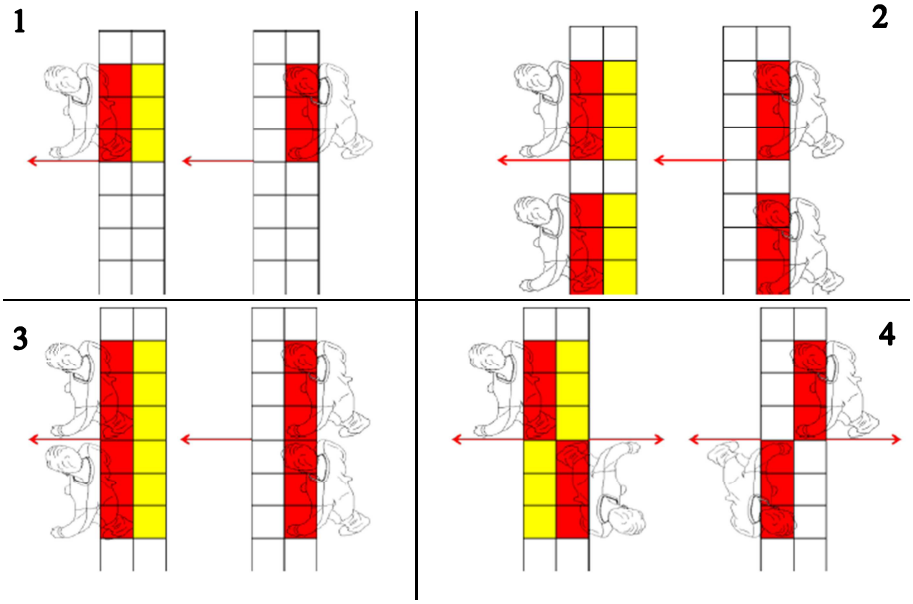


Figura 21 - Esempi di comportamento del sensore con uno o più attraversamenti

Durante le fase di progettazione e di implementazione del suddetto metodo, si è tenuto in considerazione, non solo all'accuratezza, anche l'efficienza computazionale della soluzione al fine di permettere l'elaborazione in *real-time* su sistemi *embedded* e, in generale, su piattaforme con ridotte risorse computazionali e di memoria. Per raggiungere questo risultato, si sfruttano le ottimizzazioni viste nel capitolo precedente ed inoltre si realizzano le seguenti ulteriori ottimizzazioni riferite l'algoritmo di *background subtraction*:

- ✓ si introduce la fase di *preprocessing* per ridurre il rumore e quindi aumentare l'accuratezza della soluzione. L'implementazione sfrutta le istruzioni SIMD tramite la libreria RAPP, al fine di ridurre l'onere computazionale;
- ✓ la fase di *background update* si richiama con una frequenza pari al *10%* di quella con cui opera il metodo di partenza. Difatti, nelle sperimentazioni condotte si ottiene un degrado trascurabile delle prestazioni della *foreground mask* a fronte di una riduzione dell'onere computazionale.

3.3 Descrizione del Dataset

La sperimentazione del metodo di conteggio delle persone si effettua su un *dataset* appositamente realizzato. Il *dataset* risponde a quanto richiesto dai sistemi di analisi video in ambienti reali: analizzare il comportamento del metodo al variare di fattori ambientali e non, che possono avere un significativo impatto sulle prestazioni, il tutto con l'obiettivo di raggiungere le prestazioni migliori con una minima complessità spaziale e temporale. In conseguenza di ciò, il *dataset* realizzato considera: la tecnologia di acquisizione dei *frame* (diverse tipologie di telecamere), la sorgente di illuminazione, la densità degli attraversamento (singoli e in gruppi, in entrambe le direzioni), la risoluzione delle immagini e il *frame rate* di acquisizione. Il *dataset* si definisce come $Dataset_C$, dove *C* sta per *Controlled*, ad indicare che l'acquisizione è avvenuta in un ambiente controllato dove ciascuna persona che attraversa la scena ha ricevuto specifiche istruzioni sulla modalità di attraversamento (passaggi singoli in entrambe le direzioni, a gruppi con stessa direzione, a gruppi in direzioni opposte).

Inoltre, al fine di valutare le prestazioni del metodo in uno scenario il più realistico possibile, si utilizza un secondo *dataset* in cui le persone attraversano la scena senza alcuna istruzione. Per questo motivo, nel seguito si denota $Dataset_U$ dove *U* sta per *Uncontrolled.*

Il $Dataset_C$ comprende sequenze video acquisite contemporaneamente con due *device*. Si utilizza una telecamera ottica tradizionale (sensore *RGB*) e un sensore di profondità, il *Microsoft Kinect*, che fornisce immagini a *16bits* dove l'intensità del *pixel* corrisponde alla distanza, espressa in millimetri, della telecamera dall'oggetto a cui il *pixel* corrisponde, e lavora nel *range* tra *80cm* e *400cm* (sensore *DEPTH*). I due sensori sono installati su un braccio meccanico, in modo da massimizzare la sovrapposizione tra le due viste (si veda Figura 22). Le telecamere sono ad un altezza di circa *3m* dal piano in posizione zenitale. I video di entrambi i *device* sono acquisiti a *30FPS* e con

risoluzione *640*480*. Questi sono scalati rispetto al *frame rate* (*25,20, 15, 10 e 5 FPS*) e alla risoluzione (*320*240 e 160*120*) al fine di studiarne l'impatto sulle prestazioni.

Figura 22 - Installazione dei sensori *RGB* e *DEPTH*

Il $Dataset_C$ presenta sequenze acquisite in due diversi scenari, il primo in cui la luce naturale è prevalente (scenario *outdoor*) e il secondo in cui la sorgente di illuminazione è esclusivamente artificiale (scenario *indoor*). In Figura 23 si riportano due *frame*, il primo appartiene allo scenario *indoor*, mentre il secondo allo scenario *outdoor*. Si sottolinea che, la realizzazione di un *dataset* in entrambi gli scenari permette di ottenere risultati della sperimentazione in linea con quelli conseguibili in ambienti in cui queste applicazioni sono realmente installate. In particolare, lo scenario *indoor* considera installazioni all'interno di aree chiuse (ad esempio un negozio o un centro commerciale) in cui l'obiettivo è contare il numero di persone in entrata e in uscita; in tal caso l'illuminazione è assicurata da un sistema artificiale che genera

un notevole riflesso sul pavimento. Al contrario, lo scenario *outdoor* è caratterizzato dalle variazioni di luminosità generate dalla luce solare.

Figura 23 - Scenario *indoor* e *outdoor*

Il $Dataset_C$ presenta sequenze con un incremento delle persone che attraversano la scena nella stessa direzione o in direzioni opposte. Tali sequenze permettono di valutare le *performance* del metodo in accordo alle diverse condizioni di affollamento. Nei casi più semplici, si ha la presenza di una sola persona che attraversa la scena, mentre nei due casi più complessi si ha un gruppo di sei persone che attraversano l'area nella stessa direzione o in due direzioni opposte (tre persone in una direzione e tre in un'altra). In Figura 24 si presentano alcuni *frame* estratti dal $Dataset_C$: le prime due righe si riferiscono allo scenario *indoor*, mentre le ultime due allo scenario *outdoor*, alternando sensore *RGB* e *DEPTH*. Nella prima colonna si riportano le linee virtuali utilizzate per generare la *ground truth* riferita alle applicazioni di conteggio persone (l'elenco degli attraversamenti presenti nel *dataset* con la relativa direzione di attraversamento) e per il *testing* dell'algoritmo. Le successive colonne riportano esempi di attraversamenti con un incremento del numero di persone. In Figura 25 si presenta il caso più complesso: un gruppo di sei persone presenti nella scena.

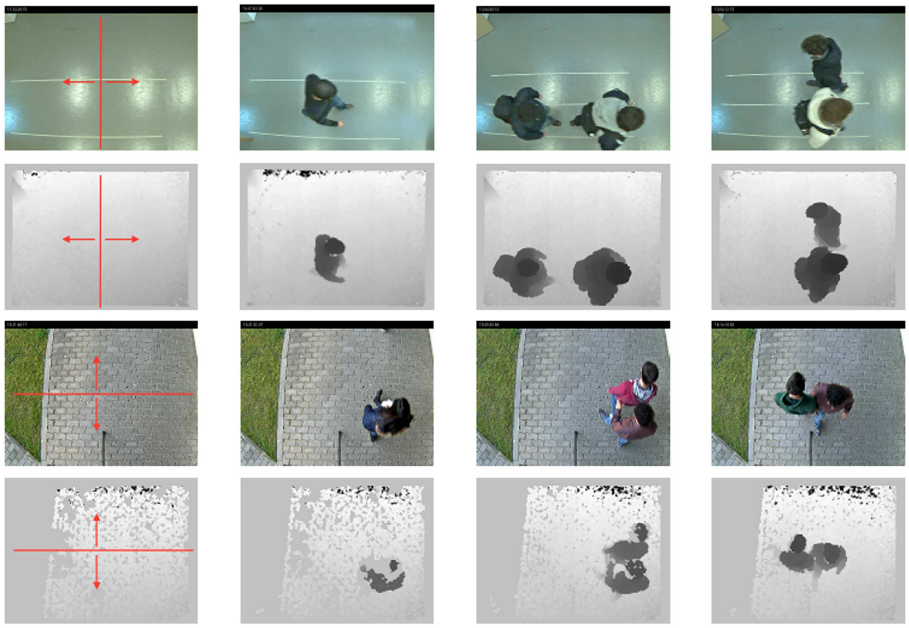


Figura 24 - Esempi di immagini estratte dal *Controlled Dataset*

Figura 25 - Attraversamento di un gruppo di sei persone in scenario *indoor* e *outdoor*

In Tabella 5 si riporta il numero di attraversamenti sulla linea virtuale divisi per tipologia (singoli o di gruppo) e per scenario di riferimento (*indoor* o *outdoor*).

Scenario	Attraversamenti	Numero
indoor	Persone singole	212
	Gruppi di persone	203
outdoor	Persone singole	236
	Gruppi di persone	317

Tabella 5 - Numero di attraversamenti per tipologia (singoli o gruppi) e scenario (*indoor* o *outdoor*)

Il *dataset* realizzato risponde a un duplice obiettivo: effettuare una sperimentazione approfondita dei metodi e rendere disponibile in letteratura una piattaforma completa di comparazione dei metodi che lavorano nell'ambito del conteggio di persone. Pertanto, il dataset è reso pubblicamente disponibile per la comunità scientifica.

Il secondo *dataset* ($Dataset_U$) è realizzato posizionando i sensori *RGB* e *DEPTH* all'ingresso di un corridoio per registrare i passaggi in modalità "non controllata", in cui le persone percorrono la scena senza rispettare uno predeterminato schema. Questo consente di ottenere sequenze non presenti nel $Dataset_C$, riferite a transiti che avvengano in un modo insolito, come ad esempio persone che restano ferme nell'area di interesse, che attraversano la linea virtuale in diagonale, che si fermano e ripartono rapidamente o cambiano la direzione di attraversamento in prossimità della linea virtuale, persone che attraversano la scena con oggetti di medie o grandi dimensioni (ad esempio un carrello o una scala) o con le braccia aperte ad altezza delle spalle e infine gruppi anche di quindici persone che percorrono la scena. Questo *dataset* è realizzato in ambiente *indoor* e contiene *264* attraversamenti singoli e *673* passaggi in gruppi di persone.

Su entrambi i *dataset* è stata realizzata la *ground truth* per la valutazione dei metodi di conteggio delle persone, utilizzando il programma VIPER (*Video Performance Evaluation Resource*) [70]. VIPER è scritto in Java e permette di creare e modificare i *metadata* riferiti alle sequenze di immagini; in particolare, fornisce una interfaccia grafica per visualizzare i singoli *frame* ed assegnare un particolare *object* (nel caso in esame di tipo "persona") con una direzione di attraversamento. Come *output* fornisce un file in formato *.xgtf* per ogni sequenza. Di seguito si riporta un esempio di *ground truth* per una sequenza *indoor*:

```
<?xml version="1.0" encoding="UTF-8"?>
<viper                          xmlns="http://lamp.cfar.umd.edu/viper#"
xmlns:data="http://lamp.cfar.umd.edu/viperdata#">
    <config>
        <descriptor name="Information" type="FILE">
            <attribute        dynamic="false"        name="SOURCETYPE"
type="http://lamp.cfar.umd.edu/viperdata#lvalue">
                <data:lvalue-possibles>
                    <data:lvalue-enum value="SEQUENCE"/>
                    <data:lvalue-enum value="FRAMES"/>
                </data:lvalue-possibles>
            </attribute>
            <attribute         dynamic="false"         name="NUMFRAMES"
type="http://lamp.cfar.umd.edu/viperdata#dvalue"/>
            <attribute         dynamic="false"         name="FRAMERATE"
type="http://lamp.cfar.umd.edu/viperdata#fvalue"/>
            <attribute       dynamic="false"       name="H-FRAME-SIZE"
type="http://lamp.cfar.umd.edu/viperdata#dvalue"/>
            <attribute       dynamic="false"       name="V-FRAME-SIZE"
type="http://lamp.cfar.umd.edu/viperdata#dvalue"/>
        </descriptor>
        <descriptor name="PERSON" type="OBJECT">
            <attribute         dynamic="false"         name="Crossing"
type="http://lamp.cfar.umd.edu/viperdata#lvalue">
                <data:lvalue-possibles>
                    <data:lvalue-enum value="LeftToRight"/>
                    <data:lvalue-enum value="RightToLeft"/>
                </data:lvalue-possibles>
            </attribute>
        </descriptor>
    </config>
    <data>
        <sourcefile filename="video.mkv">
            <file id="0" name="Information">
                <attribute name="SOURCETYPE"/>
                <attribute name="NUMFRAMES">
                    <data:dvalue value="4552"/>
                </attribute>
                <attribute name="FRAMERATE">
```

```
                    <data:fvalue value="30.0"/>
                </attribute>
                <attribute name="H-FRAME-SIZE">
                    <data:dvalue value="640"/>
                </attribute>
                <attribute name="V-FRAME-SIZE">
                    <data:dvalue value="480"/>
                </attribute>
             </file>
            <object framespan="167:167" id="0" name="PERSON">
                <attribute name="Crossing">
                    <data:lvalue value="LeftToRight"/>
                </attribute>
            </object>
        </sourcefile>
    </data>
</viper>
```

All'inizio del file si riportano le informazioni relative alla sequenza (nome, numero di *frame*, *frame rate*, risoluzione) e gli oggetti di interesse (*PERSON*) con i relativi attributi da definire per ogni occorrenza:

- ✓ *frame* corrente (*framespan*);
- ✓ direzione di attraversamento (*crossing attribute*): nelle sequenze *indoor* la direzione di attraversamento può essere *LeftToRight* o *RightToLeft* mentre nelle sequenze *outdoor* la direzione può essere *TopToBottom* o *BottomToTop*.

Di seguito si riportano le principali caratteristiche di tre *dataset* pubblici presenti in letteratura e realizzati per l'ambito di riferimento.

Il *dataset* proposto in Zhang et al. [66], pubblicato nel 2012, è tra quelli più utilizzati in letteratura. Impiega un sensore *Kinect v1* installato in posizione zenitale in ambiente *indoor*:

Caratteristiche	Descrizione
Numero totale di frame	4,334
FPS	Non specificato
Numero totale di rilevamenti	6,094
Scenario di riferimento (indoor / outdoor)	*indoor*
Tipologie di sensori utilizzati per le registrazioni	*Kinect v1*
Altezza installazione	Non specificato
Caratteristiche degli attraversamenti	Attraversamenti singoli e a gruppi di persone, prevalentemente in verticale rispetto allo scenario di riferimento
Caratteristiche delle persone	Persone (maschi e femmine) con diverse caratteristiche in altezza, colore e taglio dei capelli
Presenza di oggetti aggiuntivi nella scena	Nessun oggetto aggiuntivo di altezza rilevante
Persone con e senza accessori	Persone con alcuni oggetti in mano (fogli, borse)
Comportamenti particolari delle persone	Nessun comportamento particolare
Descrizione della ground truth	Immagini con *pixel* rossi corrispondenti alla testa delle persone

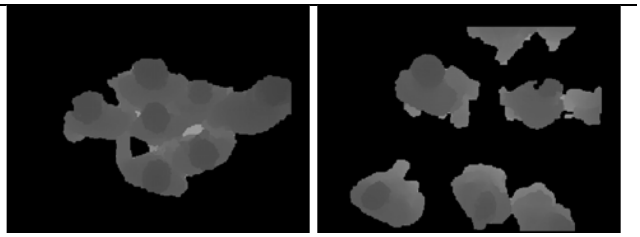
Frame di esempio	

Tabella 6 - Caratteristiche del dataset Zhang et al. [66]

Il dataset proposto in Macias-Guarasa et al. [82], pubblicato nel 2016, utilizza il sensore *Kinect v2* installato in posizione zenitale e in ambiente *indoor*:

Caratteristiche	Descrizione
Numero totale di frame	51,418
FPS	30
Numero totale di rilevamenti	21,093
Scenario di riferimento (indoor / outdoor)	*indoor*
Tipologie di sensori utilizzati per le registrazioni	Microsoft Kinect 2 (depth and infrared data) in posizione zenitale, con le seguenti caratteristiche: • Depth sensing – 512 x 424 – 30 Hz – FOV: 70 x 60 – One mode: 0.5–4.5 meters • 1080p color camera – 30 Hz (15 Hz in low light) • Active infrared (IR) capabilities – 512 x 424 – 30 Hz
Altezza installazione	3.4 metri

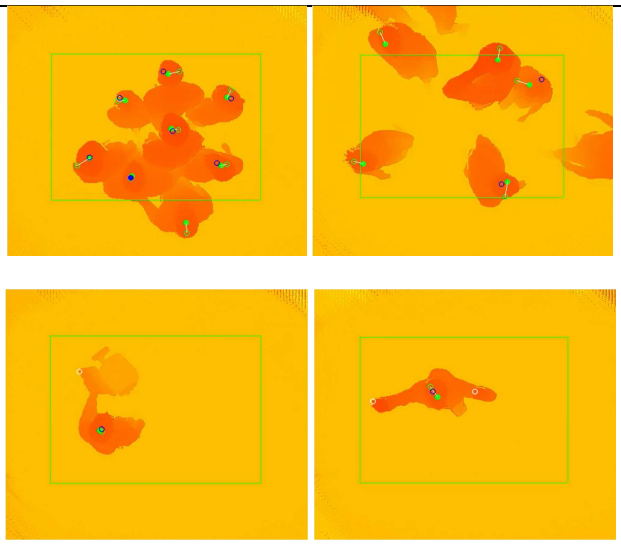
Caratteristiche degli attraversamenti	Attraversamenti singoli e a gruppi di persone (fino a 8 persone presenti contemporaneamente) in modo casuale all'interno della scena
Caratteristiche delle persone	Persone (maschi e femmine) con diverse caratteristiche in altezza, colore e taglio dei capelli
Presenza di oggetti aggiuntivi nella scena	Presenza di sedie nella scena
Persone con e senza accessori	Presenza di persone con cappello e berretto
Comportamenti particolari delle persone	Persone che parlano al telefono e muovono le proprie braccia fino ad aprirle completamente
Descrizione della ground truth	3 punti associati alla testa e 3 rilativi alle spalle di ogni individuo, ottenuti con un processo semi-automatizzato
Frame di esempio	

Tabella 7 - Caratteristiche del dataset Macias-Guarasa et al. [82]

Il *dataset* proposto in Liciotti et al. [97] è stato pubblicato nel 2016. Utilizza il sensore *Asus Xtion Pro Live RGB-D* installato in posizione zenitale e in ambiente *indoor*:

Caratteristiche	Descrizione
Numero totale di frame	60,146
FPS	30
Numero totale di rilevamenti	Si specifica solo il transitano di 100 persone diverse, senza indicare il numero totale di rilevamenti
Scenario di riferimento (indoor / outdoor)	*Indoor* L'illuminazione dell'ambiente non è costante poichè varia durante la giornata e dipende dale condizioni atmosferiche
Tipologie di sensori utilizzati per le registrazioni	*Asus Xtion Pro Live RGB-D* a risoluzione 640x480 installato in posizione zenitale
Altezza installazione	4 m
Caratteristiche degli attraversamenti	Attraversamenti in orizzontale
Caratteristiche delle persone	Le 100 persone presentano un età compresa tra i 19 e i 36 anni, 43 femmine e 57 maschi. 86 con capelli scuri, 12 con capelli chiari e 2 calvi. 55 persone con capelli corti e 43 con capelli lunghi.
Presenza di oggetti aggiuntivi nella scena	Sono presenti scrivanie ai lati degli attraversamenti
Persone con e senza accessori	I soggetti sono registrati indossano magliette, camicie, pantaloni, cappotti, sciarpe e cappelli. In particolare, 18 soggetti indossano cappotti e 7 soggetti sciarpe.
Comportamenti particolari delle persone	Non sono segnalati comportamenti particolari
Descrizione della ground truth	• Distanza pavimento-testa • Distanza pavimento-spalle • Area testa • Circonferenza della testa

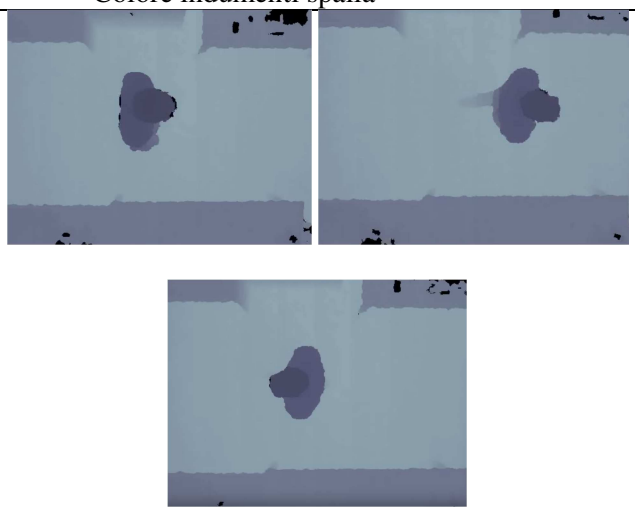
	• Circonferenza delle spalle • Larghezza delle spalle • Profondità anteroposteriore toracica HSV histogram per estrarre due ulteriori feature: ✓ Colore della testa ✓ Colore indumenti spalla
Frame di esempio	

Tabella 8 - Caratteristiche del dataset Liciotti et al. [97]

3.4 Indici Prestazionali e Risultati Sperimentali

Gli indici prestazionali utilizzati per la valutazione dei metodi sono *precision*, *recall* e *f-index*:

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

$$f-index = \frac{2 \cdot Prec \cdot Recall}{Prec+Recall}$$

dove *TP* corrisponde al numero di *true positive* (attraversamenti che sono correttamente rilevati dal metodo), *FP* il numero di *false positive* (attraversamenti erroneamente rilevati dal metodo) e *FN* il numero di *false negative* (attraversamenti non rilevati dal metodo). Al fine di considerare i possibili ritardi tra l'istante di tempo definito dalla *ground truth* e quello riportato dal metodo (dipendente dal tempo di elaborazione), si considera un conteggio corretto se il ritardo è inferiore al secondo (30 *frame*).

Si sottolinea che, i valori dei parametri di configurazione dei metodi oggetto della sperimentazione si ricavano effettuando un *test* su una sequenza separata (con circa *30* attraversamenti per ogni scenario *indoor* e *outdoor*).

Nel seguito, si riporta l'accuratezza e l'onere computazionale della soluzione utilizzando il $Dataset_C$ e il $Dataset_U$ descritti nel paragrafo 3.3 e che comprendono immagini acquisite simultaneamente da due tipologie di sensori in posizione zenitale (un sensore tradizionale e di profondità) con diverse condizioni di luminosità e di affollamento. In più, si valutano le prestazioni al variare del *frame rate* e della risoluzione delle immagini concernenti il $Dataset_C$. Quindi, la valutazione delle prestazioni del metodo proposto si effettua su entrambi i *dataset* ma con obiettivi diversi: nel $Dataset_C$ per caratterizzare l'impatto sulle prestazioni di diversi elementi, come il

livello di densità degli attraversamenti (isolati o in gruppi), lo scenario di riferimento (*indoor* o *outdoor*), il sensore utilizzato (*DEPTH* o *RGB*), la risoluzione e il *frame rate*; mentre con l'utilizzo del $Dataset_U$ si analizzano le prestazioni nel caso in cui le persone attraversano la scena senza seguire un predeterminato schema. Infine, si riportano le prestazioni raggiunte combinando i risultati del metodo con entrambi i sensori. L'utilizzo del *dataset* permette di valutare le prestazioni del metodo in condizioni realistiche, superando le limitazioni della maggior parte dei lavori presenti in letteratura che basano le sperimentazioni su *dataset* non conformi agli scenari reali. Inoltre, si presentano i risultati anche in riferimento all'utilizzo delle risorse in ogni fase dell'algoritmo, effettuando un'analisi su tre architetture differenti, passando dalle piattaforme con risorse limitate (ad es. le *embedded smart camera*) alle *high-end workstations*.

La Tabella 9 presenta i risultati del metodo proposto sul $Dataset_C$ al variare del sensore utilizzato, dello scenario di riferimento, della densità degli attraversamenti e della risoluzione del *frame*, con *frame rate* fissato a *30FPS*. I risultati sono espressi in base agli indice precedentemente definiti: *recall* (Re), *precision* (Pr) e *f-index* (f). Le risoluzione utilizzate sono la *640*480* (di acquisizione) e quelle scalate a *320*240* e *160*120.* Come previsto, si ottiene un degrado delle prestazioni al decrescere della risoluzione: a risoluzione *320*240* si ha una riduzione di circa il *2%* con sensore *RGB* e dell'*1%* con sensore *DEPTH*, mentre a risoluzione *160*120* la riduzione è di circa il *4-5%* per entrambi i sensori. In generale, si ha un'accuratezza maggiore utilizzando il sensore *DEPTH* in confronto ai risultati ottenuti con il sensore *RGB*. Difatti, l'utilizzo del sensore di profondità consente di ottenere una *foreground mask* più accurata, poiché insensibile rispetto ai problemi relativi a cambiamenti di luminosità, ombre, riflessi sul pavimento e *camouflage*. Inoltre, la principale limitazione dell'algoritmo è rappresentata dal numero elevato di *false negative,* in particolare con sensore *RGB* con cui si ha

circa un attraversamento non rilevato ogni dieci. Diversamente, il numero di *false positive* è pari a zero quando si utilizza il sensore *DEPTH*, mentre la *precision* è pari a *0.990* con sensore *RGB*. Oltre a presentare i risultati ottenuti con i singoli sensori, si riportano in tabella anche le prestazioni ottenute combinando i risultati di entrambi i sensori, in modo da massimizzare il conteggio: per ogni passaggio l'output della combinazione "*RGB V DEPTH"* è pari al contatore del singolo sensore con valore più alto. Difatti, poiché il metodo proposto presenta una migliore *precision* rispetto alla *recall*, indipendentemente dal tipo di sensore, la combinazione ha l'obiettivo di aumentare la *recall* conservando un valore elevato della *precision* (come confermato dai risultati riportati in Tabella 9).

			640 x 480			320 x 240			160 x 120		
Sensore	**Scenario**	**Densità**	**Re**	**Pr**	**f**	**Re**	**Pr**	**f**	**Re**	**Pr**	**f**
RGB	***indoor***	**Isolati**	0.967	0.990	0.979	0.934	1.000	0.966	0.849	1.000	0.918
		Gruppi	0.803	0.982	0.883	0.685	1.000	0.813	0.665	1.000	0.799
	outdoor	**Isolati**	1.000	1.000	1.000	0.992	1.000	0.996	0.919	1.000	0.958
		Gruppi	0.868	0.986	0.923	0.852	0.975	0.909	0.864	0.986	0.921
DEPTH	***indoor***	**Isolati**	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		Gruppi	0.975	1.000	0.988	0.961	1.000	0.980	0.956	1.000	0.977
	outdoor	**Isolati**	0.962	1.000	0.981	0.911	1.000	0.953	0.754	1.000	0.860
		Gruppi	0.940	1.000	0.969	0.924	1.000	0.961	0.858	0.996	0.922
RGB* V *DEPTH	***indoor***	**Isolati**	1.000	0.991	0.995	1.000	1.000	1.000	1.000	1.000	1.000
		Gruppi	0.975	0.985	0.980	0.970	1.000	0.985	0.956	1.000	0.977
	outdoor	**Isolati**	1.000	1.000	1.000	1.000	1.000	1.000	0.975	1.000	0.987
		Gruppi	0.984	0.994	0.989	0.968	0.978	0.973	0.953	0.984	0.968

Tabella 9 - Prestazioni del metodo proposto sul *Dataset*$_C$ con tre diverse risoluzioni e *frame* rate fisso a 30FPS

In Tabella 10 riporta i risultati per ogni tipologia di sensore al variare della densità degli attraversamenti (persone singole o gruppi di persone). Nel caso di sensore *RGB*, la densità incide significativamente sull'accuratezza della soluzione causando un degrado in termini di *f-index* pari allo *0.082* nei passaggi di gruppi rispetto agli attraversamenti isolati. Diversamente, le prestazioni ottenute con sensore *DEPTH* si mantengono stabili con una minima variazione dell'*f-index* pari allo *0.014*. Per quanto riguarda la soluzione "*RGB V DEPTH"*, si ottengono le prestazioni migliori con un opportuno equilibrio tra *precision* e *recall.*

Sensore	Densità	Recall	Precision	f-index
RGB	**Isolati**	0.984	0.995	0.990
	Gruppi	0.842	0.984	0.908
DEPTH	**Isolati**	0.980	1.000	0.990
	Gruppi	0.954	1.000	0.976
RGB* V *DEPTH	**Isolati**	1.000	0.995	0.998
	Gruppi	0.980	0.989	0.985

Tabella 10 - Prestazioni sul $Dataset_C$ del metodo proposto di conteggio persone, al variare del sensore utilizzato e della densità degli attraversamenti, con risoluzione 640*480 e 30*FPS*

In Tabella 11 si riporta una vista dei risultati aggregati per tipologia di scenario. Il metodo, con l'utilizzo del sensore *RGB*, presenta un comportamento migliore per lo scenario *outdoor* rispetto a quello *indoor*, anche se con una trascurabile differenza dell'*f-index* pari allo *0.023*. Una differenza simile si può osservare anche con l'utilizzo del sensore *DEPTH* ma in questo caso le *performance* migliori si hanno nello scenario *indoor*; questo risultato è motivato dalla presenza della luce solare che influisce negativamente sul sensore di profondità utilizzato, incrementando il rumore; tale condizione si riflette in una riduzione dell'accuratezza della *foreground mask.* In generale, l'utilizzo del sensore *DEPTH* fornisce le prestazioni migliori rispetto al sensore *RGB*, in particolare per gli attraversamenti di gruppi di persone. Anche in questo caso, la combinazione dei due sensori

fornisce le prestazioni migliori, con un buon bilanciamento tra *precision* e *recall*.

Sensore	Scenario	Recall	Precision	f-index
RGB	**indoor**	0.887	0.987	0.934
	outdoor	0.924	0.992	0.957
DEPTH	**indoor**	0.988	1.000	0.994
	outdoor	0.949	1.000	0.974
RGB* V *DEPTH	**indoor**	0.988	0.988	0.988
	outdoor	0.992	0.997	0.994

Tabella 11 - Prestazioni del metodo di conteggio persone proposto sul *Dataset*$_C$ al variare del sensore utilizzato e dello scenario, con risoluzione 640*480 e 30*FPS*

La Figura 26 mostra i valori dell'*f-index* con l'utilizzo del *Dataset*$_C$ a risoluzione *640*480* al variare del *frame rate* da *30* a *5 FPS* con passo pari a *5*, al fine di valutare l'impatto della variazione del *frame rate* sulle prestazioni del metodo. Come previsto, si ottiene una riduzione dell'accuratezza al diminuire del *frame rate*. Le prestazioni restano piuttosto stabili nel *range 30-15 FPS*, mentre tra i *15-5 FPS* si ha una significativa riduzione. Questo poichè con *frame rate* basso la velocità media di attraversamento delle persone diviene troppo elevata, con una mancata attivazione delle celle del sensore linea virtuale secondo la sequenza di conteggio.

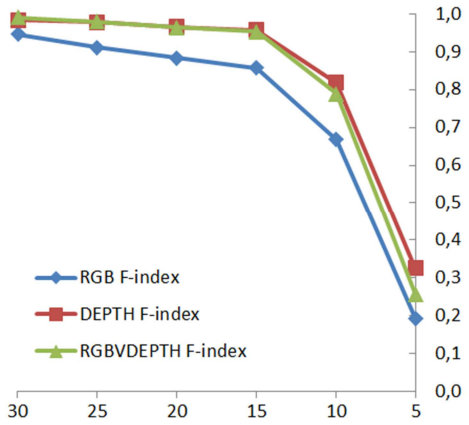


Figura 26 - Prestazioni del metodo su *Dataset*$_C$ al variare del *frame rate*

Le osservazioni fatte per il $Dataset_C$ si ritrovano anche analizzando i risultati del metodo proposto sul $Dataset_U$, presentati in Tabella 12. In tal caso l'utilizzo del sensore *DEPTH* fornisce prestazioni migliori rispetto al sensore *RGB* con una differenza più elevata rispetto al $Dataset_C$. In generale, si assiste ad un involuzione delle prestazioni rispetto a quelle ottenute con il $Dataset_C$. Da un'analisi dettagliata degli errori, le cause principali della riduzione dell'accuratezza sono da attribuire a:

- ✓ illuminazione non uniforme: lo scenario di riferimento del $Dataset_U$ (*indoor*) è caratterizzato da un'illuminazione proveniente da diverse fonti che portano a differenti ombre intorno alle persone. Questa condizione ha un impatto rilevante nel caso di utilizzo del sensore *RGB*, con un incremento dei *false positive* (con le ombre le persone occupano più strisce del sensore linea virtuale) e dei *false negative* (le persone in fila appaiono con maggiore probabilità come un unico oggetto). Mentre, con l'utilizzo del sensore *DEPTH*, tale condizione non provoca alcun impatto sull'accuratezza.
- ✓ elevata densità di persone che attraversano la scena: il $Dataset_U$ è caratterizzato da circa quindici gruppi da otto a dodici persone che attraversano la scena nella stessa direzione o in direzioni opposte. In questo caso, entrambi i sensori tendono a sottostimare il numero di persone, con una riduzione della *recall* rispetto ai risultati con $Dataset_C$.
- ✓ attraversamenti con oggetti di medie/grandi dimensioni: nel $Dataset_U$ sono presenti anche situazioni particolari (circa dodici) in cui le persone attraversano la linea virtuale con una sedia, una scala, una valigia, un borsone o con le braccia aperte. In tal caso il metodo, indipendentemente dal tipo di sensore utilizzato, genera falsi positivi se la persona trasporta lateralmente l'oggetto in modo tale da attivare più di $2*\theta_k$

strisce. Si noti comunque che queste occorrenze sono poco probabili negli scenari comunemente utilizzati per le applicazioni di conteggio delle persone.

Sensore	Densità	Recall	Precision	f-index
RGB	Isolati	0.947	0.958	0.952
	Gruppi	0.762	0.864	0.810
DEPTH	Isolati	0.973	0.973	0.973
	Gruppi	0.875	0.983	0.926
RGB* V *DEPTH	Isolati	0.981	0.963	0.972
	Gruppi	0.933	0.904	0.918

Tabella 12 - Prestazioni raggiunte dal metodo proposto sul $Dataset_U$

In Figura 27 si riportano *frame* del $Dataset_U$ che presentano alcune delle problematiche precedentemente esposte. La prima immagine in alto riporta il posizionamento del sensore nello scenario di riferimento, mentre le successive immagini presentano alcuni dei casi alternando *frame* proveniente dal sensore *RGB* e *frame* del sensore *DEPTH*.

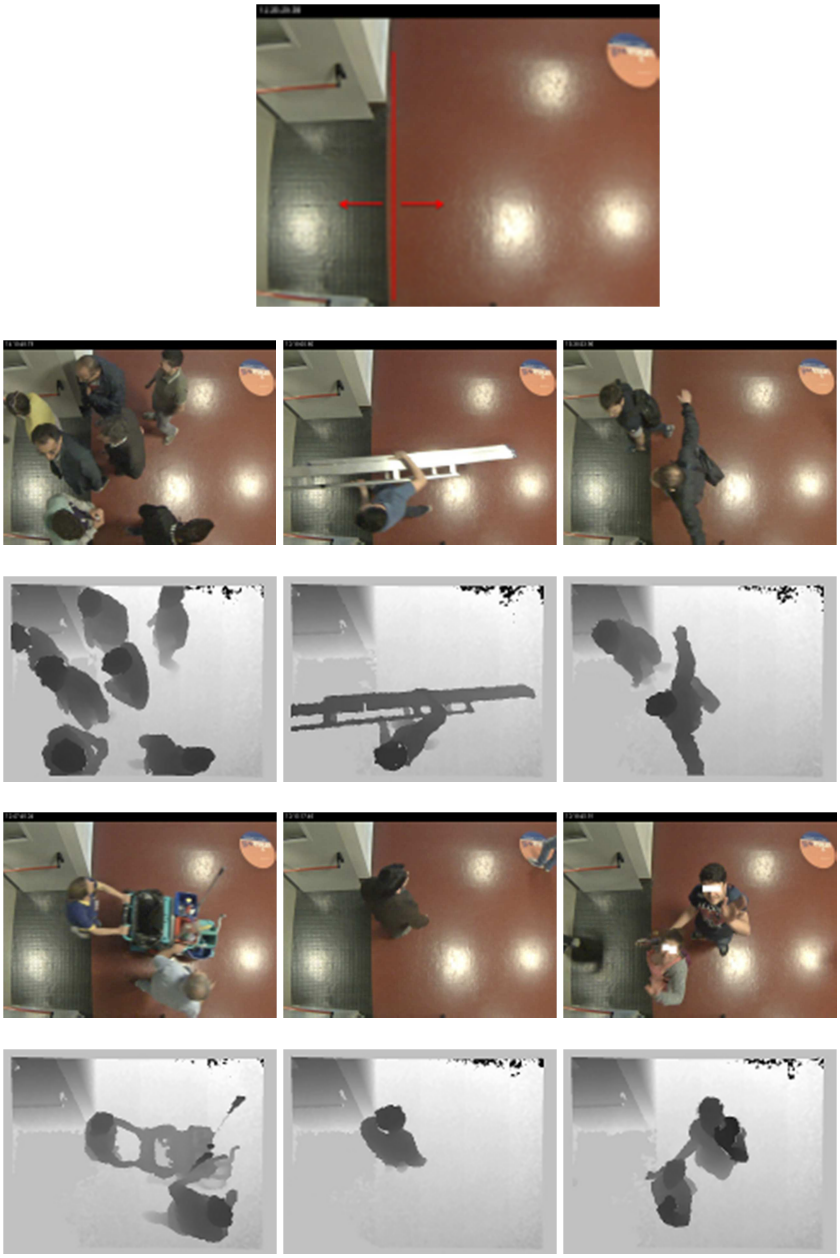


Figura 27 - Esempi di *frame* appartenenti al $Dataset_U$

In una fase successiva della sperimentazione, l'attenzione è ricaduta sul **confronto del metodo proposto con altri due metodi presenti in letteratura**:

- ✓ il metodo presentato in Del Pizzo et al. [68], a cui corrisponde l'algoritmo precedentemente descritto senza la fase di *preprocessing*;
- ✓ il metodo presentato in Chen et al. [58], che effettua il conteggio utilizzando un algoritmo di *tracking*.

In Tabella 13 si riportano i risultati con le sequenze del $Dataset_C$ a *30FPS* e risoluzione a *640*480*. Il metodo proposto con fase di *preprocessing* presenta le migliori prestazioni in termini di *recall* e *precision.* Rispetto al metodo senza la fase di *preprocessing,* si ha un incremento dell'accuratezza sia con sensore *DEPTH* che *RGB*. Il filtro gaussiano permette di ridurre le occorrenze di falsi positivi e falsi negativi dovute alla presenza di rumore nell'immagine. Le prestazioni raggiunte dal metodo basato su *tracking* sono decisamente inferiori, e questo conferma la scelta di non utilizzare questa tipologia di metodi. Difatti, l'utilizzo del *tracking* può causare diversi falsi positivi e falsi negativi quando la *foreground mask* presenta diversi *blob* associati ad un unica persona (*slits*) oppure un unico *blob* associato a più persone (*merge*). Detto ciò, i problemi principali si rilevano con l'utilizzo del sensore *RGB* in scenario *indoor* a causa della presenza dei riflessi sul pavimento e in scenario *outdoor* a causa delle ombre. Utilizzando il sensore *DEPTH*, si ha un miglioramento delle prestazioni (poiché supera i problemi dei riflessi e delle ombre), ma *split* e *merge* dei blob producono comunque un numero elevato di errori nel conteggio delle persone.

Sensore	Metodo	TP	FN	FP	Recall	Precision	f-index
RGB	Metodo proposto con preprocessing	879	89	9	0.908	0.990	0.947
	Metodo proposto senza preprocessing	874	94	20	0.903	0.978	0.939
	Metodo con *tracking*	793	175	177	0.819	0.818	0.818
DEPTH	Metodo proposto con preprocessing	935	33	0	0.966	1.000	0.983
	Metodo proposto senza preprocessing	928	40	0	0.959	1.000	0.979
	Metodo con *tracking*	852	116	77	0.880	0.917	0.898
RGB* V *DEPTH	Metodo proposto con preprocessing	958	10	7	0.990	0.993	0.991

Tabella 13 - Prestazioni sul *Dataset*$_C$ del metodo proposto con *preprocessing*, in confronto al metodo senza *preprocessing* e al metodo con *tracking* delle persone

Infine si riportano i **risultati di un'analisi sull'utilizzo delle risorse computazionali** richieste dal metodo. Per completare la predetta analisi, si considerano le seguenti tre piattaforme:

- ✓ **PC**: *Intel(R) Core(TM) i7-3770S CPU @ 3.10GHz con 4GB di RAM*;
- ✓ **MIPS**: *400MHz Multi-Thread RISC CPU con 0.5GB di RAM*;
- ✓ **ARM**: *ARMv7 Processor rev 5 (v7l) con 1CPU da 600MHz e con 0.8GB di RAM.*

Queste piattaforme rappresentano le architetture adottate nelle installazioni reali. La piattaforma PC è la più utilizzata quando si ha la necessità di connettere un numero elevato di telecamere ad una postazione centrale su cui sono in elaborazione diverse istanze dell'algoritmo (modalità *server-side*). La piattaforma MIPS consente l'elaborazione dell'algoritmo direttamente a bordo camera (modalità *edge-side*). Questa soluzione è ad oggi quella di maggiore interesse poiché consente alla telecamera di rilevare gli eventi di interesse e di inviare ad un server centrale le sole informazioni di conteggio senza dover trasmettere l'intero flusso video per l'elaborazione (riduzione della richiesta di banda della rete), inoltre evita il problema del *single point of failure* (server). Il limite di queste piattaforme è dato dalle ridotte risorse computazionali e di memoria che richiedono una fase di ingegnerizzazione degli algoritmi per poter essere elaborati direttamente a bordo camera. La terza piattaforma (ARM) si colloca, dal punto di vista hardware, tra le due precedenti, con una potenza di elaborazione inferiore al PC e maggiore rispetto alla piattaforma MIPS. Questa piattaforma viene utilizzata quando si ha la necessità di elaborare diversi flussi video di diverse telecamere che non supportano l'elaborazione *embedded*, e il numero di telecamere non è così elevato (tipicamente nell'ordine di dieci unità) ma con gruppi distribuiti su una rete geografica, ad esempio una serie di negozi distribuiti nel mondo dove in ogni negozio sono installati un numero ridotto di sistemi di conteggio delle entrate e delle uscite. La Tabella

14 riporta, per ogni fase del metodo proposto e per ogni piattaforma, il tempo di elaborazione del *frame* espresso in millisecondi e la percentuale di utilizzo della CPU utilizzando le sequenze del $Dataset_C$ a diverse risoluzioni. Analizzando i risultati, si deduce che la fase di *foreground detection* è quella computazionalmente più onerosa, e richiede circa il *60%* del tempo, mentre le ottimizzazioni riportate in particolar modo nella fase di *background update* rendono trascurabile l'impatto di quest'ultima sui tempi di elaborazione. Inoltre, la fase di *preprocessing* ha un ridotto impatto anche se non trascurabile sui tempi di elaborazione con un peso pari a circa il 10%. In particolare, si mostra una evidente differenza, in termini di tempo di elaborazione, tra le diverse piattaforme: difatti si ha un incremento di circa un ordine di grandezza passando dalla soluzione *server-side* (PC) a quella ARM, e quasi un altro ordine di grandezza passando dalla piattaforma ARM a quella *edge-side* (MIPS). Altro parametro di interesse è il *frame rate* al variare della piattaforma e della risoluzione. Il *frame rate* medio si quadruplica al dimezzarsi della risoluzione. Con risoluzione *640*480*, il *frame rate* è circa pari a *189* su un singolo core del server, supera i *700FPS* con risoluzione *320*240* e raggiunge i *2,855FPS* a risoluzione *160*120*. Con piattaforma ARM il metodo raggiunge i *17FPS* con risoluzione massima (valore compatibile con quanto richiesto dalle applicazioni di analisi video intelligente). Mentre *l'FPS* medio su MIPS a risoluzione massima è pari a *5*, valore non sufficiente per ottenere un conteggio affidabile delle persone, tuttavia a risoluzione *320*240* la piattaforma MIPS raggiunge i *23FPS*. Si consideri che, passare dalla risoluzione massima a quelle scalate, comporta una riduzione dell'accuratezza compresa tra il *2%* e il *5%*, quindi la piattaforma MIPS assicura un conteggio affidabile con risoluzione uguale o inferiore a *320*240*, accettando un limitato degrado dell'accuratezza.

Risulta importante indicare che l'implementazione del software non sfrutta la parallelizzazione del codice e quindi la presenza di più *core*

nelle piattaforme ARM e PC. Inoltre, per rendere confrontabili i risultati, su ogni piattaforma si elabora un unico flusso video. Ulteriori esperimenti hanno comunque permesso di verificare che il numero di flussi video, ad un dato *FPS,* scala linearmente con il numero di *core*.

	640 x 480			320 x 240			160 x 120		
Fase	PC	ARM	MIPS	PC	ARM	MIPS	PC	ARM	MIPS
Preprocessing	551	6,623	22,386	136	1,551	4,155	33	364	709
	10.3%	10.9%	10.8%	9.7%	10.0%	9.4%	9.1%	9.1%	7.7%
Foreground Detection	3,241	41,377	1,599,332	821	10,424	34,344	209	2,628	7,099
	60.4%	68.1%	76.8%	58.6%	67.1%	77.5%	58.3%	65.4%	77.4%
Background Update	3	54	4,295	1	17	1,086	1	8	298
	0.1%	0.1%	2.1%	0.1%	0.1%	2.5%	0.2%	0.2%	3.3%
Counting	1,574	12,968	21,347	445	3,542	4,713	116	1,016	1,062
	29.3%	20.9%	10.3%	31.7%	22.8%	10.6%	32.4%	25.3%	11.6%
***FPS* Totale**	189	17	5	728	65	23	2,855	254	109

Tabella 14 - Tempo di elaborazione del *frame* espresso in millisecondi e percentuale di utilizzo della CPU del metodo proposto su $Dataset_C$ a diverse risoluzioni

Capitolo 4. Rilevamento Automatico delle Persone

La *human detection* è il processo di estrapolazione di informazioni relative alle persone collocate nella scena inquadrata dalla telecamera. Le informazioni estratte possono essere di diverso tipo:

1. informazioni spazio-temporali: presenza, conteggio e posizione di ogni persona;
2. informazioni comportamentali di una persona o gruppo di persone;
3. informazioni fisiologiche come età, colore della pelle, peso, temperatura, ecc.

Nel presente paragrafo si focalizza l'attenzione sulla prima categoria di informazioni, in particolare si affronta il problema del *presence detection*.

Rilevare la presenza di persone nella scena tramite algoritmi di analisi video è utile in diversi ambiti applicativi. Si pensi ai sistemi di anti-intrusione, soluzioni per le *smart car*, ovvero per il rilevamento dell'uomo in prossimità di un manipolatore industriale.

I sensori tradizionali per rilevare la presenza di persone all'interno di un locale sono i sensori PIR (*Passive InfraRed*) i quali misurano i raggi infrarossi (IR) irradiati dagli oggetti nel campo di vista (Figura 28). Questa tecnologia non è in grado di rivelare persone immobili sulla scena, può confondere un animale con una persona e non estrae informazioni aggiuntive (come la posizione). Lavorando con i sensori video e algoritmi di analisi video è possibile ottenere una soluzione efficace e flessibile al costo di una spesa aggiuntiva (un sensore PIR è acquistabile ad un prezzo di circa 10€, contro i 100€ di una telecamera IP). Tale costo si assottiglia al crescere dell'area da controllare, difatti una telecamera può analizzare un'area di diversi metri.

Figura 28 - Sensore PIR

Visti i risultati ottenuti nel Capitolo 3, l'utilizzo dell'installazione zenitale e della telecamera di profondità consentono di ottenere soluzioni maggiormente efficaci: riduzione delle occlusioni e della sensibilità ai rapidi cambiamenti di luminosità e un aumento del livello di *privacy*. La Figura 29 presenta un *frame* ottenuto dalla telecamera di profondità con quattro persone presenti nella scena; l'algoritmo di rilevamento determina per ogni persona le coordinate del relativo rettangolo che la contiene.

Nel seguito si descrivono i dettagli della realizzazione e della caratterizzazione delle prestazioni di un **metodo per il rilevamento della presenza e relativa localizzazione delle persone nella scena inquadrata da telecamera fissa di profondità installata in posizione zenitale**. Il metodo proposto è progettato per raggiungere un buon compromesso tra l'accuratezza nel rilevare gli individui e la complessità computazionale. Il metodo proposto adotta un algoritmo di *background subtraction* (ABSM) al fine di rilevare gli oggetti di interesse all'interno della scena, successivamente un algoritmo con un ridotto onere computazionale viene utilizzato per filtrare il rumore dalla *foreground mask* e per determinare la posizione delle persone nella scena. Nella sperimentazione condotta, il metodo è confrontato con due metodi che utilizzano approcci diversi per rilevare la presenza di individui all'interno della scena (il primo più efficiente ma meno accurato del secondo). In particolare si rileva un'accuratezza prossima a quella del miglior metodo di confronto, con una riduzione del tempo di elaborazione di circa un ordine di grandezza.

Le problematiche principali connesse alla *detection* delle persone sono: occlusioni (quando un soggetto è parzialmente o completamente oscurato da altre persone o elementi presenti nella scena), postura della persona, presenza di oggetti aggiuntivi di medie/grandi dimensioni, stili di abbigliamento differenti, la variabilità delle condizioni di luminosità registrabili sia in ambiente interno che esterno [42], la presenza di ombre [104] e della *specular reflection* [105]. Per fronteggiare le predette problematiche, recentemente diversi autori hanno proposto l'utilizzo dei sensori di profondità in posizione zenitale (ad es. Microsoft Kinect) Inoltre, tale sensore consente di superare le regole stringenti riguardanti la privacy poiché risulta impossibile riconoscere i volti delle persone.

Il presente capitolo riporta inoltre il processo di combinazione di tre algoritmi di rilevamento di individui nella scena (il metodo proposto e i due metodi di confronto) attraverso la realizzazione di un **sistema multi-esperto con regole di combinazione per il dominio in questione**. I risultati della sperimentazione condotta dimostrano che tale combinazione permette di raggiungere prestazioni più elevate rispetto ai singoli metodi sia in scenari interni che esterni e con diverse densità di persone presenti nella scena.

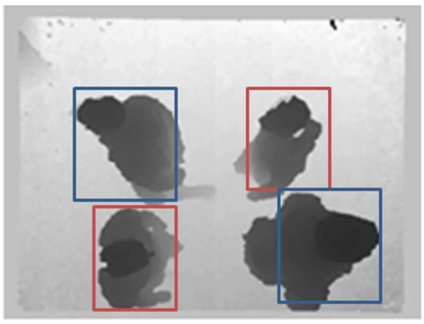


Figura 29 - Rilevamento delle persone presenti nella scena

4.1 Analisi della Letteratura

Il rilevamento delle persone da flussi video è particolarmente importante in molti contesti della *computer vision*, utile soprattutto per applicazioni di sicurezza e *business intelligence*. Esempi di applicazioni dove è richiesto il rilevamento degli individui nella scena sono: il conteggio del numero di persone che attraversano una linea virtuale [98], la stima del numero di persone in un'area [99][100], le statistiche riguardanti il tempo di permanenza delle persone in un'area specifica (come di fronte la vetrina di un negozio, o all'interno di una stanza) [101], il rilevamento di condizioni di sovraffollamento [102] e dei comportamenti anomali [103], ecc.

Gli algoritmi di rilevamento presenti in letteratura, pur essendo variegati, si dividono in due principali categorie: *supervised* e *unsupervised*. La prima categoria richiede una fase di apprendimento di un classificatore a cui si presentano una serie di campioni specificando per ognuno a quale classe appartiene. L'algoritmo di *backpropogation* utilizzato nelle reti neurali è un esempio di questa metodologia. Dietterich et al. [106] presentano le categorie di tecniche di *supervised learning* più popolari e forniscono una serie di metriche per valutarle e compararle. Gli approcci *unsupervised* non presentano una fase di addestramento ma assegnano una classe in accordo a *similarity measures* e *clustering factors*. Le *Self-organizing map* [107], utilizzate con le reti neurali, ne sono un esempio.

In generale, l'output di un classificatore può essere di diversa natura, Xu et al. [108] presentano la seguente caratterizzazione:

- ✓ *Abstract form*: l'output del classificatore è un'unica classe assegnata all'input;
- ✓ *Rank level*: i classificatori forniscono un *ranking* (ordinamento) delle classi;
- ✓ *Measurement level*: il classificatore assegna per ogni input un valore che corrisponde a quanto crede esso appartenga alla

classe. Questi classificatori sono chiamati *probabilistic classifiers*.

Una struttura generalizzabile per la *detection* delle persone si sintetizza nelle seguenti tre fasi:

1. *Preprocessing*: l'immagine in ingresso viene ottimizzata per soddisfare le specifiche dell'algoritmo e per massimizzare le caratteristiche da estrarre.
2. Estrazione delle caratteristiche di interesse: si genera una descrizione dell'immagine che l'algoritmo di classificazione è in grado di valutare.
3. Classificazione: le caratteristiche estratte vengono classificate per determinare il risultato.

In letteratura sono presenti diverse metodologie di estrapolazione delle caratteristiche. In generale, quelle più utilizzate sono:

- ✓ *Shape feature extraction*;
- ✓ *Motion feature extraction*;
- ✓ *Texture extraction techniques*;
- ✓ *Combination of features*.

Gli algoritmi che utilizzano un approccio combinato presentano le prestazioni migliori, al costo di una maggior complessità computazionale; anche se i *shape feature extraction* raggiungono un buon compromesso tra accuratezza e complessità computazionale.

Recentemente, in letteratura sono state proposte diverse soluzioni che utilizzano un *structured light zenithal depth sensor,* come il sensore *Kinect* della *Microsoft*. Zhang et al. [66] presentano un metodo per rilevare le teste delle persone presenti nella scena. Il metodo ricerca le regioni con minimi locali all'interno dell'immagine di profondità e successivamente ne valuta le caratteristiche (numero di *pixel*) per classificare ogni regione. Galčík et al. [71] ricercano le teste delle

persone considerando le regioni che rispettano criteri come dimensione, forma, e appartenenza ad una regione corrispondente alle spalle. Rauter et al. [72] introducono i *simplified local ternary patterns,* un nuovo *feature descriptor* per rilevare la parte testa-spalle di ogni persona all'interno della scena e successivamente utilizza un classificatore SVM. Vera et al. [67] propongono un metodo che utilizza un classificatore SVM per il rilevamento delle persone, con l'impiego del descrittore *histogram of oriented gradients.* Zhu et al. [73] introducono il descrittore *head and shoulder profile (HASP)* basato su *Haar wavelet features* con il quale addestrare un classificatore basato sull'algoritmo di *Adaboost* [74]. Lin et al. [75] partono dall'idea che le persone, in accordo alla posizione zenitale della telecamera, presentano una forma simile ad un ellisse e l'area della parte superiore è inferiore a quella della parte sottostante. A partire da ciò, gli autori sezionano le aree a diversi livelli alla ricerca di quelle corrispondenti alla zona testa-spalle. Tseng et al. [76] utilizzano una *graph-based-segmantation* per rilevare i *pixel* di foreground. L'immagine di profondità viene divisa in regioni collegando i *pixel* che sono valutati "vicini" in base alla *Euclidean distance*. Un ulteriore passaggio permette di unire due o più regioni in un'unica regione che rappresenta la persone presente nella scena. Burbano et al. [77] propongono un sistema composto da una rete di telecamere di profondità in posizione zenitale e per effettuare il rilevamento delle persone si utilizza un *human feature descriptor* che si basa su diverse informazioni come l'altezza, la dimensione della testa, la larghezza delle spalle, la direzione. Nalepa et al. [78] ricercano i minimi locali all'interno dell'immagine di profondità e utilizzano un algoritmo modificato di *flood fill* per creare le regioni a partire dai minimi locali trovati. In seguito il metodo raggruppa le componenti connesse che rappresentano varie parti del corpo in un singolo *blob.*

Tra i lavori presentati in letteratura nell'ultimo biennio sul problema del *people detection*, si evidenziano i seguenti che sfruttano le caratteristiche di un'installazione della telecamera di profondità in posizione zenitale. In Luna et al. [81] gli autori propongono un sistema per la detection delle persone con l'utilizzo di immagini di profondità ottenute dal sensore Kinect v2 installato in posizione zenitale ad un'altezza di 3.4m dal suolo. Gli autori propongono un algoritmo per rilevare i punti di massimo isolati presenti nell'immagine di profondità (local maxima detection), intorno ai quali si definisce una ROI (*Region of Interest*). Il feature vector definito dagli autori consta di sei componenti che vengono estratte dalla ROI. La classificazione dei feature vector avviene utilizzando le tecniche PCA (*Principal Components Analysis*), in cui si determina se il *feature vector* estratto dalla ROI intorno ai punti di massimo appartiene alla classe persone. Per la sperimentazione gli autori utilizzano un proprio dataset, descritto in [82], ottenendo una percentuale di errore in media pari a *3.1%*, anche nel caso di sequenze con un numero elevato di persone presenti nella scena e con soggetti ravvicinati. Gli autori hanno sperimentato l'algoritmo proposto in confronto con il metodo di Stahlschmidt et al. [83] e con un altro dataset presente in letteratura presentato in Zhang et al. [66]. In Li et al. [84] gli autori utilizzano l'algoritmo di Zhang et al. [66] per la fase di people detection su una piattaforma *embedded*. La soluzione proposta utilizza il sensore Kinect v2 e la piattaforma *embedded NVIDIA Jetson TK1* con quattro *core 32-bit ARM Cortex-A15 general purpose* a 2.3GHz e una *GPU (192 Kepler a 852MHz)*. Prima della fase di detection gli autori utilizzano l'algoritmo di *background* subtraction di Wateosot et al. [85] per ottenere la *foreground mask* su cui applicare l'algoritmo di *waterfilling*. In Vera et al. [91] gli autori utilizzano più telecamere di profondità installate in posizione zenitale per ottenere un unico sistema di *detection* delle persone. In particolare, il metodo rileva singolarmente per ogni telecamera le persone presenti nella scena e successivamente combina i risultati (considerando la

distanza tra le telecamere e il timestamp di rilevamente) per ottenere un unico risultato. La *detection* delle singole persone avviene ricercando i minimi locali (*head detection*) all'interno dell'immagine di profondità, filtrati dopo aver applicato un algoritmo di *background subtraction* [23]. Le regioni risultati sono considerate "persona" se soddisfano criteri basati sulla dimensione e la forma. I falsi negativi e falsi positivi sono presenti nel caso di persone troppo vicine, con presenza di oggetti in mano o nel caso indossino cappelli o berretti. In Stahlschmidt et al. [92] gli autori propongono un metodo per la *detection* ed il *tracking* delle persone utilizzando le immagini di prodonfità acquisite da una telecamera ToF installata in posizione zenitale. Ad una prima fase di preprocessing per ridurre il rumore introdotto dal sensore, segue una fase di *people detection* basata sul *matched filter* [93] e una fase di *tracking* delle persone con un filtro di *Karman* lineare. In Kim et al. [95] gli autori effettuano il rilevamento delle persone con telecamera di profondità (*Kinect*) in posizione zenitale per proporre un metodo di stima della posizione di puntamento a schermo di un utente senza l'utilizzo di un *mouse*, utile soprattutto per lavorare con *display* di elevate dimensioni. Nella fase di estrazione del *feature vector* riferito alla testa della persona, gli autori utilizzano *Histogram of Oriented Normal Vectors (HONV)* [96], mentre nella fase di classificazione il metodo utilizza un classificatore SVM. Gli autori propongono un metodo per la segmentazione della testa delle persone che lavora sui punti finali della sagoma di ogni persona (ricavano i punti di flesso utilizzando il metodo *iterative binary partitioning* (IBPM) per determinare la posizione delle mani. Infine, effettuando il *tracking* delle mani delle persone rilevate nella fase precedente e convertendo le coordinate delle mani e della testa nel mondo reale, si ottiene la stima del puntamento.

4.2 Metodo Proposto e Metodi di Confronto

Di seguito si descrive il metodo proposto (denominato ABSM_DETECTION) per il rilevamento delle persone nella scena inquadrata da una telecamera di profondità in posizione zenitale, progettato per raggiungere un'accuratezza elevata con un utilizzo ridotto delle risorse computazionali. Le fasi di cui si compone il metodo sono illustrate in Figura 30.

Figura 30 - Fasi del metodo ABSM_DETECTION

La prima fase corrisponde all'algoritmo di *background subtraction* presentato nel Capitolo 2 (*Adaptive Background Subtraction Method*). Nella fase successiva, la *foreground mask* viene migliorata attraverso una fase di *postprocessing* (un'operazione morfologica di apertura a cui corrisponde un'operazione di erosione seguita da una dilatazione). Nelle ultime due fasi il metodo ricerca le componenti connesse e applica una serie di filtri per selezionare i soli *box* concernenti una persona. In particolare, il primo filtro lavora sulle dimensioni del *box* (lunghezza e altezza) per trascurare quelli al di sotto una certa soglia (*minSize*), successivamente si applica un filtro che determina se il *box* fa riferimento a due o più persone, anche in questo caso considerando le dimensioni del *box* in confronto a una soglia predefinita (*minSizePersons*). I *box* risultanti da quest'ultimo filtro sono divisi in due parti uguali lungo i lati che superano la soglia predefinita. Ogni *box* in uscita dai precedenti filtri viene considerato come valido se il numero di *pixel* di *foreground* inclusi in esso è maggiore di una determinata soglia (*minNumPixels*), così da trascurare i falsi rilevamenti. La Figura 31 presenta tre esempi (uno per ogni riga) di funzionamento del metodo nel caso in cui due o più persone ravvicinate attraversano la scena al punto da implicare un unico *box* in uscita dalla fase di *Blob Detection*. Per ogni esempio si illustrano i

risultati dei tre predetti filtri applicati nella fase *blob filter and split* dell'algoritmo (verifica delle soglie *minSize, minSizePersons* e *minNumPixels*), l'*output* di ogni filtro corrisponde ad una colonna diversa. Nel primo esempio due persone che attraversano la scena e sono rilevate come un unico *box* (prima colonna) che presentando una lunghezza superiore alla soglia viene diviso in due di pari dimensioni (seconda colonna), entrambi superano la soglia sul numero di *pixel* di *foreground*. I successivi due esempi illustrano attraversamenti di due e tre persone in cui il *box* risultante supera la soglia *minSizePersons* sia in lunghezza che in altezza, quindi il metodo lo divide in quattro *box* con medesime dimensioni (seconda colonna). Nell'ultima colonna vengono filtrati i *box* che presentano un numero di *pixel* di *foreground* inferiore alla soglia *minNumPixels* (due *box* gialli nel secondo esempio, un *box* giallo nel terzo). Il metodo eredita le ottimizzazioni su complessità temporale e spaziale analizzati nel capitolo precedente per applicazioni *real-time*, inoltre la fase di filtraggio richiede solo operazioni su interi (conteggio numero di *pixel* di *foreground* e ridimensionamento del *box*) e confronti tra interi.

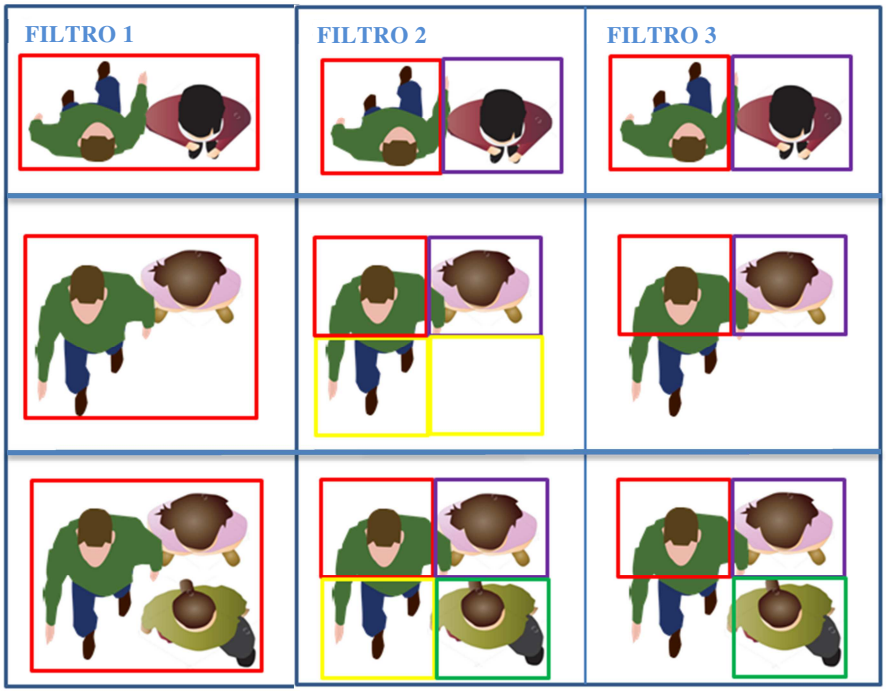


Figura 31 - Esempi di funzionamento del metodo ABSM_DETECTION

Per caratterizzare l'accuratezza del metodo proposto, si effettua un confronto con altri due metodi di rilevamento delle persone presenti in letteratura, che utilizzano due metodologie di *detection* completamente diverse tra di loro, anche in confronto al metodo precedentemente descritto. In particolare il primo metodo appartiene alla categoria degli *unsupervised* e rileva gli individui ricercando la testa essendo la parte del corpo più vicina al sensore; mentre il secondo metodo appartiene alla categoria *supervised* e quindi richiede l'addestramento di un classificatore.

Nel primo metodo (chiamato WATERFILLING), Zhang et al. [66], gli autori partono dalla considerazione che la testa sia sempre la parte della persona più vicina alla telecamera (essendo questa in posizione zenitale). Di conseguenza, il problema di rilevare le teste e quindi le persone nella scena si converte nella ricerca delle regioni che

comprendono i minimi locali. L'algoritmo di WATERFILLING si basa sull'omonimo processo ideato per progettare e realizzare strategie di equalizzazione sui canali nell'ambito dei sistemi di comunicazione. Si pensi al comportamento naturale dell'acqua piovana che tende a scorrere lungo le regioni convesse a causa della forza di gravità e si raccoglie e livella all'interno delle regioni cave. In tal caso è possibile valutare la quantità di acqua accumulata per ricavare informazioni. Considerando una *depht map*, generata da un sensore di profondità in posizione zenitale, come una funzione rappresentativa dell'informazione di profondità del *pixel* e ricordando che gli oggetti più vicini al sensore hanno un valore di profondità minore, individuare la testa delle persone nella scena corrisponde a trovare le regioni di minimo locale della funzione. Simulando le gocce di pioggia con caduta uniforme si individuano le regioni cave con la presenza di acqua. La *depth map* può essere vista come una serie di rilievi e avvallamenti, una goccia di pioggia su un rilievo scorrerà direttamente fino alla zona cava che a poco a poco raccoglierà una certa quantità d'acqua. Quanto descritto si può riassumere nelle fasi illustrate in Figura 32.

Figura 32 - Fasi del metodo WATERFILLING

Nella fase *Water Drop*, si implementa il processo di *water filling*: a partire dall'immagine di profondità (prima illustrazione della Figura 33) il metodo simula la caduta della pioggia che si andrà ad accumulare nei bacini , questi ultimi rappresentano le aree in cui sono presenti i minimi locali (seconda illustrazione della Figura 33). Nella fase di *Water Filter*, una operazione di sogliatura permette di trascurare i *pixel* appartenenti a falsi positivi o parti della spalla di una persona (nella terza illustrazione della Figura 33, a partire dalle quattro regioni individuate, la regione *D* viene completamente filtrata

poiché al di sotto della soglia definita, mentre le altre vengono ridimensionate). Nelle fasi di *Blob Detection* e *Blob Filter*, il metodo ricerca le componenti connesse all'interno dell'immagine e analizza l'area di ogni regione individuata al fine di filtrare quelle con caratteristiche non appartenenti ad una persona (area inferiore ad un dato valore). Alla fine, il metodo applica un algoritmo di *tracking* per il conteggio degli attraversamenti. Per la sperimentazione del metodo, gli autori hanno condiviso il codice implementato in linguaggio *C++* utilizzando la libreria *OpenCV*.

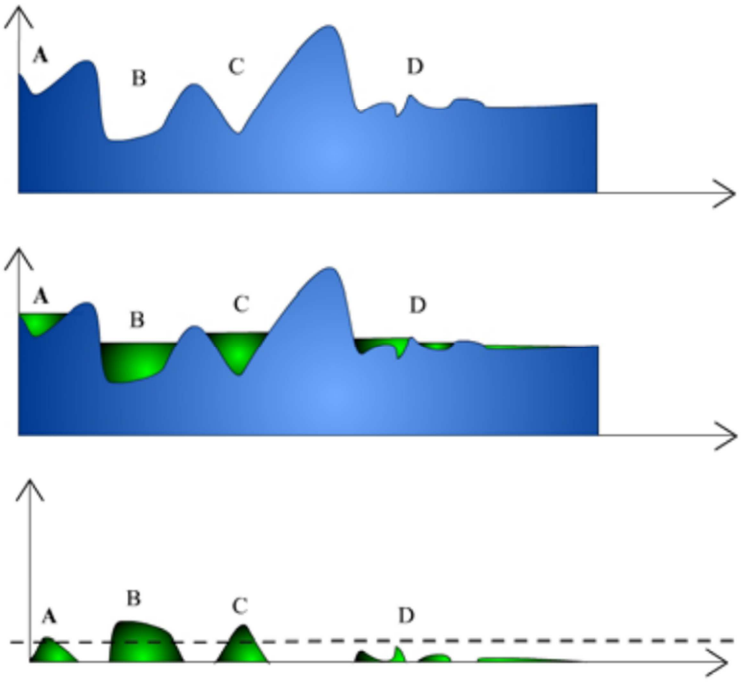


Figura 33 - Processo di water filling

Il secondo metodo di confronto è quello proposto da Vera et al. [67] basato su un'applicazione del *Dalal and Triggs' method of the histograms of oriented gradients (HOG) [80]* e sul classificatore *SVM* [79]. L'algoritmo prevede due fasi: la fase di addestramento (*learning*) basata su un set predefinito di esempi negativi e positivi per addestrare

un classificatore in grado di decidere se ogni regione dell'immagine di dimensioni fissate (*detection window*) è "persona" o "non-persona", e la fase di riconoscimento (*detection*) che usa il classificatore per effettuare la scansione dell'immagine riportando posizione e probabilità della presenza di una o più persone. Ogni *detection window* è caratterizzata da un descrittore HOG (Figura 34).

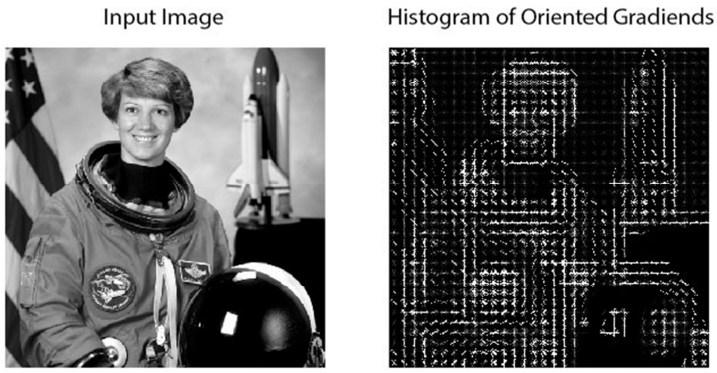


Figura 34 - Esempio di istogramma a gradienti orientato

Nel metodo di *Dalal* le *features* di ogni immagine si basano sull'*histograms of oriented gradients* e sono estratte a livello di cella (ogni cella fa parte di un blocco, e ogni immagine è divisa in più blocchi). L'orientamento del gradiente è raggruppato in *nine-bin histograms*. La Figura 35 illustra le fasi del metodo.

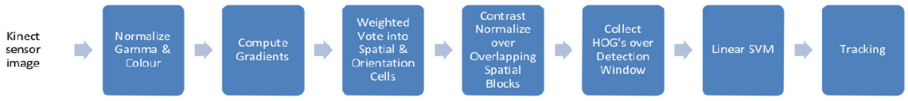


Figura 35 - Fasi del metodo HOG-SVM

L'immagine in ingresso viene normalizzata attraverso una correzione di gamma e, mediante l'applicazione di una maschera, si ricavano i gradienti per la costruzione dell'istogramma. L'immagine ottenuta è divisa in celle rettangolari o circolari per ciascuna delle quali si genera un vettore orientato rappresentativo del gradiente con un calcolo

pesato sui voti dei singoli gradienti. L'istogramma viene normalizzato con un fattore di normalizzazione calcolato per ogni blocco di *2x2* celle, che secondo *Dalal e Triggs* è il valore ottimale, per migliorare l'invarianza a diverse condizioni di luminosità, contrasto e ombre.

Successivamente si scansiona l'immagine per ottenere la *detection window* concatenando gli istogrammi che la compongono e creando un *feature vector* che viene classificato mediante il classificatore SVM lineare. Il classificatore SVM è un algoritmo di classificazione che genera un iperpiano di separazione tale da massimizzare la distanza dei punti più vicini delle classi di dati di *training*. Ogni *feature vector* è visto come un punto in uno spazio *p-dimensionale*. Se i dati di *training* non risultassero separabili linearmente impedendo la generazione dell'iperpiano, allora i campioni verrebbero mappati dal classificatore in uno spazio di dimensione più elevata in cui linearmente separabili. Matematicamente, se lo spazio dei campioni originali ha dimensione n, e lo spazio di dimensione superiore ha dimensione m con $m > n$, allora si deve trovare una funzione:

$$\Phi : R^n \rightarrow R^m$$

tale che nel nuovo spazio i campioni siano linearmente separabili. Alla fine della catena di elaborazione poiché si possono ottenere, attorno ad una stessa persona, più *detection windows* con riscontro positivo, di applica un metodo come *Mean-Shift* o *Random Forest* per far convergere la risposta un'unica *detection*. Dopo la fase di classificazione tramite SVM, il metodo utilizza un algoritmo di *tracking* per il conteggio degli attraversamenti. Nel seguito si indica il presente metodo con il nome di HOG-SVM. Il metodo è implementato in linguaggio *C++* utilizzando la libreria *OpenCV 2.4.13*.

4.3 Sistema multi-esperto

Per la classificazione degli oggetti sono stati proposti in letteratura diversi *feature descriptor* (SIFT [109], MSER [110], HOG [80] e SURF [111]) che lavorano efficacemente su particolari classi di oggetti o in particolari scenari, ma è sperimentalmente dimostrato che la costruzione di classificatore perfetto per ogni compito è ad oggi impossibile. Pertanto diviene indispensabile l'utilizzo di più classificatori (chiamati esperti), effettuando una combinazione dei risultati, al fine di ottenere un miglioramento delle prestazioni. Quanto detto rappresenta l'obiettivo di un sistema multi-classificatore (*Multiple Classifier System*, MCS) o multi-esperto (*Multiple Expert System*, MES). Difatti, agire su un metodo già sperimentato per provare ad accrescerne le prestazioni potrebbe richiedere un enorme sforzo producendo scarsi miglioramenti dei risultati. Nel compiere una combinazione è opportuno non perdere i punti di forza di ciascun esperto considerando quanto sono indipendenti e diversi (in che modo essi prendono una decisione) e quali le regole di combinazione impiegare. In particolare, Dietterich [106] indica che si potrebbero raggiungere risultati migliori dei singoli classificatori se il tasso di riconoscimento di ogni singolo esperto risultasse maggiore dello *0.5*, con errori effettuati dai singoli classificatori non correlati tra loro.

In letteratura sono presenti diversi *survey* e *review* sui MES [112][113][114][115][116][117] in cui si evidenzia come tali sistemi siano applicati con successo in diversi ambiti applicativi, tra i quali: *handwrite recognition* [118][138][119][120], *speaker recognition* [121][122][123][124], *medical diagnosis* [125][126][128][129], *fingerprint recognition* [130][131], *automatic image annotation* [132], *multimodal personal identification* [133], *face detection* [134], *movie segmentation* [135][136] e *network intrusion detection* [137]. Inoltre si affronta il problema di valutare l'affidabilità della risposta fornita dal MES. Difatti, quando gli esperti non sono in accordo sulla risposta potrebbe essere difficile prendere la giusta decisione di classificazione

e quindi di norma la risposta del MES non può essere considerata sempre affidabile. Situazioni critiche si verificano quando, per un dato ingresso, due o più classi sono valutate simili o quando nessuna classe può essere considerata sicura. In queste condizioni bisogna considerare se rifiutare un campione (cioè non assegnarlo ad una classe) anziché incorrere in un eventuale errore di classificazione. In letteratura sono presenti diversi lavori che affrontano il predetto problema, come in Foggia et al. [139] e in De Stefano et al. [140].

Come detto, i sistemi multi-esperto combinano diversi classificatori per migliorare le prestazioni complessive. Ci sono diversi approcci in letteratura per combinare i classificatori. Le diverse strategie si possono raggruppare in due macro categorie: *fixed rule fusion* and *trained rule fusion* [113][133].

Per la categoria *fixed rule*, basata su un *Bayesian decision model* [118], le regole di combinazione non richiedono un ulteriore addestramento e quindi utilizzano direttamente i risultati dei classificatori. In questa categoria troviamo, ad esempio, le seguenti regole: *MAX, MIN, SUM, PROD, AVM, Average vote, Weighted average vote, Majority voting* [118][113][141][142][143][144], dove vote e sum sono le più diffuse. Le *fixed rule* sono efficaci quando i classificatori presentano un'accuratezza simile e gli output non sono correlati [145].

Per quanto concerne la categoria *trained rule fusion*, questa considera la stessa fusione come un problema di classificazione, e quindi necessita di una fase di addestramento. Esempi di questa categoria sono *Dempster-Shafer Theory* [146], *Decision Template* [147] e *Behavior Knowledge Space* [148], ecc.

Quando bisogna scegliere tra la categoria *fixed rule* e quella *trained rule* è importante tenere in considerazione le principali caratteristiche di ogni approccio. Per un'analisi sulla predetta questione ed in

generale sulla letteratura riguardante la combinazione dei classificatori si raccomanda il lavoro di Kuncheva [158].

In generale, si individuano tre tipologie di funzioni di combinazione, conseguenti del tipo di *output* fornito dai singoli classificatori:

1. Quando i classificatori assegnano una *label* per ogni *input* il multi-esperto dovrà assegnare una *label* definitiva;
2. Nel caso di classificatori *ranking*, ognuno produrrà un vettore di interi in cui ogni elemento corrisponde ad una delle N classi. I *rank* dei diversi classificatori sono unificati per ottenere una "classifica finale" utile per la decisione. *L'output* del multi-esperto sarà un vettore di interi risultante.
3. Quando i classificatori producono, per ogni input, un vettore di numeri reali (di dimensioni pari al numero di classi) in cui ogni elemento corrisponde a quanto il classificatore ritiene che l'*input* appartenga alla classe in esame (*measurement level*).

Lu [149] divide i sistemi multi-esperto in tre categorie (come illustrato in Figura 36):

- ✓ Seriale: *l'output* di un classificatore è utilizzato come *input* per il classificatore successive;
- ✓ Parallelo: i risultati di tutti i classificatori sono analizzati complessivamente per ottenere un unico risultato. La scelta della metodologia di combinazione è determinante per il risultato finale.
- ✓ Gerarchico: combina le precedenti categorie per ottenere un unico risultato.

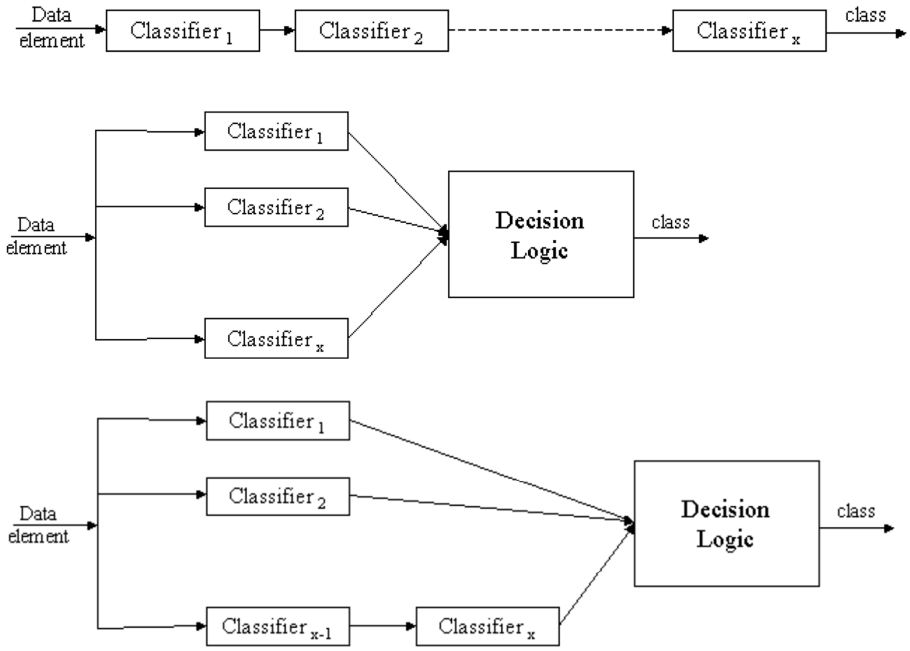


Figura 36 - Categorie di sistemi multi-esperto

Canuto [150] divide le regole per la combinazione dei classificatori nelle seguenti categorie:

- ✓ *Linear combination methods*: utilizzano funzioni lineari come *sum* e *prod* per la combinazione dei risultati dei classificatori;
- ✓ *Non-linear combination methods*: utilizzano classificatori basati sul *rank* come *majority voting*;
- ✓ *Statistical*: utilizzano metodi come *Dempster-Shafer technique* e *Bayesian combination*;
- ✓ *Computationally intelligent (CI)*: utilizzano reti neurali e algoritmi genetici.

Inoltre indica altri tipi di categorie in accordo alla presenza o assenza di un grado di *confidence* (*non-confidence*) associato ai singoli classificatori, cioè un peso o un'importanza associato all'output di

ogni classificatore prima di combinarlo nel sistema multi-esperto. I metodi che non si basano sulla *confidence* considerano i classificatori allo stesso livello nel momento in cui si effettua la combinazione.

Di seguito si presentano le regole di combinazione più utilizzate in letteratura:

- ✓ *MAX –maximum*: viene scelto l'output del classificatore con il valore di *confidence* più alto;
- ✓ *MIN – minimum*: viene scelto l'output del classificatore con il valore di *confidence* più basso;
- ✓ *SUM – summation*: utilizza un approccio *non-confidence-based* in cui si sommano gli output di ogni esperto. Una variante associa un peso ad ogni classificatore e applica una sommatoria pesata (*confidence-based*);
- ✓ *PROD – product*: effettua un prodotto tra gli output dei singoli classificatori senza assegnare alcun peso agli esperti (*non-confidence-based*) oppure assegnando un determinato peso (*confidence-based*);
- ✓ *AVG – average/median*: divide la sommatoria degli *output* per il numero dei classificatori considerati, anche in questo caso la regola può essere o meno *confidence-based* e lavora meglio nel caso in cui gli output dei classificatori sono *measurement level*;
- ✓ *AVERAGE VOTE*: ogni classificatore restituisce un array di valori tra 0 e 1 che indica il livello di compatibilità dell'ingresso con ognuna delle classi (*measurement level*), la regola in esame aggiunge voti per ogni classe e seleziona quella con il voto maggiore;
- ✓ *WEIGHTED AVERAGE VOTE*: simile ad *average vote* ma ad ogni classe i assegna un peso;
- ✓ *MAJORITY VOTING*: in questo caso si seleziona la classe più popolare, ad esempio se abbiamo tre classificatori, che presentano errori non correlati, i quali devono decidere su un

nuovo input *x*, se $C_1(x)$ fornisce una classificazione errata il risultato di $C_2(x)$ e $C_3(x)$ può essere corretto, in tal caso la regola *majority vote* fornirà la classificazione di $C_2(x)$ e $C_3(x)$ ottenendo un risultato corretto. Questa regola risulta efficace quando i classificatori restituiscono voti binari (la classe c'è oppure no) e risulta tra quelle più utilizzate in letteratura;

- ✓ *BAYESIAN COMBINATION*: utilizzate per combinare linearmente le predizioni probabilistiche pesando la probabilità a posteriori [151].

Alcuni lavori utilizzano *Oracle* per provare la bontà del livello di accuratezza raggiunto dal multi-esperto realizzato [152]. *Oracle* è realizzato in modo tale che se almeno un classificatore fornisce la risposta corretta allora in uscita si avrà la risposta corretta.

Di seguito si riportano alcuni lavori recenti presenti in letteratura sull'utilizzo dei MES su tematiche attinenti l'*image analysis*: Hou et al. [154] utilizzano una combinazione di classificatori per accrescere l'accuratezza degli algoritmi di *object classification*. In particolare, effettuano una sperimentazione estesa (su quattro *dataset*) per verificare l'*average classifier combination* al variare di *features, kernel, soft label* e dei classificatori utilizzati, quindi in riferimento agli approcci *feature combination* (combina diverse *feature* prima della classificazione) e *classifier fusion* (combinare i risultati dei singoli classificatori per avere una decisione finale) e quindi rilevare quale delle combinazioni fornisce i risultati migliori. Gli autori partono dal fatto che Gehler et al. [160] per l'ambito *feature combination)* e Altmann et al.[161] e Heerden et al. [162] per l'ambito *classifier fusion),* affermano che le principali *features* e metodi di *classifier fusion* presenti in letteratura lavorano mediamente bene. Pertanto gli autori hanno analizzato, su un *dataset* ampio, la *average combination*, che mostra prestazioni migliori rispetto ad altre strategie di fusione (come *major voting* [161] e *product combination* [160]). I risultati della sperimentazione confermano un aumento delle

prestazioni in diverse combinazioni e non registrano un significativo aumento delle prestazioni nel caso di utilizzo di tutti gli elementi presi in considerazione. Nguyen et al. [155] propongono una combinazione di classificatori basata sul *Bayesian inference framework* denominata *VIG*, dove gli autori propongono di *variational inference* (VI) per stimare i parametri di una *multivariate Gaussian density distribution.* Per la valutazione delle prestazioni, gli autori utilizzano due *dataset* [163][164], il primo con immagini di varia natura (auto, bilancia, …) mentre il secondo in ambito medico con circa *15,363* immagini con *193* categorie. Gli autori hanno confrontato il metodo proposto con altri presenti in letteratura in riferimento all'accuratezza e all'onere computazionale. Sharma et al. [156] propongono un miglioramento delle tecniche di *rank level fusion* per sistemi *multibiometric* utilizzando combinazioni seriali e parallele. La sperimentazione avviene su tre dataset: *NIST BSSR1* [165], *Face Recognition* [166] e *Grand Challenge V2.0 LG4000 iris images* [167]. Grover et al. [153] presentano un *multimodal biometric authentication* basato su FKP (*finger-knuckle-print*). Per migliorare l'accuratezza dei singoli FKP gli autori hanno analizzato l'utilizzo della *entropy function* per la fusione *rank level* (utilizzando per la prima volta il *Choquet integral*). Foggia et al. [157] propongono un metodo per rilevare la presenza di fuoco nella scena. Il sistema multi-esperto realizzato combina i risultati di diversi classificatori basati sull'analisi della forma, del colore e del movimento della fiamma. A partire dai risultati dei tre esperti, il sistema utilizza la regola di combinazione *weighted voting* poiché gli autori partono dall'idea che i tre classificatori hanno diverse priorità di voto in accordo al proprio *learning space*. La sperimentazione, effettuata su un *dataset* acquisito dagli autori (in ambiente interno ed esterno con *62690* immagini) e altri presenti in letteratura (per un totale di *31250* immagini), conferma un aumento dell'accuratezza con una riduzione significativa dei falsi positivi. Dash et al. [173] propongono una strategia di *classifier fusion* per la classificazione di *multi-class texture*, combinando i risultati di due classificatori. Il

framework risultante è chiamato *Classification Confidence-based Multiple Classifier Approach* (CCMCA). I classificatori utilizzati sono *Neural Network* e *Naive Bayes*. Durante la sperimentazione, svolta su un *texture database* [168][169], gli autori hanno confrontando i risultati non solo rispetto ai singoli classificatori utilizzati ma anche rispetto alle regole più comuni in letteratura (*Majority Voting, Maximum, Minimum, Product* e *Mean*) ottenendo una maggiore accuratezza. De Marsico et al. [159] descrivono un'architettura biometrica per implementare un'applicazione di *face recognition*, in accordo ad etnia, genere ed età, per l'identificazione degli individui. Gli autori utilizzano un approccio a più esperti, con classificatori formati su diverse classi demografiche ed eseguiti in parallelo. In seguito un modulo supervisore combina i risultati restituiti dai singoli esperti per ottenere un unico risultato. Come segnalato dagli autori, questa strategia può raggiungere prestazioni leggermente inferiori, a causa del fatto che le risposte di diversi esperti possono introdurre ulteriori errori nel risultato finale. Il vantaggio risiede nell'assenza di una fase di *preprocessing* manuale e automatica per il riconoscimento.

A partire dall'analisi della letteratura si realizza un sistema multi-esperto con l'obiettivo di verificare il miglioramento delle prestazioni in seguito alla combinazione delle risposte appartenenti ai tre metodi di rilevamento delle persone descritti nel paragrafo 4.2 (definiti esperti). Per quanto si conosce, non sono presenti in letteratura applicazioni del MES per il problema in esame.

Come prima scelta progettuale, si implementa un MES di tipo "parallelo" in cui i risultati di tutti i classificatori sono analizzati complessivamente per ottenere un unico risultato (si veda Figura 37). Successivamente si passa alla scelta delle regole di combinazione da utilizzare. Tra le due categorie viste in precedenza, sono considerate le regole che appartengono alla categoria *fixed rule,* le quali prendono

una decisione avvalendosi direttamente dei risultati dei tre esperti. Inoltre, la scelta di quali regole utilizzare è dipesa dagli approcci utilizzati dai singoli esperti. Come visto nel paragrafo 4.2, due dei tre classificatori sono *unsupervised* e forniscono come output la sola informazione "persona/non-persona" (*abstract form*). Ne consegue che, come primo approccio al sistema multi-esperto, le regole di combinazione utilizzate sono le seguenti:

- ✓ Regola a MAGGIORANZA: se si combinano le risposte di *n* esperti e di questi almeno $\frac{n+1}{2}$ sono concordi, allora la loro soluzione è considerata come corretta;
- ✓ Regola del MASSIMO: la risposta corrisponde, se esiste, solo a quella data da tutti gli *n* esperti senza esclusione. Questa regola è molto restrittiva, infatti se un solo esperto è in disaccordo con tutti gli altri non si produce alcun risultato (corretto o meno);
- ✓ Regola del MINIMO: la risposta corrisponde a quella data da ciascun esperto anche se tra loro non c'è alcun accordo, quindi si considerano anche le risposte singole di ogni esperto. Questa regola è largamente concessiva e non riesce ad eliminare gli errori che un singolo esperto potrebbe commettere.

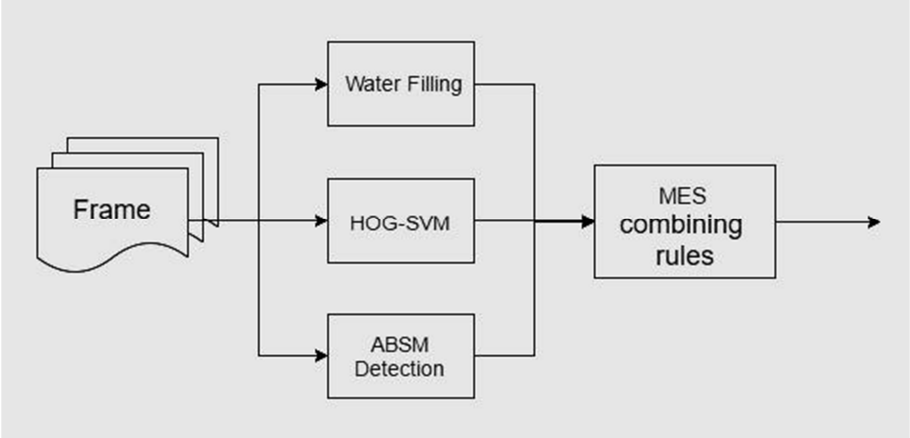


Figura 37 - Sistema multi-esperto realizzato

Sebbene esistano regole anche più complesse, quelle selezionate si ritengono adatte allo scopo da perseguire, ovvero stabilire se con un sistema multi-esperto si riesca ad ottenere un incremento delle prestazioni. Come successivo sviluppo del presente lavoro si potrà introdurre una misura di affidabilità associata a ciascun esperto o alla risposta di ciascun esperto (utilizzando diversi classificatori) e applicando ulteriori regole di combinazione tra quelle analizzate in precedenza. Per di più, si potrà considerare l'applicazione di un multi-esperto seriale o gerarchico.

Il MES realizzato lavora sui risultati dei singoli esperti (WATERFILLING, HOG-SVM e ABSM_DETECTION) ottenuti per ogni *frame* secondo lo schema riportato in Figura 38.

Figura 38 - Diagramma di flusso del sistema multi-esperto

L'algoritmo riceve le persone rilevate da ciascun esperto per il *frame* corrente come vettori di *bounding box* (che possono far riferimento anche a falsi rilevamenti). Nella fase successiva, l'algoritmo determina tutte le combinazioni mescolando i risultati dei singoli esperti. Le combinazioni si realizzano partendo dal numero massimo di elementi che si possono raggruppare (pari al numero di esperti), tale numero si decrementa fino a considerare combinazioni di singoli elementi appartenenti ad ogni esperto. Poiché nel caso in esame si hanno tre esperti in ingresso, si passa da combinazioni di tre elementi, fino a combinazioni di un solo elemento. A partire dalla combinazione determinate, l'algoritmo seleziona quelle la cui percentuale di sovrapposizione è maggiore di una certa soglia β:

$$\frac{Area\left(\bigcap_{i=0}^{k-1} bounding\ box\right)}{Area\left(\bigcup_{i=0}^{k-1} bounding\ box\right)} > \beta$$

dove k è il numero di *bounding box* della combinazione e β è un parametro dell'algoritmo del MES. Quando viene selezionata una combinazione che verifica la disequazione, tutte le altre combinazioni che contengono almeno un elemento in comune con essa vengono automaticamente scartate (in modo da non considerare più di una volta la stesse *detection*). Per ogni combinazione selezionata, si determina il *bounding box* risultante, dato dalla media delle coordinate dei *box* interessati (media su posizione, altezza e larghezza). Nell'ultima fase

dell'algoritmo vengono generati i vettori di output in base alle regole di combinazione associate al sistema. Di conseguenza, la risposta del MES consiste in uno o più vettori di *bounding box*, uno per ogni regola utilizzata.

L'algoritmo per il sistema multi-esperto è implementato in linguaggio *C++* e presenta i seguenti parametri di ingresso:

- ✓ le regole di decisione da attivare (MINIMO, MASSIMO, MAGGIORANZA);
- ✓ il numero degli esperti;
- ✓ il vettore dei *bounding box* per ogni esperto (*output* di ogni metodo);
- ✓ la soglia β sulla sovrapposizione tra i *bounding box* della combinazione;

4.4 Descrizione del Dataset

La sperimentazione dei metodi di rilevamento delle persone si effettua su un *dataset* appositamente realizzato e descritto nel *paragrafo 3.3*.

In particolare si utilizza il $Dataset_C$ sul quale si realizzata la *ground truth* per le applicazioni di rilevamento delle persone. In questo caso, un algoritmo scritto in *C++* con l'utilizzo della libreria OpenCV (*Open Source Computer Vision Library*) versione *2.4.13* (per dettagli sulla libreria si rimanda al paragrafo 5.4), consente di disegnare, per ogni *frame*, il più piccolo quadrato che contiene la testa di una persona e il più piccolo quadrato che include le spalle e la testa di una persona. La doppia rappresentazione è motivata dal fatto che, come riportato nel seguito, gli algoritmi di *people detection* con telecamera in posizione zenitale presenti in letteratura operano rilevando la testa o l'area comprendente le spalle e la testa di un individuo. In Figura 39 si riporta un esempio di *ground truth* per le applicazione di rilevamento delle persone all'interno della scena.

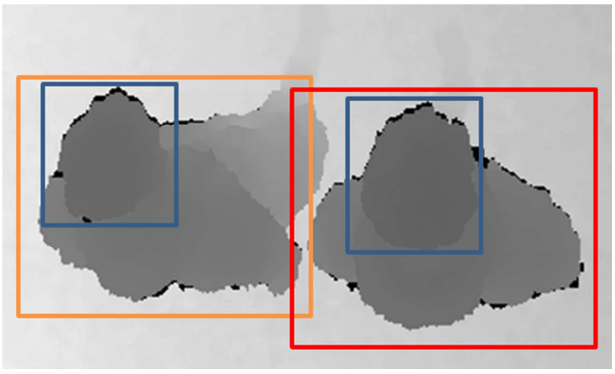


Figura 39 - Esempio di ground truth per il rilevamento delle persone

Al fine di valutare correttamente i metodi di rilevamento delle persone, sono considerati ininfluenti le *detection* in cui le teste e le spalle delle persone compaiono parzialmente nella scena. A partire da ciò e considerando che il sensore di profondità utilizzato introduce un errore corrispondente alla presenza di *null-pixel* in un intorno di circa *25 pixel* (come evidenziato in Figura 40), si considerano solo le *detection* che avvengono al di fuori di un intorno pari a *50 pixel*. Detto ciò, il numero massimo di persone presenti contemporaneamente nella scena risulta essere pari a quattro. In Tabella 15 si riporta il numero di *frame* di cui si compone il $Dataset_C$ al variare del numero di persone presenti nella scena e dello scenario di riferimento.

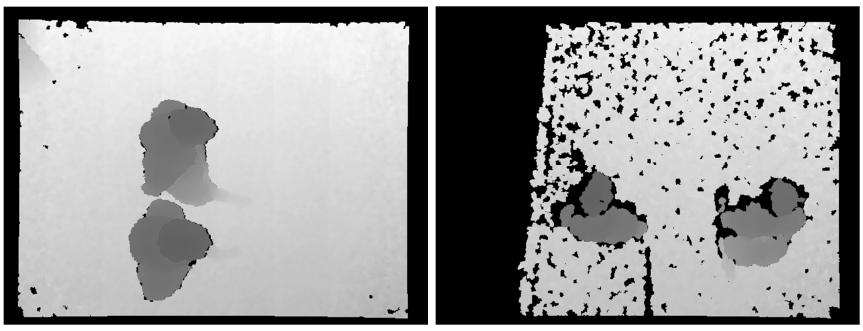


Figura 40 - Esempi di immagini ottenute dal sensore di profondità

Persone	*indoor*	*outdoor*	Totale
0	23,626	26,010	49,636
1	7,788	8,753	16,541
2	1,652	2,501	4,153
3	438	427	865
4	95	108	203

Tabella 15 - Numero di *frame* al variare del numero di persone e dello scenario per il $Dataset_C$

Al completamento della *ground truth* di ogni sequenza, si genera un file, in formato *.xml*, in cui sono presenti le coordinate dei rettangoli riferiti ad ogni individuo presente nella scena. Di seguito si riporta un esempio di *ground truth* per una sequenza *indoor*:

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <file name="D_I_G_1_head_shoulders" num_frames="4340" fps="30"
frame_width="640" frame_height="480"> </file>
    <objects>
        <person frame="99" id="1" head_rect="[22, 156]-[82, 216]"
head_shoulders_rect="[22, 151]-[82, 260]"/>
        <person frame="100" id="2" head_rect="[37, 169]-[96, 228]"
head_shoulders_rect="[37, 160]-[96, 274]"/>
        <person frame="101" id="3" head_rect="[51, 162]-[111, 222]"
head_shoulders_rect="[51, 158]-[111, 268]"/>
        <person frame="102" id="4" head_rect="[65, 167]-[120, 222]"
head_shoulders_rect="[65, 162]-[120, 274]"/>
        <person frame="103" id="5" head_rect="[81, 166]-[131, 216]"
head_shoulders_rect="[81, 164]-[131, 271]"/>
        <person frame="104" id="6" head_rect="[93, 160]-[150, 217]"
head_shoulders_rect="[93, 150]-[150, 269]"/>
        <person frame="104" id="7" head_rect="[24, 339]-[71, 386]"
head_shoulders_rect="[24, 300]-[71, 438]"/>
        <person frame="105" id="8" head_rect="[104, 161]-[162, 219]"
head_shoulders_rect="[104, 150]-[162, 260]"/>
        <person frame="105" id="9" head_rect="[33, 338]-[86, 391]"
head_shoulders_rect="[33, 303]-[87, 443]"/>
    </objects>
</data>
```

Il file presenta nella parte iniziale le informazioni relative alla sequenza (nome del file, numero di *frame*, *frame* rate, risoluzione). Per ogni *detection*, si aggiunge in *objects* un oggetto *person* con i seguenti attributi:

- ✓ *frame* corrente (*frame*);

- ✓ coordinate del rettangolo riferito alla testa della persona (*head_rect*): $[x_1, y_1]$-$[x_2, y_2]$ dove (x_1, y_1) corrisponde al punto in alto a sinistra del rettangolo, mentre (x_2, y_2) al punto in basso a destra del rettangolo;
- ✓ coordinate del rettangolo riferito all'area contenete la testa e le spalle della persona (*head_shoulders_rect*): $[x'_1, y'_1]$-$[x'_2, y'_2]$ dove (x'_1, y'_1) corrisponde al punto in alto a sinistra del rettangolo, mentre (x'_2, y'_2) al punto in basso a destra del rettangolo;

Un'analisi su ulteriori *dataset* presenti in letteratura è riportata nel Paragrafo 3.3.

4.5 Indici Prestazionali e Risultati Sperimentali

Gli indici prestazionali utilizzati per la valutazione delle prestazioni dei metodi sul $Dataset_C$ sono *precision*, *recall* e *f-index*.

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

$$f-index = \frac{2 \cdot Prec \cdot Recall}{Prec+Recall}$$

dove *TP* corrisponde al numero di *true positive* (rilevamenti che sono correttamente individuati dal metodo), *FP* il numero di *false positive* (rilevamenti erroneamente indicati dal metodo) e *FN* il numero di *false negative* (rilevamenti non individuati dal metodo). La *Precision* rappresenta la percentuale di *detection* corrette rispetto a tutte quelle effettuate, ed è quindi la percentuale con cui l'algoritmo ha eseguito *detection* corrispondenti a verità. La *Recall* rappresenta la percentuale di persone che l'algoritmo ha rilevato rispetto al numero di persone che erano effettivamente presenti sulla scena, quindi valuta la capacità dell'algoritmo di rilevare una persona. Il valore di *f-index* rappresenta

la media armonica dei due valori precedenti. La Figura 41 illustra alcuni esempi di *true positive, false negative* e *false positive.*

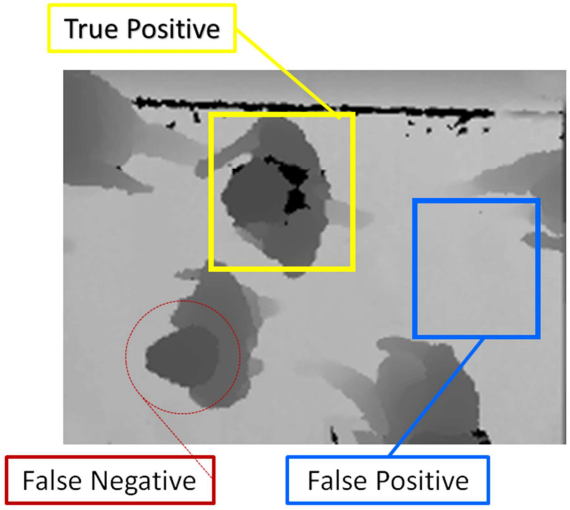


Figura 41 - Esempi di TP, FN e FP per il rilevamento della persona

In particolare, una persona si considera individuata correttamente se la seguente condizione viene rispettata:

$$\frac{area(B_d \cap B_g)}{area(B_d \cup B_g)} > 0.5$$

dove B_d e B_g sono rispettivamente i *bounding box* del metodo e della *ground truth.*

A partire dal $Dataset_C$ si ottiene il *training set* e il *test set*. Il primo si utilizza per selezionare i valori dei parametri che forniscono le prestazioni migliori, mentre il secondo per valutare le prestazioni complessive dei metodi.

Dal $Dataset_C$ si estraggono *102 frame* (non consecutive con la presenza di una o più persone) per realizzare il *training set* sul quale si effettuano le operazioni riportate in Figura 42, dove il primo percorso

(1) fa riferimento al metodo HOG-SVM, mentre il secondo (2) agli altri due metodi.

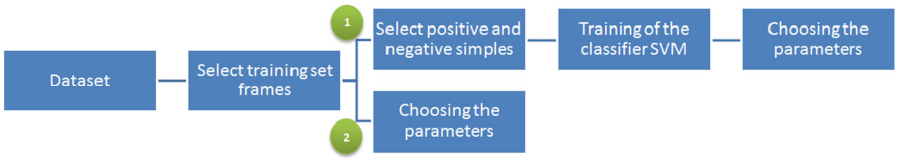


Figura 42 - Diagramma di flusso dei metodi di rilevamento con l'utilizzo del *training set*

Il metodo HOG-SVM utilizza il *training set* per acquisire i campioni negativi e positivi utili all'addestramento del classificatore. In particolare, per l'acquisizione dei campioni si utilizza un algoritmo, appositamente realizzato, che consente all'utente di selezionare a video i campioni (disegnare le coordinate del rettangolo che include la persona) e realizzare un processo semi-automatico di classificazione. Ogni campione selezionato (a risoluzione *128*128*) viene associato ad una fascia di attraversamento in base all'angolo con cui la persona attraversa lo scenario (angolo rispetto al *reference coordinate system*, in cui l'asse *X* corrisponde alle coordinate *0°-180°*) al fine di permettere l'addestramento di più classificatori (ognuno specializzato al rilevamento di una predeterminata fascia). Le fasce di attraversamento sono:

- *Fascia 1: [331°, 30°] – [151°, 210°]*
- *Fascia 2: [241°, 300°] – [61°, 120°]*
- *Fascia 3: [31°, 60°] – [211°, 240°]*
- *Fascia 4: [121°, 150°] – [301°, 330°]*

Per ogni campione selezionato si effettuano le seguenti operazioni:

1. assegnazione della fascia di riferimento;
2. rotazione con passo pari a *10°* al fine di ricoprire tutte le fasce predefinite;

3. *flip* orizzontale e rotazione con passo pari a *10°* per ricoprire tutte le fasce predefinite.

In questo modo si ha un incremento delle dimensioni del *training set* e un addestramento dei classificatori utilizzando gli stessi campioni; ad esempio considerando un *frame* in cui è presente un'unica persona (unico campione) e attraversamenti lungo la *Fascia 1*, si hanno *11* campioni ruotati, *1* campione a cui si applica l'operazione di *flip* orizzontale e *11* campioni ruotati a partire da quello a cui è applicato il *flip*, per un totale di *24* campioni positivi per ogni *frame* in cui è presente un'unica persona. Detto ciò e considerando i *102 frame* selezionati per il training set, si ottengono *4176* campioni positivi per ogni fascia di attraversamento.

L'algoritmo, al termine dell'acquisizione dei campioni positivi di un *frame*, genera automaticamente un ugual numero di campioni negativi selezionando casualmente *box* di dimensione *128*128* che possono sovrapporsi ai campioni positivi al più per una percentuale pari a *40%* (valore scelto in accordo a un test effettuato su alcuni *frame* del *training set*). Da sottolineare che il *dataset* contiene passaggi lungo la "Fascia1" per lo scenario *indoor* e passaggi lungo la "Fascia2" per lo scenario *outdoor*. A partire da tale premessa, pur realizzando un *training set* che copra i *360°*, si utilizzano i soli campioni compresi nelle fasce "Fascia1" e "Fascia2" per addestrare rispettivamente due classificatori SVM, anche se, in accordo al processo di acquisizione, entrambi i classificatori sono addestrati con gli stessi campioni. Nel seguito, per "*classificatore1*" si indica il classificatore addestrato per lo scenario *indoor* e per "*classificatore2*" quello per lo scenario *outdoor*.

Al termine della fase di selezione dei campioni positivi e negativi dal *training set*, si procede all'addestramento dei classificatori. Il classificatore viene addestrato fornendo in ingresso il *feature vector* di ogni campione ottenuto utilizzando HOG. In particolare, all'algoritmo

di addestramento viene fornita una matrice con un numero di righe pari al numero dei campioni negativi e positivi riferiti alla fascia di interesse, e un numero di colonne pari alla dimensione del *feature vector* di ogni campione con l'aggiunta di una colonna per indicare se il vettore si riferisce ad un campione positivo (*1*) o negativo (*-1*). L'output di questa fase corrisponde a due file corrispondenti ai due classificatori addestrati, da utilizzare rispettivamente per lo scenario *indoor* e *outdoor*.

Nell'ultima fase, comune ai tre metodi, si utilizza il *training set* per selezionare i valori dei parametri che mostrano le prestazioni migliori. Si procede variando ciascun parametro in un *range* in accordo ai valori riportati dagli autori dei metodi e allo scenario di riferimento. I risultati della sperimentazione sul *training set* forniscono i seguenti risultati:

- il metodo HOG-SVM presenta prestazioni simili con lo stesso *set* di valori al variare dello scenario di riferimento, questo è in linea con quanto atteso, poiché entrambi i classificatori sono addestrati utilizzando gli stessi campioni ruotati;
- il metodo WATERFILLING richiede valori dei parametri diversi al variare dello scenario. Questa differenza è motivata dalla presenza di maggiore rumore in ambiente *outdoor*. Difatti, analizzando la prima e la terza immagine della Figura 43, due *frame* appartenenti al *training set* per ogni scenario, è evidente la presenza di un numero maggiore di *null-pixel* (*pixel* neri) nello scenario *outdoor*, da attribuire agli effetti della luce solare sul sensore utilizzato. Di conseguenza, si procede modificando i parametri per evidenziare maggiormente le teste, al costo di avere nella *foreground mask* anche *pixel* appartenenti a parti del collo e delle spalle della persona e riducendo le soglie nelle fasi di *Water Filter* e *Blob Filter* di un valore pari a circa il *30%*;

- il metodo ABSM_DETECTION richiede valori leggermente diversi al variare dello scenario. Anche in questo caso, la differenza è motivata dalla presenza di maggiore rumore in ambiente *outdoor*. In particolare, il parametro che rappresenta la soglia per stabilire se il box corrisponde a due o più persone è ridotto di circa il 20% mentre gli altri parametri restano invariati.

Figura 43 - Risultati del WATERFILLING sul training set: la prima e la terza colonna mostrano i *frame* di ingresso, mentre la seconda e la quarta i risultati del metodo. Le prime due colonne si riferiscono allo scenario *indoor* mentre le ultime due a quello *outdoor*.

I *frame* che non appartengono al *training set* fanno parte del *test set*, con il quale si effettua una sperimentazione per valutare le prestazioni di ogni metodo al variare dello scenario di riferimento e della densità degli attraversamenti. La Tabella 16 riporta i risultati della sperimentazione sul *test set* con il metodo HOG-SVM al variare dello scenario e della densità di persone presenti nella scena. Le prestazioni raggiunte non variano significativamente al variare dello scenario. In ambiente *indoor*, il metodo presenta un numero elevato di falsi positivi dovuto alle seguenti occorrenze: più rilevamenti associati ad un'unica persona e rilevamenti aggiuntivi tra due o più persone. Queste occorrenze sono più frequenti negli attraversamenti che avvengono nella parte superiore del *frame*, difatti essendo l'installazione della telecamera non perfettamente perpendicolare al piano le persone lungo quella fascia appaiono di lato. I falsi negativi corrispondono a persone non rilevate o parzialmente rilevate (dove la percentuale di sovrapposizione rispetto la *ground truth* è al di sotto della soglia del *50%*). Nello scenario *outdoor* si ha un incremento dei

falsi negativi dovuto principalmente al maggior rumore introdotto dal sensore (nessun rilevamento, un unico rilevamento per ogni coppia di persone, rilevamento parziale con una percentuale di sovrapposizione inferiore alla soglia del *50%*); mentre la riduzione del numero dei falsi positivi è dovuto ad una diminuzione degli attraversamenti di due o più persone particolarmente ravvicinate, in confronto allo scenario *indoor*. In generale, il metodo HOG-SVM presenta buone prestazioni in entrambi gli scenari e al variare del numero di persone presenti nella scena. In Figura 44 si mostrano alcune occorrenze di falsi negativi e falsi positivi in scenario *indoor* (prime due righe) e *outdoor* (ultime due righe).

La Tabella 17 riporta i risultati del metodo WATERFILLING. In tal caso si ha un comportamento diverso a seconda dello scenario di riferimento, con una riduzione del 25% delle prestazioni passando dallo scenario *indoor* a quello *outdoor* (dovuto all'aumento del rumore introdotto dal sensore in tale scenario). Con riferimento allo scenario *indoor*, i falsi negativi e i falsi positivi sono dovuti alla presenza di *null-pixel* nell'area in cui è presente la persona. Difatti, il metodo non rileva correttamente le teste oppure presenta rilevamenti aggiuntivi lungo le spalle delle persone. In ambiente esterno le persone sono maggiormente frammentate con un incremento dei falsi positivi in corrispondenza delle spalle delle persone. In generale, il metodo WATERFILLING presenta buone prestazioni nello scenario *indoor* e al variare del numero di persone presenti nella scena, mentre le prestazioni si riducono nello scenario *outdoor*, dovuto al rumore introdotto dal sensore in ambiente esterno. Per quest'ultimo caso, l'utilizzo di un'altra tipologia di sensore, ad esempio la nuova versione del *Microsoft Kinect*, potrebbe accrescere l'accuratezza del metodo. In Figura 45 si mostrano alcune occorrenze di falsi negativi e falsi positivi in scenario *indoor* (prime due righe) e *outdoor* (ultime due righe).

La Tabella 18 riporta i risultati della sperimentazione sul *test set* con il metodo ABSM_DETECTION al variare dello scenario e della densità di persone presenti nella scena. Le prestazioni raggiunte non variano significativamente al variare dello scenario. Nello scenario *indoor* si hanno le seguenti occorrenze di falsi positivi e falsi negativi: *box* che presentano un numero di *pixel* di *foreground* inferiore alla soglia pur indicando correttamente una persona, *box* che presentano una percentuale di sovrapposizione con la *ground truth* inferiore alla soglia del *50%* (quando un individuo presenta oggetti aggiuntivi come ad esempio uno zaino, oppure quando il metodo rileva anche parti inferiori della persona, in particolare per attraversamenti nella parte alta del *frame* dove le persone appaiono non perpendicolari alla telecamera). In altri casi il metodo non rileva correttamente la presenza di più individui nella scena poiché il *box* risultante ha una dimensione inferiore alla soglia *minSizePersons* oppure quando il frazionamento del *box* porta ad una soluzione che non include correttamente una persona. Nello scenario *outdoor* si hanno prestazioni migliori in presenza di gruppi di individui, poiché si ha una riduzione degli attraversamenti particolarmente ravvicinati rispetto allo scenario *indoor*. Il maggior rumore presente in questo scenario porta a casi in cui il *box* comprende solo parti di una persona, ciò comporta una dimensione inferiore alla soglia con una riduzione delle prestazioni nel caso di passaggi singoli. In generale, il metodo ABSM_DETECTION presenta buone prestazioni in entrambi gli scenari e al variare del numero di persone presenti nella scena. In Figura 46 si mostrano alcune occorrenze di falsi negativi e falsi positivi in scenario *indoor* (prime due righe) e *outdoor* (ultime due righe).

Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
indoor	1	7,788	0	199	0.975	1.000	0.987
	2	3,303	2	122	0.964	0.999	0.982
	3	1,313	1	75	0.946	0.999	0.972
	4	381	0	14	0.965	1.000	0.982
outdoor	1	8,728	25	50	0.994	0.997	0.996
	2	4,870	132	63	0.987	0.974	0.980
	3	1,267	15	9	0.993	0.988	0.991
	4	419	14	5	0.988	0.968	0.978

Tabella 16 - Prestazioni del metodo HOG-SVM

Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
indoor	1	7,796	749	280	0.965	0.912	0.938
	2	3,132	496	346	0.901	0.863	0.882
	3	1,306	166	113	0.920	0.887	0.903
	4	403	43	24	0.944	0.904	0.923
outdoor	1	8,425	2,054	6,858	0.551	0.804	0.654
	2	5,012	799	3,646	0.579	0.863	0.693
	3	1,335	199	1,088	0.551	0.870	0.675
	4	462	61	334	0.580	0.883	0.701

Tabella 17 - Prestazioni del metodo WATERFILLING

Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
indoor	1	7,761	27	34	0.996	0.997	0.996
	2	3,205	100	83	0.975	0.970	0.972
	3	1,241	73	69	0.947	0.944	0.946
	4	355	26	21	0.944	0.932	0.938
outdoor	1	8,534	219	132	0.985	0.975	0.980
	2	4,812	190	70	0.986	0.962	0.974
	3	1,219	63	29	0.977	0.951	0.964
	4	411	22	7	0.983	0.949	0.966

Tabella 18 - Prestazioni del metodo ABSM_DETECTION

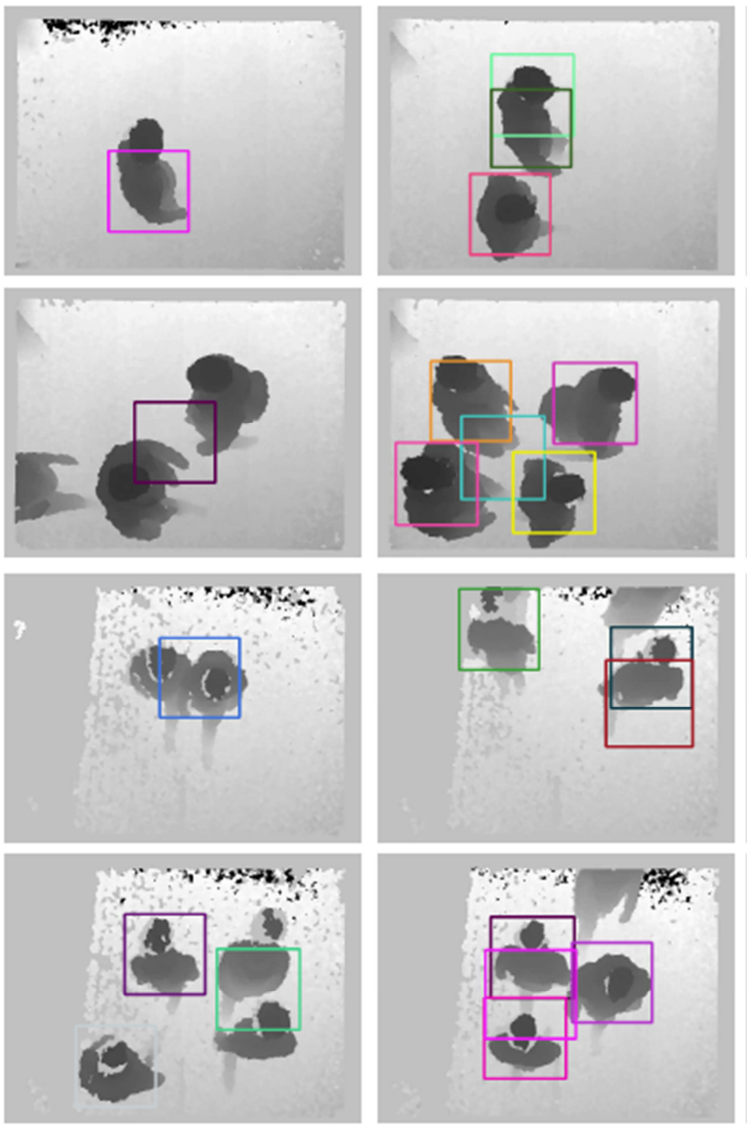


Figura 44 - Esempi di FN e FP del metodo HOG-SVM

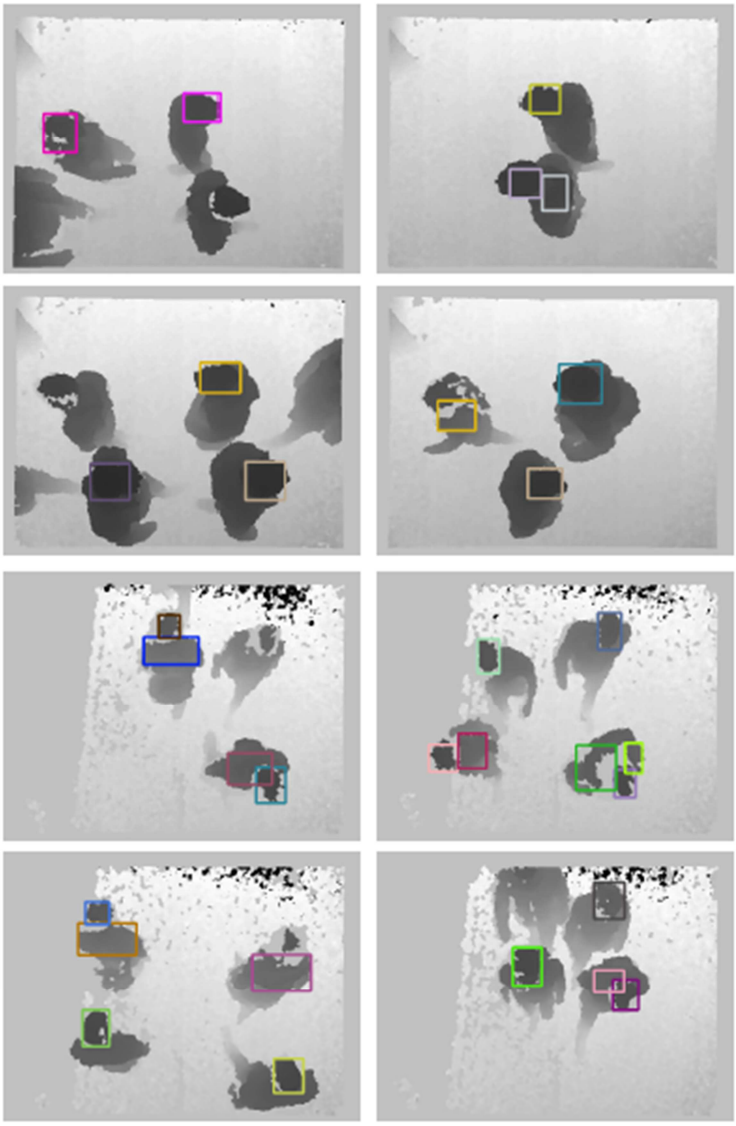


Figura 45 - Esempi di FN e FP del metodo WATERFILLING

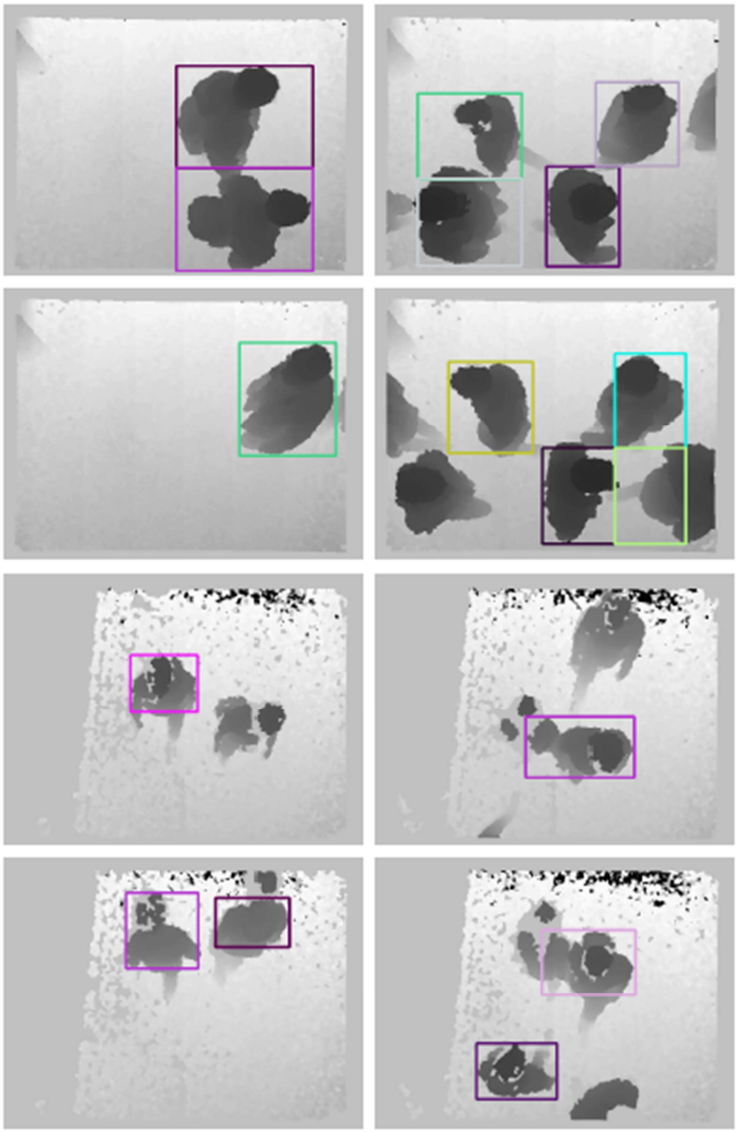


Figura 46 - Esempi di FN e FP del metodo ABSM-Detecion

Altro parametro di valutazione da considerare è l'**onere computazionale** associato ai singoli metodi di rilevamento. Di seguito si presentano i risultati di un'analisi sull'utilizzo delle risorse computazionali richieste dai metodi avvalendosi di una piattaforma PC con le caratteristiche riportate in Tabella 19. Per valutare correttamente le prestazioni, tutti i metodo sono elaborati sulla stessa piattaforma hardware e implementati con la medesima libreria di analisi video (*OpenCV 2.4.13*). Nella Tabella 20 si riportano i valori medi (per frame) del tempo di elaborazione (espresso in millisecondi) e della percentuale sul tempo totale, per ogni fase del metodo analizzato. Inoltre nell'ultima colonna è presente il *frame rate* medio per ogni metodo. Al metodo HOG-SVM fa riferimento un'unica fase, definita *Multi Scale Detection*, in cui a partire dal classificatore addestrato si ottengono i risultati richiamando la funzione *detectMultiScale* di *OpenCV* (per dettagli si rimanda alla documentazione ufficiale[1]).

Parametro	Caratteristiche
OS	Ubuntu 16.04.1 LTS
Vision Library	Opencv 2.4.13
CPU	Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz
FPU	Si
Cache alignment	64
Address sizes	36 bits physical, 48 bits virtual
Cache size	3072 kB
RAM	3898888 kB

Tabella 19 - Caratteristiche della piattaforma PC utilizzata

[1] http://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html

Metodo	Fase	Tempi (ms)	%	FPS
ABSM_DETECTION	Foreground Detection	7.646	42.120	56
	Background Subtraction	8.499	46.821	
	Morphological Operation	0.632	3.482	
	Blob Detection	0.980	5.395	
	Blob Filter and Split	0.397	2.182	
WATERFILLING	Water Drop	227.798	98.714	5
	Water Filter	1.994	0.873	
	Morphological Operation	0.011	0.005	
	Blob Detection	0.920	0.404	
	Blob Filter	0.009	0.004	
HOG-SVM	Multi Scale Detection	707.161	100	1

Tabella 20 – Utilizzo delle risorse computazionali richieste dai singoli metodi adoperando una piattaforma PC

Dall'analisi dei risultati in Tabella 20, il metodo HOG-SVM presenta i tempi di elaborazione più elevati e di conseguenza *frame rate* con valore più basso (pari a *1*); difatti, rispetto a quest'ultimo, WATERFILLING impiega circa il *33%* del tempo con *frame rate* medio pari a *5*, mentre ABSM_DETECTION, il metodo più performante, mostra tempi pari a circa il *3%* rispetto ad HOG-SVM e circa l'*8%* rispetto a WATERFILLING, con frame rate medio pari a *56*. Analizzando nel dettaglio il metodo ABSM_DETECTION si evince che, le fasi utilizzate per ottenere la *foreground mask* (*foreground detection* e *background update*) sono computazionalmente le più onerose e richiedono circa il *90%* del tempo, mentre le altre fasi hanno un ridotto impatto sui tempi di elaborazione, con quella di *blob detection* (tra queste ultime la più rilevante) pari a circa il *5%*. Per quanto concerne il metodo WATERFILLING, la fase di *water drop* è la più onerosa con circa il

98% del tempo, mentre le altre fasi hanno un ridotto impatto sui tempi di elaborazione, dove le più significative sono *water filling* e *blob detection* con un peso rispettivamente pari a circa lo *0.9%* e *0.4%*.

I risultati presentati in precedenza sono riportati nei seguenti lavori del 2017:

- *V. Carletti, L. Del Pizzo, G. Percannella, and M. Vento. Benchmarking two algorithms for people detection from topview depth cameras. In International Conference on Pattern Recognition and Image Analysis. Springer, 2017.*
- *V. Carletti, L. Del Pizzo, G. Percannella, and M. Vento. An efficient and effective method for people detection from top-view depth cameras. In 14th IEEE International Conference on Advanced Video and Signal based Surveillance, 2017.*

Di seguito si riportano i risultati dell'**analisi sperimentale sul sistema multi-esperto** presentato nel paragrafo 4.3, effettuata variando la soglia sulla percentuale di sovrapposizione (parametro β). Questa valutazione è necessaria poiché gli esperti utilizzano approcci differenti per individuare le persone nella scena e questo porta a *bounding box* disuguali. In Tabella 21, Tabella 22 e Tabella 23 si riportano i parametri prestazionali delle tre regole di decisione del MES al variare dello scenario con il valore di $\beta=0.5$. Si noti che la regola a MAGGIORANZA è quella con le prestazioni migliori, questo è vero anche al variare del valore di β.

Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
indoor	1	7,788	0	5,890	0.569	1.000	0.726
	2	3,305	0	2,732	0.547	1.000	0.708
	3	1,314	0	1,167	0.530	1.000	0.692
	4	381	0	315	0.547	1.000	0.708
outdoor	1	8,753	0	7,940	0.524	1.000	0.688
	2	5,000	2	4,166	0.545	1.000	0.706
	3	1,282	0	1,133	0.531	1.000	0.694
	4	433	0	360	0.546	1.000	0.706

Tabella 21 – Multi-esperto con regola del MINIMO e soglia a 0.5

Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
indoor	1	1,487	6,301	1	0.999	0.191	0.321
	2	521	2,784	1	0.998	0.158	0.272
	3	168	1,146	0	1.000	0.128	0.227
	4	57	324	0	1.000	0.150	0.260
outdoor	1	4,174	4,579	13	0.997	0.477	0.645
	2	2,862	2,140	7	0.998	0.572	0.727
	3	795	487	1	0.999	0.620	0.765
	4	260	173	0	1.000	0.600	0.750

Tabella 22 - Multi-esperto con regola del MASSIMO e soglia a 0.5

Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
indoor	1	7,713	75	72	0.991	0.990	0.991
	2	3,172	133	44	0.986	0.960	0.973
	3	1,247	67	17	0.987	0.949	0.967
	4	365	16	0	1.000	0.958	0.979
outdoor	1	8,719	34	245	0.973	0.996	0.984
	2	4,939	63	148	0.971	0.987	0.979
	3	1,276	6	26	0.980	0.995	0.988
	4	429	4	7	0.984	0.991	0.987

Tabella 23 - Multi-esperto con regola di MAGGIORANZA e soglia a 0.5

Di seguito si confrontano le prestazioni (*Precision*, *Recall* ed *f-index*) degli esperti e delle regole del MES, per quest'ultimo si riporta l'andamento al variare del parametro β (le funzioni dei tre esperti rimangono costanti perché non dipendono dal parametro). La Figura 47 riporta la *precision* di tutti gli esperti e di tutte le regole del MES in

funzione di β. Come atteso, la regola del MASSIMO presenta la *precision* maggiore, difatti decidendo all'unanimità è in grado di ridurre al minimo i falsi positivi; segue la regola di MAGGIORANZA che discende leggermente rispetto a quella di MASSIMO per $\beta>0.3$, ma è al di sopra della retta di HOG-SVM, con cui arriva a coincidere per $\beta=0.5$. Entrambe le regole presentano prestazioni superiori rispetto a quella del MINIMO, che a sua volta supera WATERFILLING (il meno vantaggioso tra gli esperti) solo con $\beta<0.4$. Difatti, per la regola del MINIMO è soddisfatta se almeno un esperto fornisce una *detection,* questo porta ad un incremento dei falsi positivi.

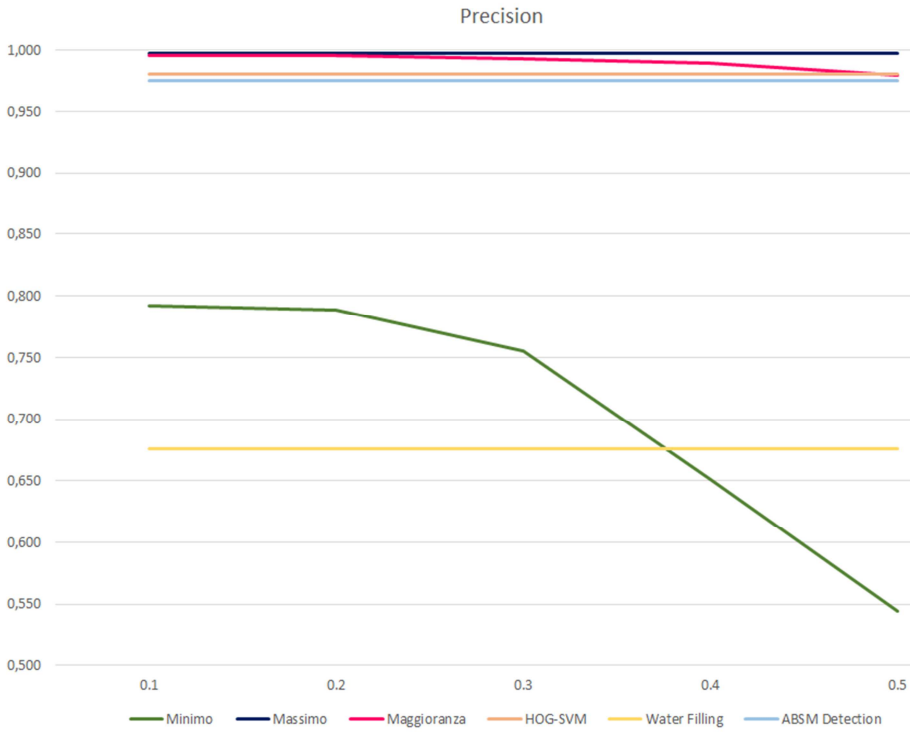


Figura 47 - Precision degli esperti e del multi-esperto

La Figura 48 riporta la *recall* di tutti gli esperti e di tutte le regole del MES in funzione di β. Le funzioni delle regole di MINIMO e MAGGIORANZA sono costanti e pari all'unità, in linea con i risultati di HOG-SVM. La regola di MASSIMO, basandosi solo sulle combinazioni in cui sono presenti tutti gli esperti, perde alcune persone presenti nella scena, inoltre analizzando il grafico essa coincide con ABSM Detection fino a $\beta=0.2$, poi peggiora fino ad essere superata anche da WATERFILLING per valori di $\beta>0.3$.

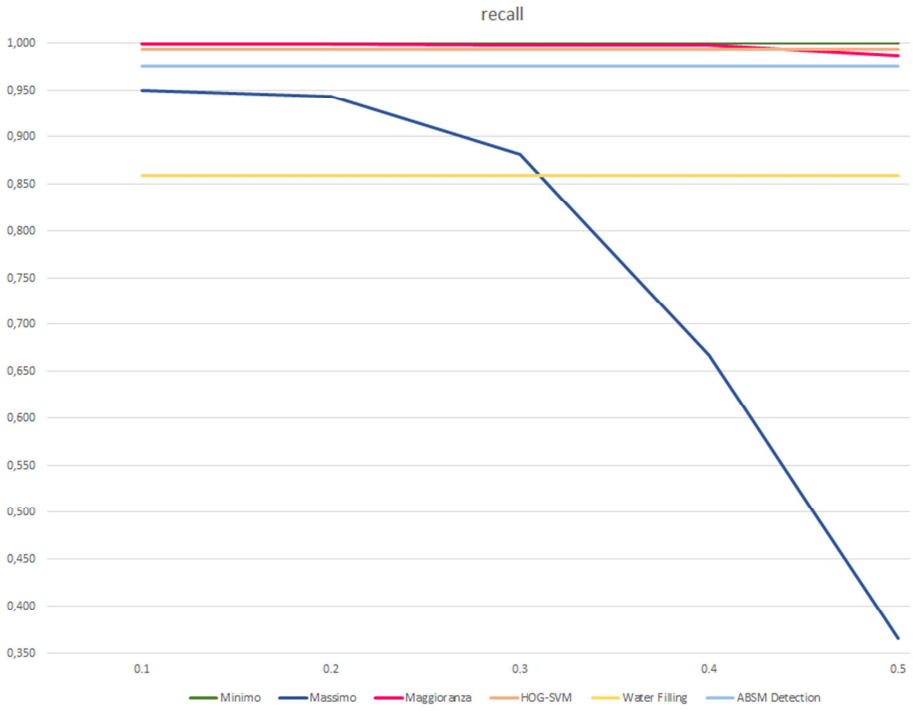


Figura 48 - Recall degli esperti e del multi-esperto

La Figura 49 riporta l'*f-index* di tutti gli esperti e di tutte le regole del MES in funzione di β. La regola a MAGGIORANZA ha mediamente prestazioni migliori rispetto alle altre regole del MES, inoltre si comporta decisamente meglio rispetto a WATERFILLING e ABSM_DETECTION e tende ad essere migliore di HOG-SVM (dello *0.010* con β*=0.1*). Visti i risultati, si può affermare che il MES con regola di MAGGIORANZA presenta il comportamento migliore anche in confronto con i singoli esperti.

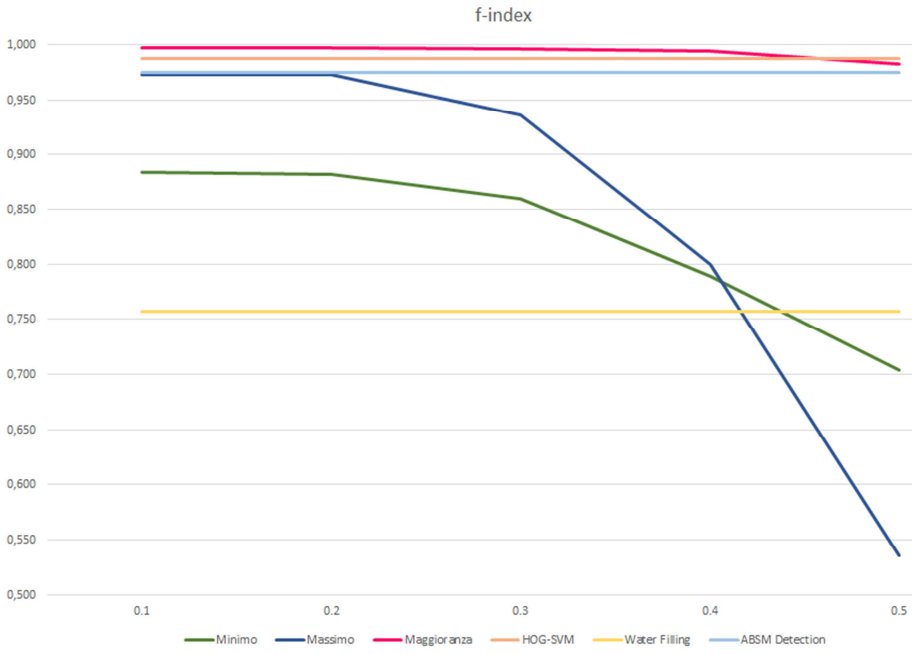


Figura 49 - f-index degli esperti e del multi-esperto

Di seguito si confrontano le prestazioni (*Precision*, *Recall* ed *f-index*) degli esperti e di tutte le regole del MES al variare dello scenario di riferimento e del parametro β.

Le Figura 50 e Figura 51 riportano la *precision,* rispettivamente nello scenario *indoor* e *outdoor,* di tutti gli esperti e di tutte le regole del MES in funzione del parametro β. Le prestazioni migliori si ottengono

con la regola del MASSIMO, di MAGGIORANZA, gli esperti HOG-SVM e ABSM_DETECTION. Di questi, HOG-SVM nello scenario *indoor* fornisce una soluzione meno efficace; mentre nello scenario *outdoor* è ABSM_DETECTION che riduce le prestazioni. In entrambi gli scenari le due regole del MES sono mediamente le migliori. Si noti, nello scenario outdoor, un brusco calo delle prestazioni sia di WATERFILLING che della regola di MINIMO, le prestazioni di quest'ultima sono chiaramente influenzate dal netto peggioramento dell'esperto.

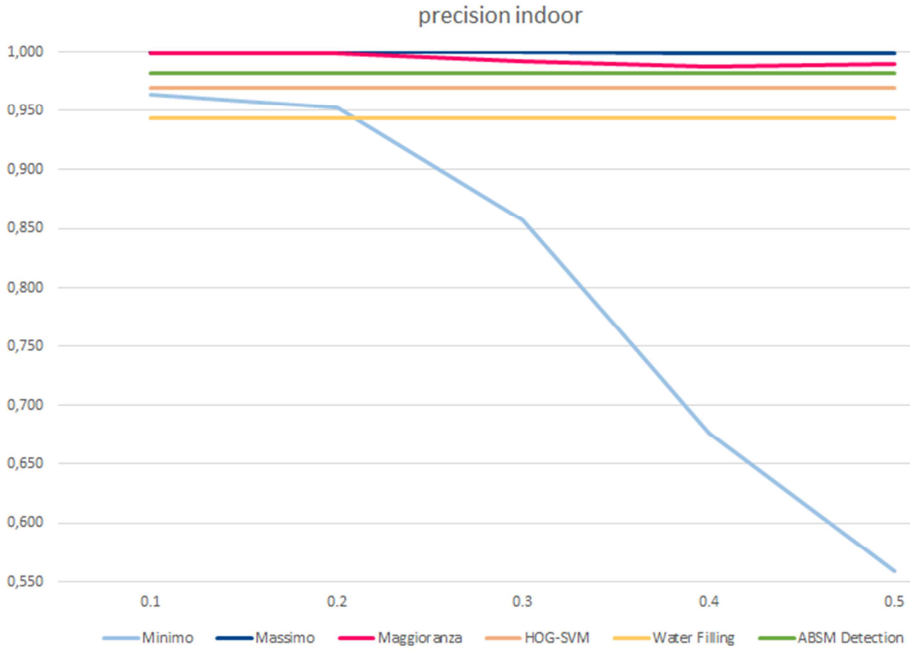


Figura 50 - Precision su scenario indoor al variare del parametro β

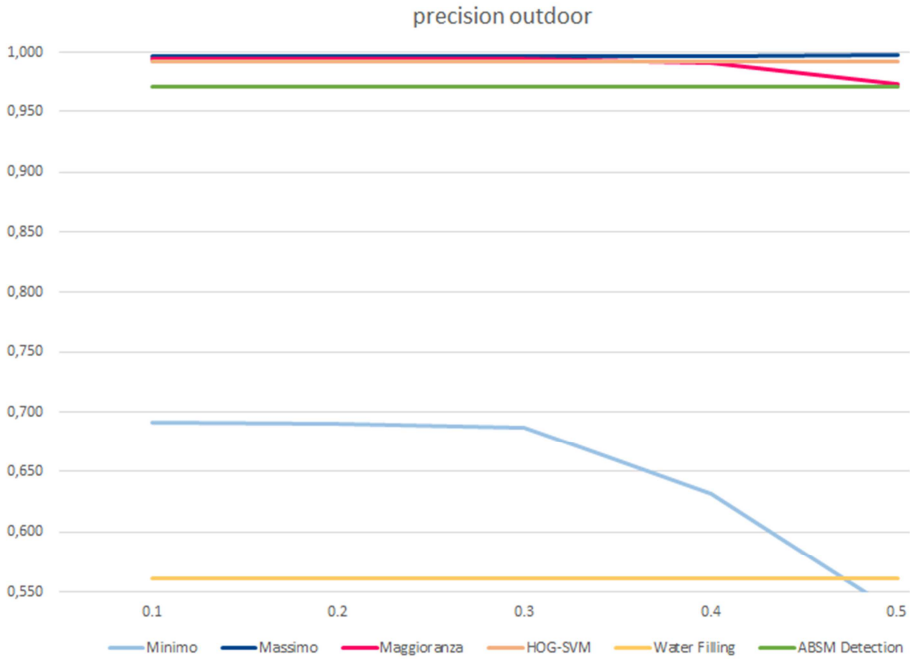


Figura 51 - Precision su scenario outdoor al variare del parametro β

Le Figura 52 e Figura 53 riportano la *recall,* rispettivamente nello scenario *indoor* e *outdoor,* di tutti gli esperti e di tutte le regole del MES in funzione di β. Le prestazioni migliori si ottengono con la regola del MINIMO, di MAGGIORANZA e gli esperti HOG-SVM e ABSM_DETECTION. Si noti che nello scenario *indoor* con $\beta>0.4$ la regola di MAGGIORANZA ha prestazioni leggermente inferiori a HOG-SVM, mentre si equivalgono con valori inferiori di β. Nello scenario *outdoor* le regole di MINIMO e MASSIMO hanno prestazioni maggiori rispetto a HOG-SVM.

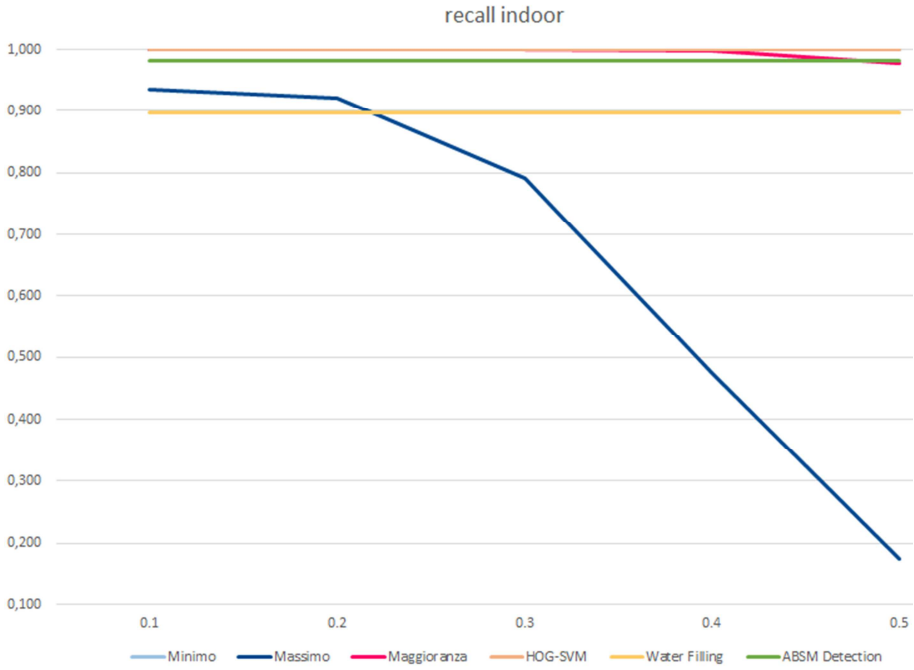


Figura 52 - Recall su scenario indoor al variare del parametro β

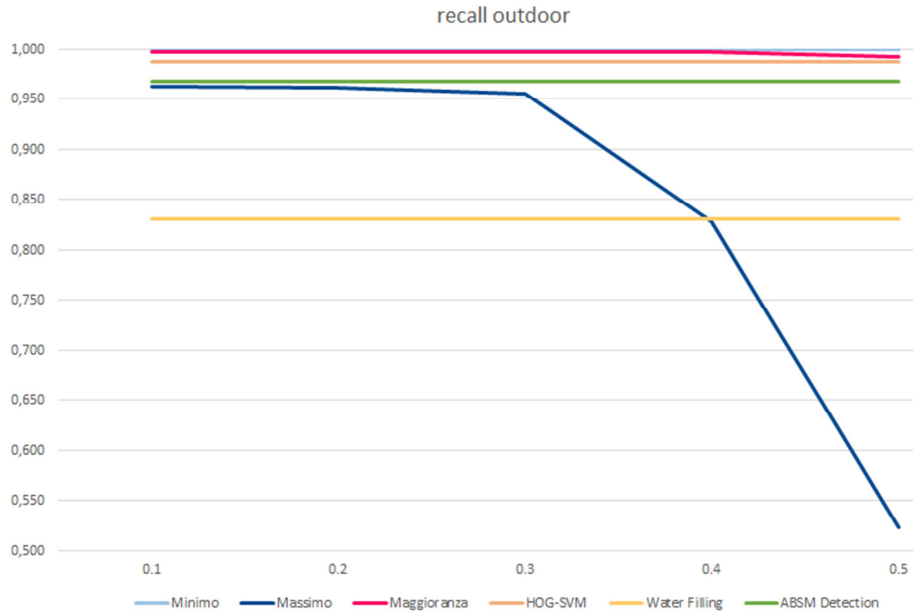


Figura 53 - Recall su scenario outdoor al variare del parametro β

Le Figura 54 e Figura 55 riportano l'*f-index,* rispettivamente sullo scenario *indoor* e *outdoor,* di tutti gli esperti e di tutte le regole del MES in funzione del parametro β. Nello scenario *indoor*, per $\beta<0.3$ tutti le regole presentano un *f-index* superiore allo *0.900* mentre con $\beta>0.2$ MINIMO e MASSIMO mostrano un netto peggioramento delle prestazioni. La regola di MAGGIORANZA presenta prestazioni superiori per $\beta<0.5$ rispetto alle altre regole e agli esperti, mentre per $\beta=0.5$ converge con HOG-SVM e ABSM_DETECTION. Nello scenario *outdoor*, WATERFILLING mostra i suoi limiti utilizzando immagini di profondità con molto rumore, il calo delle prestazioni influenza la regola del MINIMO. Anche in questo caso, la regola di MAGGIORANZA presenta prestazioni superiori agli altri per $\beta<0.5$, mentre per $\beta=0.5$ le prestazioni sono leggermente inferiori rispetto a HOG-SVM.

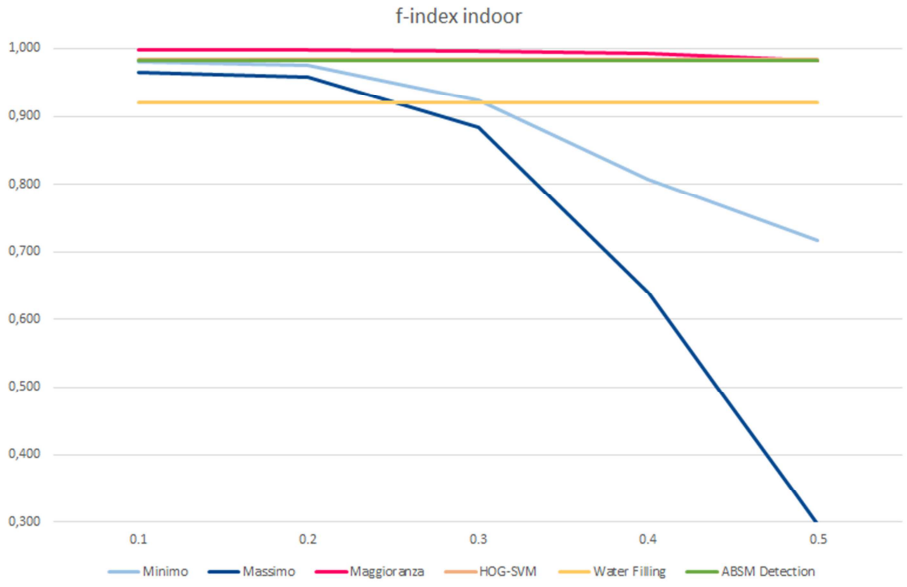


Figura 54 - f-index su scenario indoor al variare del parametro β

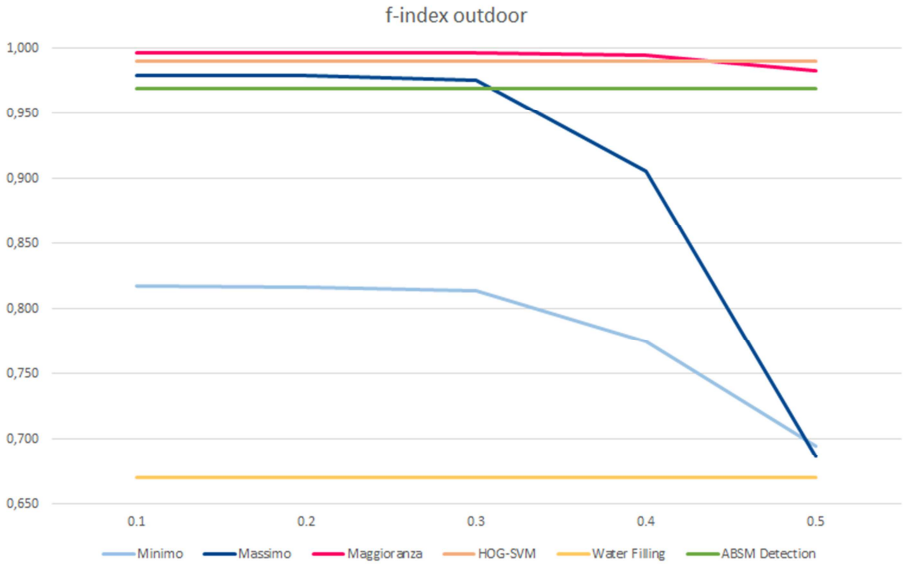


Figura 55 - f-index su scenario outdoor al variare del parametro β

In definitiva, si può affermare che con un valore di $\beta=0.4$ (o inferiore) le prestazioni del MES con regola di MAGGIORANZA superano quelle dei singoli esperti, al variare dello scenario di riferimento e della densità di persone presenti nella scena. La Figura 56 riporta la variazione delle prestazione dei singoli esperti rispetto al MES con regola di MAGGIORANZA e parametro β impostato a *0.3*.

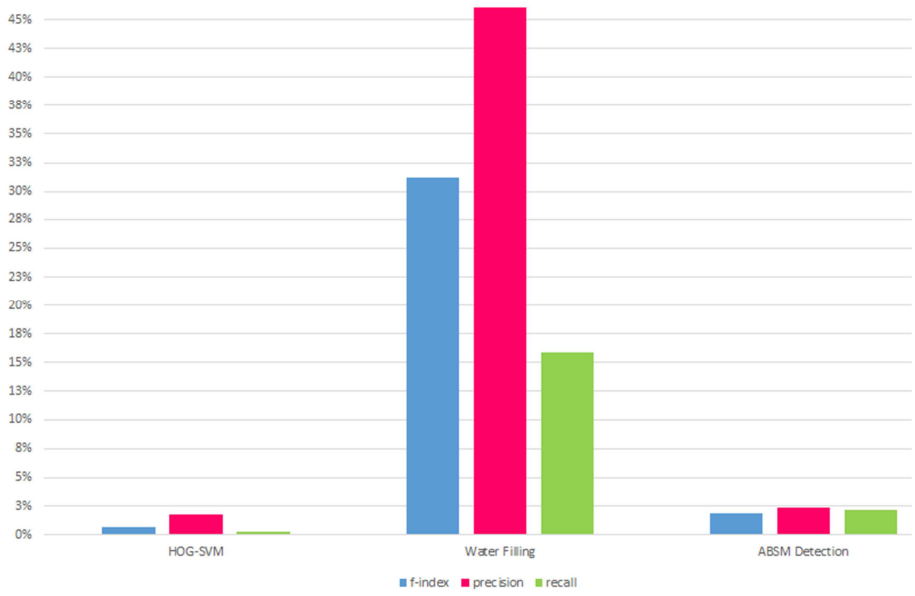


Figura 56 - Variazione prestazioni rispetto al multi-esperto (regola MAGGIORANZA *β=0.3*)

In base a quanto detto, si analizzano nel dettaglio le prestazioni di HOG-SVM e del MES con regola di MAGGIORANZA e *β=0.3*, riportati nella Tabella 24. Esaminando i risultati al variare dello scenario, come riportato in Tabella 25, si desume che il sistema multi-esperto presenta una significativa riduzione dei FP nello scenario *indoor* (aumento della *precision* pari a *0.023*) e nello scenario *outdoor* (aumento della *precision* pari a *0.002*) dove si registra anche una significativa diminuzione dei FN (aumento della *recall* pari a *0.010*) dovuta alla combinazione dei risultati degli altri due esperti (quando HOG-SVM non rileva nessuna persona) e, principalmente, quando la combinazione del rettangolo di HOG-SVM con almeno un altro dei due esperti consente di ottenere un nuovo rettangolo che si sovrappone con la *ground truth* superando la soglia del *50% (*decrementando in questo caso anche i FP come ad es. quando l'esperto rileva un *box* che include non solo testa e spalla ma anche altre parti del corpo, oppure quando rileva un solo *box* centrale tra due persone molto ravvicinate).

Il miglioramento sui FP, particolarmente evidente nello scenario *indoor*, è intrinseco della regola di MAGGIORANZA: considerando solo combinazione di almeno due esperti, si evitano i FP che HOG-SVM rileva tra due o più persone ravvicinate, quando una persona sta uscendo dalla scena (dove l'esperto rileva un *box* in corrispondenza della schiena della persona) e casi in cui il *box* rilevato non supera la soglia di sovrapposizione con la *ground truth* (come chiarito in precedenza). In tal scenario, anche l'esperto WATERFILLING presenta buone prestazione, e ciò consente al MES di ottenere un limitato numero di FP in ambiente interno. Una ulteriore considerazione riguarda l'aumento di FN in ambiente *indoor*, tale situazione è dovuta a casi in cui il solo esperto HOG-SVM rileva correttamente la persona nella scena; tale incremento risulta insignificante ai fini delle prestazioni finali del MES (riduzione della *recall* pari a *0.001*).

Quindi il sistema multi-esperto riesce a compensare l'aumento di rumore che si rileva nello scenario *outdoor*, utilizzando soprattutto i risultati di ABSM_DETECTION che permettono di ottenere, rispetto a HOG-SVM, *box* che si sovrappongono maggiormente alla *ground truth*.

Quanto affermato in precedenza è comprovato anche dai risultati presenti in Tabella 26, che riporta il confronto tra HOG_SVM e MES, con regola di MAGGIORANZA e $\beta=0.3$, al variare del numero di persone presenti nella scena. Difatti, è evidente il netto miglioramento del multi-esperto in *frame* in cui sono presenti gruppi di persone.

Metodo	Scenario	Persone	TP	FN	FP	Precision	Recall	f-index
HOG-SVM	*indoor*	1	7,788	0	199	0.975	1.000	0.987
		2	3,303	2	122	0.964	0.999	0.982
		3	1,313	1	75	0.946	0.999	0.972
		4	381	0	14	0.965	1.000	0.982
	outdoor	1	8,728	25	50	0.994	0.997	0.996
		2	4,870	132	63	0.987	0.974	0.980
		3	1,267	15	9	0.993	0.988	0.991
		4	419	14	5	0.988	0.968	0.978
MAGGIORANZA β=0.3	*indoor*	1	7,785	3	41	0.995	1.000	0.997
		2	3,300	5	40	0.988	0.998	0.993
		3	1,313	1	16	0.988	0.999	0.994
		4	381	0	2	0.995	1.000	0.997
	outdoor	1	8,742	11	47	0.995	0.999	0.997
		2	4,985	17	30	0.994	0.997	0.995
		3	1,276	6	14	0.989	0.995	0.992
		4	432	1	1	0.998	0.998	0.998

Tabella 24 - Confronto tra HOG_SVM e multi-esperto con regola di MAGGIORANZA e *β=0.3*, al variare dello scenario e del numero di persone

Metodo	Scenario	TP	FN	FP	Precision	Recall	f-index
HOG-SVM	*indoor*	12,785	3	410	0.969	1.000	0.984
	outdoor	15,284	186	127	0.992	0.988	0.990
MAGGIORANZA β=0.3	*indoor*	12,779	9	99	0.992	0.999	0.996
	outdoor	15,435	35	92	0.994	0.998	0.996

Tabella 25 - Confronto tra HOG_SVM e multi-esperto con regola di MAGGIORANZA e *β=0.3*, al variare dello scenario

Metodo	Persone	TP	FN	FP	Precision	Recall	f-index
HOG-SVM	1	16516	25	249	0.985	0.998	0.992
	2	8173	134	185	0.978	0.984	0.981
	3	2580	16	84	0.968	0.994	0.981
	4	800	14	19	0.977	0.983	0.980
***MAGGIORANZA* β=0.3**	1	16527	14	88	0.995	0.999	0.997
	2	8285	22	70	0.992	0.997	0.994
	3	2589	7	30	0.989	0.997	0.993
	4	813	1	3	0.996	0.999	0.998

Tabella 26 - Confronto tra HOG-SVM e multi-esperto con regola di MAGGIORANZA e *β=0.3*, al variare del numero di persone

Di seguito si presentano alcuni *frame* che mostrano le casistiche sopraindicate. Con riferimento allo scenario *indoor*, le Figura 57 e Figura 58 presentano rispettivamente casi in cui le prestazioni del MES (linea rossa) sono migliori rispetto a quelle di HOG-SVM (linea verde) e casi in cui entrambi i metodi presentano falsi positivi e falsi negativi. Le Figura 59 e Figura 60 illustrano rispettivamente casi in cui il MES (linea rossa) presenta prestazioni migliori rispetto a quelle di HOG-SVM (linea verde) e casi in cui entrambi i metodi presentano falsi positivi e falsi negativi con telecamera installata in ambiente esterno.

I casi in cui il MES non supera i limiti di HOG-SVM sono dovuti a due principali casistiche (Figura 58 e Figura 60):

1. almeno un altro esperto ha rilevato un *box* aggiuntiva tra due o più persone o per la stessa persona;
2. il rettangolo risultante dalla combinazione di due o più esperti non raggiunge la soglia di sovrapposizione con la *ground truth*.

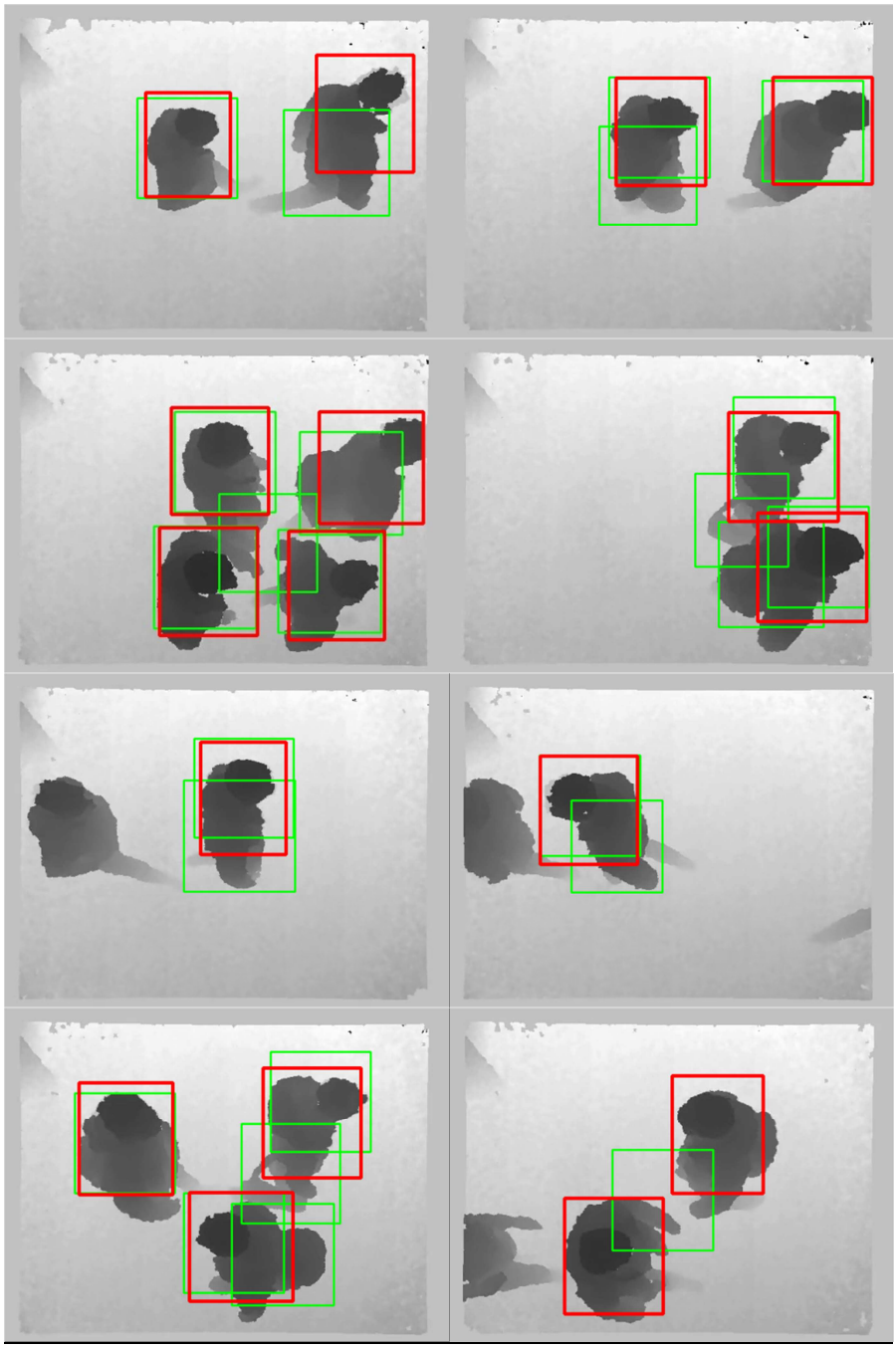


Figura 57 - Casi di prestazioni migliori del multi-esperto (linea rossa) rispetto a HOG-SVM (linea verde) in scenario *indoor*

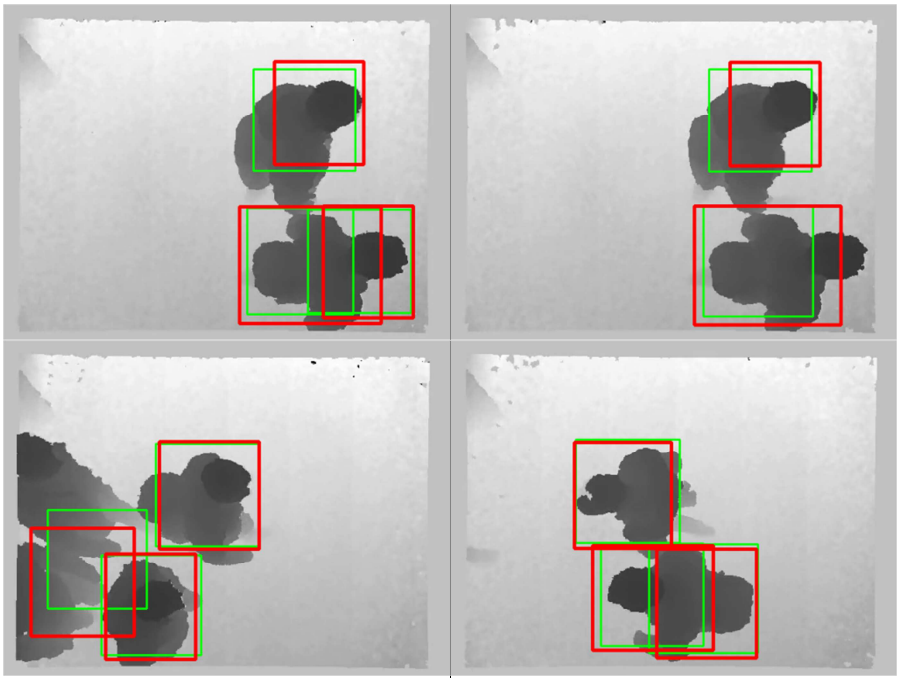


Figura 58 - Casi in cui multi-esperto (linea rossa) e HOG-SVM (linea verde) presentano FP e FN in scenario *indoor*

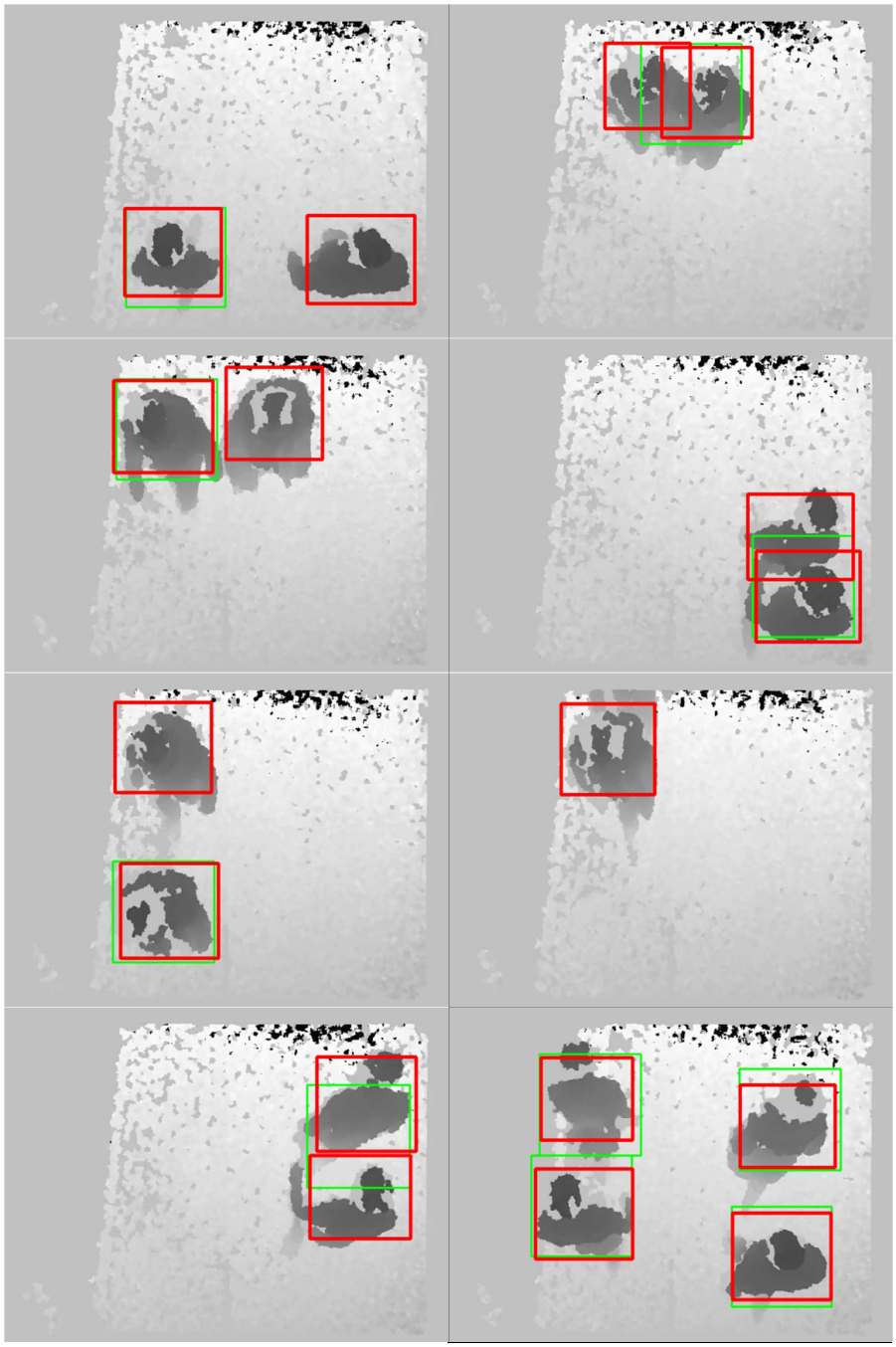


Figura 59 - Casi di prestazioni migliori del multi-esperto (linea rossa) rispetto a HOG-SVM (linea verde) in scenario *outdoor*

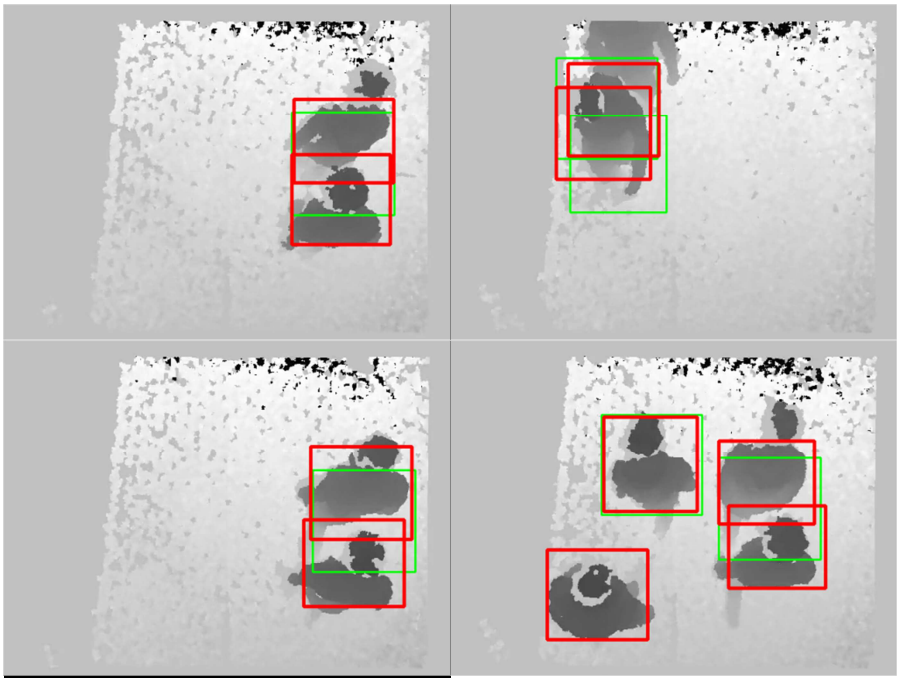


Figura 60 - Casi in cui multi-esperto (linea rossa) e HOG-SVM (linea verde) presentano FP e FN in scenario *outdoor*

Per di più, si analizzano i casi in cui il MES aumenta i FN nello scenario *indoor* rispetto alla soluzione HOG-SVM. La Figura 61 riporta alcuni *false negative* del multi-esperto in ambiente *indoor,* le persone presenti nella scena sono correttamente rilevate solo dall'esperto HOG-SVM ed essendo la regola di MAGGIORANZA basata sulle combinazioni di almeno due esperti tale *box* non viene considerato dal sistema multi-esperto.

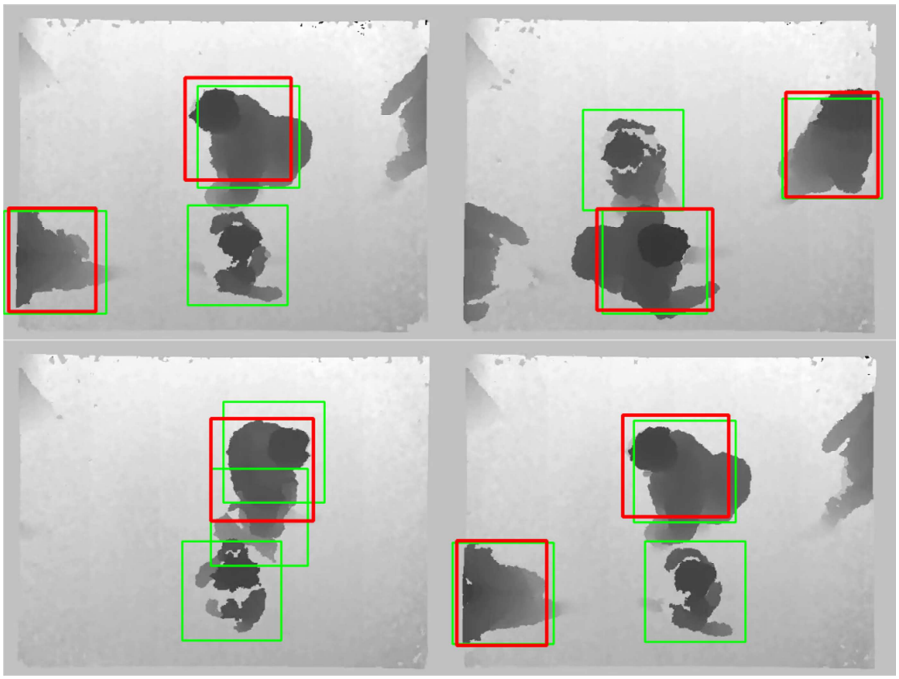


Figura 61 - Casi in cui il multi-esperto (linea rossa) non rileva tutte le persone nella scena, a differenza di HOG-SVM (linea verde) in scenario indoor

Per concludere l'analisi sull'accuratezza del multi-esperto si riportano i risultati ottenuti con l'**oracolo**, il quale, a partire dalla conoscenza della *ground truth,* restituisce "1" (*detection*) se il risultato di almeno un esperto viene associato alla GT e "0" (*no-detection*) se la *ground truth* stabilisce *no-detection* e almeno un esperto indica *no-detection*. Ne consegue che l'oracolo può commette un errore quando tutti gli esperti concordano su *detection* (o *non-detection*) e la *ground truth* indica l'opposto. In Tabella 27 e Tabella 28 si riportano i risultati dell'oracolo con $\beta=0.3$ rispettivamente nello scenario *indoor* e *outdoor*. Le prima quattro colonne rappresentano tutte le possibili combinazioni tra *ground truth* e i risultati dei singoli esperti (ad esclusione della combinazione con tutti "0"). Per ogni combinazione si riporta il risultato del multi-esperto con regola di maggioranza (colonna "MAGG") e dell'oracolo, in rosso si evidenziano i casi di

falsi positivi mentre in arancio i falsi negativi. L'ultima colonna restituisce il numero di occorrenze per ogni predetta combinazione. Dall'analisi dei risultati si evince che la regola di maggioranza ha raggiunto circa le prestazioni massime ottenibili con i tre esperti di partenza, difatti il maggior numero di evenienze, sia per quanto riguarda i falsi positivi che i falsi negativi, sono presenti nelle combinazioni in cui sia il multi-esperto che oracolo compiono un errore. Una ulteriore analisi riguarda la distribuzione delle occorrenze nelle altre combinazioni in cui solo il multi-esperto commette un errore: nei i casi con GT=0, sia in scenario *indoor* che *outdoor,* la distribuzione dei falsi positivi è piuttosto uniforme tra gli esperti, mentre nel caso di GT=1 il metodo HOG-SVM mostra il maggior numero di occorrenze rispetto agli altri esperti (4 in ambiente *indoor*, 9 in *outdoor*), tale aspetto è stato sottolineato anche in precedenza analizzando la Figura 61 e si riferisce a quei casi in cui il rumore introdotto dal sensore porta il solo metodo HOG-SVM a rilevare correttamente la persona.

GT	HOG-SVM	WATER	ABSM	MAGG	ORAC	#
0	0	0	1	0	0	139
0	0	1	0	0	0	674
0	0	1	1	1	0	13
0	1	0	0	0	0	319
0	1	0	1	1	0	11
0	1	1	0	1	0	16
0	1	1	1	1	1	59
1	0	0	0	0	0	5
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	1	1	451
1	1	0	0	0	1	4
1	1	0	1	1	1	3256
1	1	1	0	1	1	526
1	1	1	1	1	1	8546

Tabella 27 - Risultati dell'oracolo nello scenario *indoor* per *β=0.3*

GT	HOG-SVM	WATER	ABSM	MAGG	ORAC	#
0	0	0	1	0	0	362
0	0	1	0	0	0	11743
0	0	1	1	1	0	14
0	1	0	0	0	0	37
0	1	0	1	1	0	9
0	1	1	0	1	0	13
0	1	1	1	1	1	56
1	0	0	0	0	0	24
1	0	0	1	0	1	0
1	0	1	0	0	1	2
1	0	1	1	1	1	451
1	1	0	0	0	1	9
1	1	0	1	1	1	986
1	1	1	0	1	1	1169
1	1	1	1	1	1	12829

Tabella 28 - Risultati dell'oracolo nello scenario *outdoor* per $\beta=0.3$

Capitolo 5. Telecamere e Librerie Utilizzate

5.1 Telecamere Axis

Le *smart camera* utilizzate nelle sperimentazioni sono dell'*Axis Communications*, azienda svedese leader mondiale nella produzione di telecamere IP e codificatori video analogico-digitali. Nel 1999 lancia sul mercato la prima telecamera IP equipaggiata con una versione di Linux per sistemi *embedded*. Ad oggi tutti le telecamere IP prodotte da Axis sono equipaggiate con un sistema operativo basato su Linux.

Le telecamere Axis hanno un'interfaccia di rete (*RJ45* e *Wi-Fi*), ed un web server integrato, utilizzato dall'utente (operatore umano o VMS) per ottenere il flusso video e per la configurazione della stessa telecamera.

A partire dal processore ARTPEC-3 (Dicembre 2009) Axis ha riservato sulle proprie telecamere una parte delle capacità elaborative per l'esecuzione di applicazioni di analisi video realizzate da sviluppatori di terze parti. Axis a tal proposito fornisce un SDK (*Software Development Kit*) per lo sviluppo in linguaggio *C/C++* [53]. L'*Axis Embedded Development SDK* è un insieme di *tool* e interfacce che semplificano lo sviluppo di applicazione di terze parti per i prodotti di Axis.

Le applicazioni realizzate devono superare dei controlli prestazionali attraverso un *tool* Axis di verifica compatibilità. Tali controlli sono volti a verificare che le applicazioni installate non vadano a saturare le risorse della telecamera, e quindi non ci sia un significativo impatto sulle funzionalità di base. Con la propria applicazione in esecuzione sulla telecamera, il *tool* verifica che, per un tempo di almeno *24 ore*, siano rispettate le seguenti condizioni:

- ✓ la telecamera deve riuscire ad erogare almeno 2 flussi video a risoluzione *640*480*, uno di tipo MJPEG (*Motion Joint Photographic Experts Group*) e uno di tipo RTSP (*Real Time Streaming Protocol*), ad un *frame rate* medio maggiore o uguale a *23 FPS*;
- ✓ l'applicazione oggetto di test non deve raggiungere la percentuale massima di utilizzo del processore della telecamera;
- ✓ l'applicazione durante la fase di test deve essere configurate in modalità di funzionamento tipico (nel caso in esame si è provveduto a posizionare la telecamera nei pressi di un varco per il conteggio delle persone);
- ✓ l'applicazione deve effettuare scritture su disco con intervalli di almeno *864 secondi*; quindi al massimo *100* scritture nell'arco di *24 ore* (equi distribuite nel tempo);
- ✓ al termine delle *24 ore* il tool produce come risultato un file XML con i valori delle misurazioni.

Una generica telecamera Axis si presenta come un sistema con le seguenti caratteristiche:

- ✓ Sistema operativo Linux-based;
- ✓ Memoria FLASH;
- ✓ Interfaccia di rete ethernet;
- ✓ Server HTTP;
- ✓ Sorgente video.

La Tabella 29 presenta le caratteristiche tecniche delle telecamere utilizzate per le sperimentazioni descritte nei capitoli precedenti, in particolare per i test degli algoritmo di *moving object detection* e *people counting*.

	CRIS	MIPS
CPU	200 MHz RISC CPU with 5-stage pipeline	400 MHz Multi-Thread RISC CPU with 9-stage pipeline
SIMD	4x8 bit simultaneosly	2x16 bit or 4x8 bit simultaneosly
Arithmetic	Floating-point operations are not supported. 32x32 multiplication for fixed-point and integer	Floating-point operations are not supported. 32x32 multiplication for fixed-point and integer. Support for saturating arithmetic
Cache	Separate instruction and data L1 caches, each of 16 kB 2-way Associative	Separate instruction and data L1 caches, each of 64 kB 4-way Associative
Memory Bus	32 bit data and address width	32 bit data and address width
MMU	Separate Instruction and Data. TLB 64 entry	Separate Instruction and Data. TLB 64 entry
Linux	Support for kernel 2.6	Support for kernel 2.6

Tabella 29 - Caratteristiche tecniche delle architetture CRIS e MIPS

Come indicato in precedenza, Axis fornisce un insieme di strumenti necessari allo sviluppo di applicazioni per l'elaborazione a bordo camera: un pacchetto per ogni compilatore (CRIS e MIPS) per ambiente Linux su architettura *x86*. Poiché l'architettura della macchina su cui viene eseguita la compilazione è diversa dall'architettura della telecamera, si effettua una *cross-compilazione* (compilazione incrociata).

Axis fornisce, inoltre, una serie di interfacce indispensabili per la realizzazione di applicazioni di analisi video: *capture, licensekey, param, net_http, event*.

Tramite l'interfaccia **capture.h** si ha la possibilità di accedere al flusso video catturato dalla telecamera, in formato compresso JPEG o

in formato RAW (non compresso). La sintassi per l'apertura di un flusso video è analoga a quella di una richiesta *HTTP GET*. Di seguito si riporta un esempio di apertura del flusso:

capture_open_stream (IMAGE_UNCOMPRESSED, "fps=30&sdk_format=I420&resolution=320x240")

In seguito si può procedere ad acquisire i successivi *frame*. Ciascun *frame* è univocamente identificato tramite il *timestamp*. Il *frame rate* massimo dipende della telecamera utilizzata. I *frame* ottenuti sono *array di byte*, quindi senza alcuna struttura dati di incapsulamento. I formati disponibili sono:

- ✓ *JPEG*: essendo un formato compresso risulta difficilmente utilizzabile per algoritmi di analisi video. Inoltre, poiché il formato *JPEG* è *lossy* si potrebbe avere una perdita di informazione fondamentale per l'algoritmo di analisi video;
- ✓ *Y800*: rappresenta un'immagine monocanale in scala di grigi, ogni *pixel* è rappresentato da *1 byte*. Le dimensioni di un *frame* risultano essere *M*N byte*;
- ✓ *I420*: rappresenta un formato a tre canali, nello spazio di colore detto *YUV*. Il canale *Y* rappresenta la luminanza, i canali *U* e *V* contengono informazioni di crominanza. I tre canali sono planari (non interlacciati tra loro, per cui tre diversi array di byte) nell'ordine *Y*, *U* e *V*, ed i canali relativi alla crominanza sono campionati sia in orizzontale che in verticale, con un intervallo di campionamento pari a due *pixel*. Un *pixel* occupa *1 byte* per il canale *Y*, ed *1/4 byte* per ciascuno dei canali *U* e *V* (*12 bit* complessivamente), di conseguenza le dimensioni un *frame* sono pari a *3/2*(M*N) byte*. Inoltre, il singolo canale *Y*, se isolato, corrisponde al formato *Y800;*
- ✓ *UYVY*: rappresenta un formato a tre canali, nello spazio di colore detto *YUV*, ma interlacciato. Ogni *pixel* del canale *Y* occupa *1 byte*, mentre ogni *pixel* dei canali *U* e *V* occupa *1/2*

byte. Quindi ogni *frame* occupa *2 byte* per *pixel*, cioè *2*(M*N) byte.*

Il formato utilizzato nelle sperimentazioni per l'acquisizione dei *frame* dalla *smart camera* è il *I420*. Questa scelta è motivata dalla necessità di avere un'immagine con le informazioni sui tre canali e poiché la libreria RAPP opera soltanto su immagini planari monocanali (con la possibilità di utilizzare i tre canali come tre immagini diverse).

L'interfaccia **licensekey.h** consente la gestione delle licenze commerciali Axis per consentire l'esecuzione dell'applicazione ai soli utenti che ne hanno diritto. Il controllo avviene in accordo all'ID univoco associato all'applicazione e all'indirizzo MAC (*Media Access Control*) dell'interfaccia di rete della telecamera.

L'interfaccia **param.h** consente di effettuare operazioni di lettura/scrittura sul *file system*. Utile, ad esempio, per il salvataggio di parametri di funzionamento necessari alle applicazioni.

L'interfaccia **net_http.h** consente all'applicazione di gestire le richieste HTTP, facendo uso del web server presente nella telecamera. Una funzionalità corrisponde alla possibilità di gestire, tramite CGI (*Common Gateway Interface*), pagine dinamiche a cui associare delle funzioni di *callback* per la gestione delle richieste da parte dell'utente o delle applicazione in esecuzione sulla telecamera. Questo permette di realizzare pagine per consentire agli utenti una configurazione *user friendly*, o per la visualizzazione dei risultati in tempo reale, come descritto nel paragrafo 0, tramite *web browser*.

L'interfaccia **event.h** consente di interfacciarsi con il sistema di gestione degli eventi incorporato nella telecamera. Si ha quindi la possibilità di richiamare un particolare evento dall'applicazione sviluppata al verificarsi di determinate condizioni. Ad esempio, per inviare un *frame* al server FTP (*File Transfer Protocol*) in seguito alla

rilevazione di un intrusione all'interno di un area riservata, oppure avviare la registrazione sulla *SD card*.

In seguito alla compilazione della propria applicazione tramite SDK Axis, si ottiene un pacchetto di tipo EAP (*Executable Axis Packet*) che presenta almeno i seguenti file:

- ✓ *param.conf*: un file con i parametri per l'applicazione gestito tramite l'interfaccia *param.h;*
- ✓ *html:* una cartella con le pagine HTML gestite tramite interfaccia *net_http.h*
- ✓ *package.conf:* contiene diverse informazioni sull'applicazione tra le quali nome, versione, id, licenza, pagina di configurazione, permessi, produttore.

Il pacchetto EAP può essere installato sulla telecamera in maniera automatica, in seguito alla compilazione, oppure manualmente utilizzando l'interfaccia web messa a disposizione dal produttore.

Le *smart camera Axis* più recenti consentono di avviare contemporaneamente più applicazioni.

Per lavorare correttamente con la SDK di Axis è richiesta una conoscenza approfondità del linguaggio di programmazione C, dei *tool* come *gcc* e *make*, e del sistema Linux.

5.2 Microsoft *Kinect*

Il dispositivo *Kinect* (Figura 62) e una periferica di gioco per la console *XBOX 360* di *Microsoft*, realizzata inizialmente come semplice dispositivo di controllo dei movimenti per applicazioni video ludiche. Tuttavia, grazie al costo contenuto e alle diverse possibilità di applicazioni offerte, la periferica ha avuto un notevole successo anche nell'ambito della ricerca. Il sensore *Kinect* è stato annunciato al pubblico nel 2009 durante la conferenza stampa della Microsoft all'E3

2009 (*Electronic Entertainment Expo*) con il nome *Project Natal*, poi rinominato *Kinect* alla presentazione ufficiale all'E3 2010. La periferica è in vendita dal 2010 in Europa.

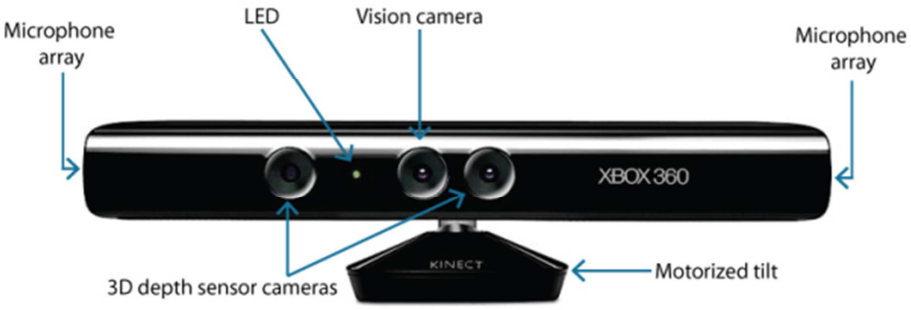


Figura 62 - Sensore *Kinect* della Microsoft

L'hardware del *Kinect* si basa su tecnologie della *3DV*, una compagnia israeliana specializzata in tecnologie di riconoscimento dei movimenti tramite videocamere digitali che Microsoft ha prima finanziato e poi acquisito nel 2009, e sul lavoro della israeliana *PrimeSense*, che ha in seguito fornito in licenza la tecnologia a Microsoft. Il software del *Kinect* è stato, invece, sviluppato internamente ai *Microsoft Game Studios*.

Il *Kinect* è una periferica dotata di una telecamera *RGB*, un sensore di profondità a raggi infrarossi composto da un proiettore e da una telecamera sensibile alla stessa banda.

Il dispositivo dispone anche di un array di microfoni utilizzato dal sistema per la calibrazione dell'ambiente in cui si trova.

Per alimentare la periferica servono *12 W*, e poiché le porte USB sono in grado di fornire in media *2.5 W* di potenza, il *Kinect* richiede un cavo di alimentazione supplementare.

Il *Kinect* per ottenere l'informazione di profondità utilizza la luce strutturata creata dall'emettitore a infrarossi. In particolare, l'emettitore ad infrarossi proietta nell'ambiente *spot* luminosi secondo uno specifico *pattern* (Figura 63).

Figura 63 - Pattern infrarosso emesso dal *Kinect*

Passare da tale *pattern* all'informazione 3D è un'operazione matematica ben definita. La disposizione dei punti del pattern è nota al software del *Kinect* (pattern di riferimento), cioè quello visto dal sensore IR come se fosse proiettato su una superficie posta ad una distanza ben definita e perfettamente parallela al piano della telecamera di profondità. Per ognuno dei punti proiettati l'algoritmo interno al *Kinect* calcola la distanza dal sensore triangolando la posizione attuale con quella del pattern di riferimento.

5.3 Libreria RAPP

La libreria RAPP [55] è una libreria *ANSI C open-source* e portabile su diverse tipologie di piattaforme, che consente allo sviluppatore l'utilizzo di funzioni ottimizzate e affidabili per la manipolazione a

basso livello delle immagini. Opera su immagini binarie con profondità ad *8 bit* e presenta le seguenti caratteristiche:

- ✓ non utilizzano dati *floating-point;*
- ✓ supportano sistemi *little-endian* e *big-endian;*
- ✓ sono scritte in *ISO C99*;
- ✓ non allocano memoria internamente, l'allocazione e la deallocazione è affidata allo sviluppatore;
- ✓ supportano architetture a *16*, *32* e *64 bit*;
- ✓ utilizzano nelle interfacce tipi di dati standard;
- ✓ utilizzano le istruzioni SIMD.

Le immagini elaborate da RAPP sono vettori di *byte* a cui è possibile accedere conoscendo dimensione e *offset*. Il sistema di coordinate utilizzato colloca l'origine degli assi nell'angolo in alto a sinistra dell'immagine, la direzione positiva dell'asse X è verso destra, la direzione positiva dell'asse Y è verso il basso.

In Figura 64 si illustra l'area di memoria (*buf*) in cui è allocata una immagine binaria dove ogni cella rappresenta un *byte*: *off* è l'offset rispetto al primo *byte* dell'immagine (*4 bit*) per le immagini binarie, *width* e *height* la dimensione dell'immagine, *dim* la dimensione in *byte* in una riga, *rapp_alignment* la lunghezza della parola che dipende dall'architettura e nell'esempio è pari a *4 byte*. La zona colorata in celeste corrisponde a quella occupata dall'immagine, l'area in grigio è quella effettivamente occupata in memoria, per ragioni di allineamento ed efficienza. La libreria presenta i seguenti vincoli:

1. la memoria allocata deve iniziare in un intorno di *rapp_alignment;*
2. la dimensione della memoria allocata deve essere un multiplo di *rapp_alignment;*
3. la dimensione in *byte* di una riga deve essere un multiplo di *rapp_alignment.*

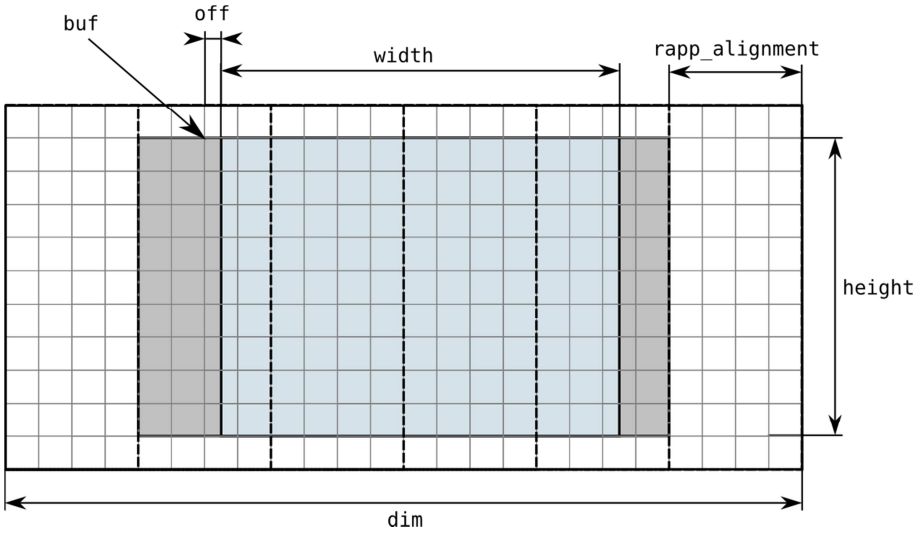


Figura 64 - Rappresentazione di un'immagine allocata in memoria, non allineata

L'allineamento dipende dall'architettura e consente alla libreria di essere portabile su diverse piattaforme.

Poiché spesso è più efficiente elaborare dati allineati rispetto a dati disallineati, le interfacce di RAPP richiedono esplicitamente immagini allineate. Per cui, si hanno i seguenti vincoli:

1. l'area di memoria *buf* deve essere allineata;
2. per le immagini binarie l'offset *off* deve essere pari a zero.

La Figura 65 illustra un'immagine allineata.

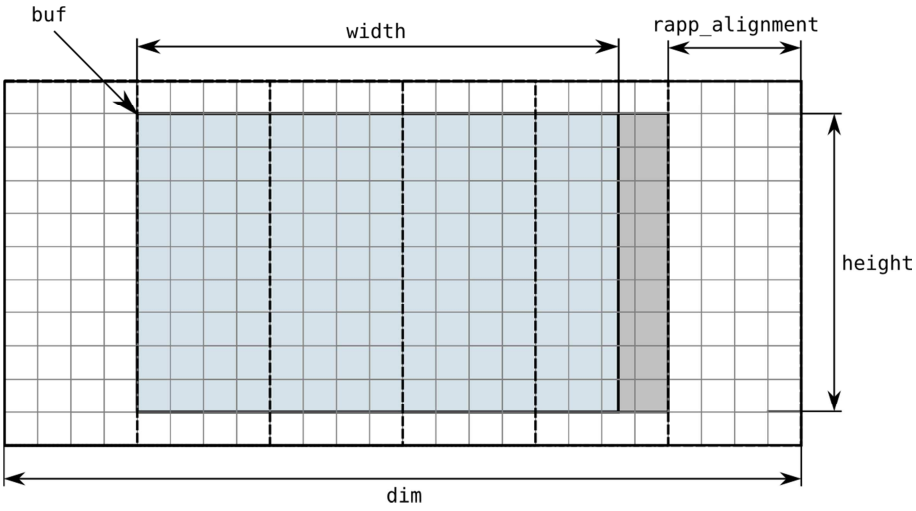


Figura 65 - Rappresentazione di un'immagine allocata in memoria, allineata

La libreria RAPP è progettata per effettuare le più comuni operazioni di elaborazione delle immagini, ad esempio, *spatial filtering, segmentation* e *labeling*, sfruttando le istruzioni SIMD. Per un elenco completo delle operazioni si rimanda alla documentazione ufficiale [55].

5.4 Libreria OpenCV

La libreria OpenCV (*Open Source Computer Vision Library*) è una libreria scritta in *C* e *C++*, costituita da diverse centinaia di funzioni utili nel campo dell'*image processing* e della *computer vision*. Nata da un progetto della Intel che nel 1999 decise di sviluppare un *framework* per la *computer vision* utilizzabile per i loro processori. La libreria è *open source* e presenta una serie completa di funzionalità. La licenza di distribuzione è priva di *royalty* e ciò consente il suo utilizzo anche in prodotti commerciali a condizione di mantenere le note di *copyright*. Altro punto di vantaggio è la portabilità della libreria, sono infatti disponibili le versioni per i sistemi *Microsoft Windows*, *Linux* e *Mac OS X*.

La libreria ha una struttura modulare, come riportato in Figura 66, dove ogni modulo implementa delle specifiche funzionalità. Il modulo *CXCORE* contiene le strutture dati di base con le rispettive funzioni di inizializzazione, le funzioni matematiche, le funzioni di lettura, scrittura e memorizzazione dati, le funzioni di sistema e di gestione degli errori. Il modulo *CV* contiene le funzioni relative all'*image processing*, le funzioni di analisi strutturale e del moto, di *object detection* e ricostruzione 3D. *HIGHGUI* contiene le funzioni GUI (*Graphical User Interface*) e quelle di salvataggio e caricamento immagini, per di più le funzioni di acquisizione video e di gestione delle telecamere. Il modulo *ML* (Machine Learning) contiene classi e funzioni relative all'implementazione e gestione di reti neurali, in particolare di tipo *multilayer perceptrons* (MPL), di classificazione statistica e *clustering* di dati. Infine vi è il modulo *CVAUX* che contiene sia le funzioni basate su algoritmi ancora in fase di sperimentazione, il cui futuro è quello di migrare nel modulo *CV*, e sia le funzioni considerate obsolete e quindi non più supportate. Le funzionalità di tale modulo sono rivolte, ad esempio, alla corrispondenza stereo, al *tracking* 3D, al riconoscimento degli oggetti.

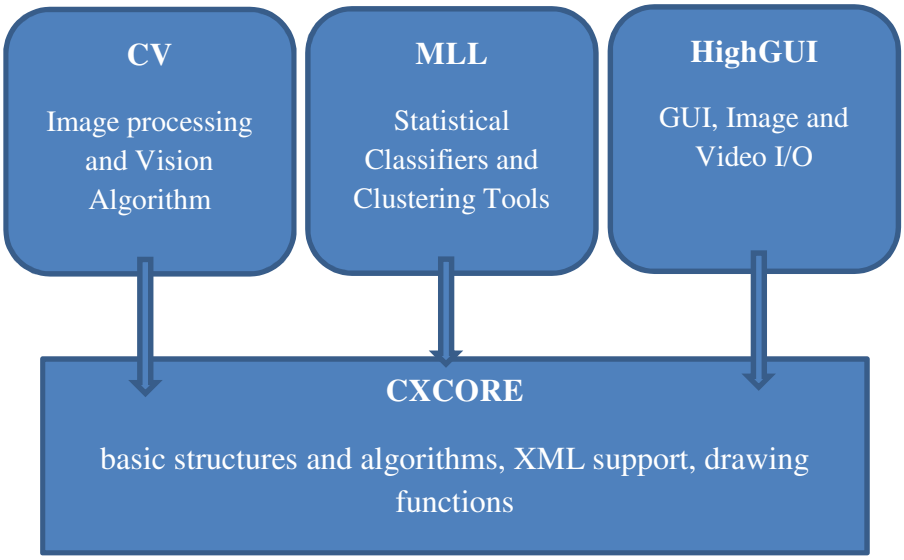


Figura 66 - Architettura della libreria OpenCV

Conclusioni

Il lavoro svolto durante il Dottorato di ricerca, esposto nel presente elaborato, è incentrato sul tema della *Computer Vision* e può essere suddiviso in tre fasi principali.

Nella prima parte è stato effettuato un approfondito studio dell'ambiente di sviluppo (strumenti di compilazione ed installazione su *smart camera* della *Axis Communications* e delle librerie per l'analisi delle immagini) e della letteratura scientifica relativa alle tecniche di *foreground detection* (il primo *step* di numerosi algoritmi di analisi video). L'attenzione è ricaduta sulle soluzioni progettate ed implementate per l'elaborazione di flussi video acquisiti da telecamere fisse per sistemi *embedded*, quindi per dispositivi caratterizzati da ridotte capacità elaborative e di memoria. Lo studio della letteratura scientifica è stato accompagnato da un'attività progettuale e sperimentale finalizzata all'ingegnerizzazione di un algoritmo di *foreground detection* (ampiamente adoperato in letteratura) per consentire la sua esecuzione su dispositivi a basse prestazioni. Per lo sviluppo di tale attività sono state definite ed implementate diverse ottimizzazioni volte a ridurre l'onere computazionale dell'algoritmo. Le ottimizzazioni introdotte sono state successivamente oggetto di un'ampia analisi sperimentale per caratterizzare le prestazioni del metodo su diverse famiglie di processori disponibili su *smart camera* commerciali. Le ottimizzazioni hanno riguardato l'utilizzo delle istruzioni SIMD, la strutturazione del codice per ridurre il numero di cache miss, l'utilizzo delle ottimizzazioni del compilatore, la riduzione delle operazioni in virgola mobile e delle allocazioni dinamiche, ecc. I risultati, riportati nel primo lavoro pubblicato nel 2014 [54], hanno dimostrato che le ottimizzazioni introdotte consentono la esecuzione del metodo su sistemi reali presenti sul mercato.

Nella seconda fase, grazie all'esperienza maturata, è stato realizzato un metodo per il conteggio delle persone che attraversano una linea virtuale mediante la elaborazione del flusso video in tempo reale, acquisito da telecamera fissa installata in posizione zenitale. Successivamente ne sono state caratterizzate le prestazioni in confronto con altri approcci presenti in letteratura. Il modulo, progettato per raggiungere una elevata accuratezza (sia in scenari interni che esterni) con un utilizzo ridotto di risorse computazionali e di memoria, è stato valutato considerando i diversi fattori che possono avere un significativo impatto sulle prestazioni: il tipo di sensore impiegato per l'acquisizione del flusso video, lo scenario di riferimento, la densità degli attraversamenti (persona singola e gruppi di persone), il *frame* rate di acquisizione e la risoluzione del *frame*. Per tale scopo è stato realizzato un *dataset* per il problema in esame con l'impiego di una telecamera ottica tradizionale e di un sensore di profondità installati in posizione zenitale. Il *dataset* è caratterizzato da diversi flussi di persone in transito e da registrazioni in scenari sia interni che esterni. Tale *dataset* è pubblicamente disponibile per la comunità scientifica. In confronto con uno dei *dataset* più utilizzati in letteratura, Zhang et al. [66], quello proposto presenta una complessità notevolmente maggiore e tipica delle applicazioni reali. Rispetto ai *dataset* più recenti presentati da Macias-Guarasa et al. [82] e Liciotti et al. [97], quello proposto presenta un numero maggiore di immagini acquisite e di persone rilevate, con acquisizioni effettuate anche in scenari esterni oltre che interni, inoltre la densità degli attraversamenti risulta maggiore rispetto al *dataset* di Liciotti e paragonabile a quello di Macias-Guarasa. Da questo confronto si deduce che il *dataset* può essere migliorato considerando l'utilizzo di sensori di ultima generazione (come la Kinect v2) e prevedendo la presenza nella scena di oggetti aggiuntivi al fine di estendere l'ambiente di test ad ulteriori casi reali.

Il metodo proposto per il conteggio degli individui presenti nella scena utilizza un algoritmo di *background subtraction* e un sensore virtuale che adotta un automa a stati finiti per determinare la direzione di attraversamento. I risultati riportati in via preliminare nel lavoro del 2015 [68] e poi in forma estesa nel 2016 [69] evidenziano, dal confronto con altri approcci presenti in letteratura, l'efficacia del metodo proposto. La sperimentazione su differenti architetture *hardware* conferma la possibilità di utilizzo del metodo sia in modalità *server-side* che *edge-side*.

Nell'ultima fase è stato proposto un metodo (denominato ABSM_DETECTION), efficace ed efficiente, per il rilevamento e la localizzazione delle persone presenti nella scena inquadrata da telecamera di profondità installata in posizione zenitale. Il metodo utilizza un algoritmo di *background subtraction* per ottenere la *foreground mask* e successivamente impiega operazioni di *postprocessing* per ridurre il rumore, individuare le componenti connesse e localizzare le persone. Per la fase di sperimentazione, il metodo è stato confrontato con due diversi approcci disponibili in letteratura che lavorano con telecamera di profondità in posizione zenitale: il primo di tipo *supervised (HOG-SVM),* il secondo di tipo *unsupervised (WATERFILLING)* più efficiente ma meno accurato del primo. La sperimentazione, su un *dataset* pubblico caratterizzato da due scenari (interno ed esterno) e da diverse densità di persone presenti nella scena, permette di affermare che il metodo proposto raggiunge un'accuratezza prossima a HOG-SVM con una riduzione del tempo di elaborazione di circa un ordine di grandezza. I risultati ottenuti sono riportati in due lavori del 2017 [170][171].

La validità del lavoro effettuato sugli algoritmi di rilevamento e conteggio è confermata anche dalle recenti pubblicazioni presenti in letteratura sul problema in esame ([84][88][90][91][92][94][81][95]) che sottolineano l'importanza dei concetti che hanno accompagnato il presente studio di tesi, tra i quali l'onere computazionale della

soluzione (algoritmi con ridotta complessità e utilizzo di piattaforme *embedded* e GPU), l'aspetto tecnologico (immagini provenienti dai sensori di profondità), la sperimentazione su *dataset* complessi e l'orientamento verso soluzioni che possano essere utilizzate anche in ambienti reali (come ad es. all'ingresso di un autobus per il conteggio del numero dei passeggeri).

In accordo con quanto detto, una possibile evoluzione del lavoro svolto riguarda la sperimentazione dei metodi proposti su diversi scenari reali richiesti dal mercato di riferimento, come ad esempio nel trasporto pubblico, dove le telecamere sono installate in posizione zenitale ma prossime alle teste delle persone (accesso ad un autobus, treno o metro). Inoltre, sarà utile sperimentare gli algoritmi su diverse tipologie di sensori (*stereo camera* o *time of flight*) per valutare quale tecnologia si adatta meglio al problema in esame e allo scenario di riferimento. Da evidenziare che, rispetto a diverse soluzioni presenti in letteratura, i metodi proposti non prevedono una fase di classificazione e l'utilizzo dell'algoritmo di *tracking* (come livello di filtraggio successivo). Questa scelta è motivata dall'elevata accuratezza ottenuta nelle fasi precedenti, dipesa anche dal *dataset* utilizzato in cui sono presenti in minima parte altri oggetti al di fuori delle persone. Ne consegue che, future sperimentazioni su *dataset* più recenti, come quelli analizzati nel paragrafo 3.3 ([82][97]), potranno fornire ulteriori elementi di comparazione per analizzare l'accuratezza dei metodi anche in presenza di oggetti come sedie, scrivanie, carrelli, valigie, ecc.

Una ulteriore attività dell'ultima fase ha riguardato la realizzazione di un sistema multi-esperto, il quale combina i risultati dei tre metodi (chiamati esperti) precedentemente esposti (ABSM_DETECTION, WATERFILLING e HOG-SVM) con regole di combinazione non particolarmente complesse. Il sistema realizzato è di tipo parallelo con regole appartenenti alla categoria *fixed rule*, che si avvalgono direttamente dei risultati fornita dai tre esperti per generare la

decisione finale. Sebbene esistano regole anche più complesse, quelle selezionate si ritengono adatte allo scopo da perseguire, ovvero stabilire se con un sistema multi-esperto si riesca ad ottenere un incremento delle prestazioni. La sperimentazione, realizzata con l'utilizzo delle immagini provenienti da una telecamera di profondità installata in posizione zenitale, attesta l'efficacia del sistema multi-esperto realizzato, il quale raggiunge prestazioni più elevate rispetto ai singoli esperti al variare dello scenario e della densità di individui presenti nella scena. Come ulteriore sviluppo del presente lavoro si potrà introdurre una misura di affidabilità associata a ciascun esperto o alla risposta di ciascun esperto (utilizzando diverse tipologie di classificatori), oltre ad applicare ulteriori regole di combinazione tra quelle analizzate nel paragrafo 4.3. Per di più, si potrà considerare l'applicazione di un multi-esperto seriale o gerarchico.

In conclusione, visti i risultati conseguiti dai metodi proposti risulta possibile una loro applicazione per realizzare algoritmi in grado di rilevare comportamenti anomali delle persone (ad es. individui a terra o con traiettorie anomale) in aree particolarmente sensibili come in un aeroporto o in una metropolitana.

Bibliografia

[1]. Chen, T. P., Haussecker, H., Bovyrin, A., Belenov, R., Rodyushkin, K., Kuranoc, A., & Eruhimov, V. (2005). Computer Vision Workload Analysis: Case Study of Video Surveillance Systems. Intel Technology Journal, 9(2)

[2]. Anderson, C. H., Burt, P. J., & Van Der Wal, G. S. (1985, December). Change detection and tracking using pyramid transform techniques. In 1985 Cambridge Symposium (pp. 72-78). International Society for Optics and Photonics.

[3]. Rosin, P. L., & Ellis, T. J. (1995, July). Image difference threshold strategies and shadow detection. In BMVC (Vol. 95, pp. 347-356).

[4]. Stauffer, C., & Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. IEEE Transactions on pattern analysis and machine intelligence, 22(8), 747-757.

[5]. Cucchiara, R., Grana, C., Piccardi, M., & Prati, A. (2003). Detecting moving objects, ghosts, and shadows in video streams. IEEE transactions on pattern analysis and machine intelligence, 25(10), 1337-1342.

[6]. Sheikh, Y., & Shah, M. (2005). Bayesian modeling of dynamic scenes for object detection. IEEE transactions on pattern analysis and machine intelligence, 27(11), 1778-1792

[7]. Kim, K., Chalidabhongse, T. H., Harwood, D., & Davis, L. (2004, October). Background modeling and subtraction by codebook construction. In Image Processing, 2004. ICIP'04. 2004 International Conference on (Vol. 5, pp. 3061-3064). IEEE.

[8]. Maddalena, L., & Petrosino, A. (2008, September). A self-organizing neural system for background and foreground modeling. In International Conference on Artificial Neural Networks (pp. 652-661). Springer Berlin Heidelberg.

[9]. Li, L., Huang, W., Gu, I. Y., & Tian, Q. (2003, November). Foreground object detection from videos containing complex background. In Proceedings of the eleventh ACM international conference on Multimedia (pp. 2-10). ACM.

[10]. Heikkila, M., & Pietikainen, M. (2006). A texture-based method for modeling the background and detecting moving objects. IEEE transactions on pattern analysis and machine intelligence, 28(4), 657-662.

[11]. Lin, H. H., Liu, T. L., & Chuang, J. H. (2009). Learning a scene background model via classification. IEEE Transactions on Signal Processing, 57(5), 1641-1654.

[12]. Zhao, H., Yang, H., & Zheng, S. (2009, October). An Efficient Method for Detecting Ghosts and Left Objects in Intelligent Video Surveillance. In Image and Signal Processing, 2009. CISP'09. 2nd International Congress on (pp. 1-6). IEEE.

[13]. Varcheie, P. D. Z., Sills-Lavoie, M., & Bilodeau, G. A. (2010). A multiscale region-based motion detection and background subtraction algorithm. Sensors, 10(2), 1041-1061.

[14]. Piccardi, M., 2004. Background subtraction techniques: A review,pp. 3099–3104.

[15]. Benezeth, Y., Jodoin, P., Emile, B., Laurent, H., Rosenberger, C., 2008. Review and evaluation of commonly-implemented background subtraction algorithms.

[16]. Sobral, A., Vacavant, A., 2014. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. Computer Vision and Image Understanding 122, 4–21.

[17]. Jeeva, S., Sivabalakrishnan, M., 2015. Survey on background modeling and foreground detection for real time video surveillance, pp. 566–571.

[18]. Kulchandani, J., Dangarwala, K., 2015. Moving object detection: Review of recent research trends.

[19]. Li, L., Huang, W., Gu, I.H., Tian, Q., 2004. Statistical modeling of complex backgrounds for foreground object detection. IEEE Transactions on Image Processing 13, 1459–1472.

[20]. Chiu, C.C., Ku, M.Y., Liang, L.W., 2010. A robust object segmentation system using a probability-based background extraction algorithm. IEEE Transactions on Circuits and Systems for Video Technology 20, 518–528.

[21]. Tsai, D.M., Chiu, W.Y., 2008. Motion detection using fourier image reconstruction. Pattern Recognition Letters 29, 2145–2155.

[22]. Cheung, S.C., Kamath, C., 2004. Robust techniques for background subtraction in urban traffic video, pp. 881–892.

[23]. Zivkovic, Z., 2004. Improved adaptive gaussian mixture model for background subtraction, pp. 28–31.

[24]. Hofmann, M., Tiefenbacher, P., Rigoll, G., 2012. Background segmentation with feedback: The *pixel*-based adaptive segmenter, pp. 38–43.

[25]. Yao, J., Odobez, J.M., 2007. Multi-layer background subtraction based on color and texture.

[26]. Carletti, V., Del Pizzo, L., Percannella, G., & Vento, M. (2014, November). Foreground detection optimization for SoCs embedded on smart cameras. In Proceedings of the International Conference on Distributed Smart Cameras (p. 31). ACM.

[27]. Wang, Q., Zhou, P., Wu, J., Long, C., 2012. Ra d: Resource-aware fast foreground detection in *embedded* smart cameras

[28]. Tessens, L., Morbee, M., Philips,W., Kleihorst, R., Aghajan, H., 2009. Efficient approximate foreground detection for low-resource devices.

[29]. Casares, M., Velipasalar, S., Pinto, A., 2010. Light-weight salient foreground detection for *embedded* smart cameras. Computer Vision and Image Understanding 114, 1223–1237.

[30]. Bang, J., Kim, D., Eom, H., 2012. Motion object and regional detection method using block-based background difference video *frame*s, pp. 350–357.

[31]. Ye, X., & Wan, W. (2014, July). Fast background modeling using GMM on GPU. In Audio, Language and Image Processing (ICALIP), 2014 International Conference on (pp. 937-941). IEEE.

[32]. Sanchez-Ferreira, C., Mori, J. Y., & Llanos, C. H. (2012, March). Background subtraction algorithm for moving object detection in FPGA. In Programmable Logic (SPL), 2012 VIII Southern Conference on (pp. 1-6). IEEE.

[33]. S. Apewokin, B. Valentine, J. Choi, L. Wills, S. Wills, "Real-Time Adaptive Background Modeling for Multicore *Embedded* Systems", Signal Processing System, Volume 62, Number 1, pages 65-76, 2011.

[34]. Shoaib, M., Elbrandt, T., Zaretskiy, E., Ostermann, J., 2012. Hierarchical bayer-pattern based background subtraction for low resource devices, pp. 1867–1870.

[35]. Gruenwedel, S., Petrovic, N. I., Jovanov, L., Nino-Castaneda, J. O., Pizurica, A., & Philips, W. (2013). Efficient foreground detection for real-time surveillance applications. Electronics Letters, 49(18), 1143-1145.

[36]. Conte, D., Foggia, P., Petretta, M., Tufano, F., & Vento, M. (2005, August). Meeting the application requirements of intelligent video surveillance systems in moving object detection. In International Conference on Pattern Recognition and Image Analysis (pp. 653-662). Springer Berlin Heidelberg.

[37]. Tessens, L., Morbee, M., Lee, H., Philips, W., Aghajan, H., 2008. Principal view determination for camera selection in distributed smart camera networks.

[38]. Guo, J.M., Liu, Y.F., Hsia, C.H., Shih, M.H., Hsu, C.S., 2011. Hierarchical method for foreground detection using codebook model. IEEE Transactions on Circuits and Systems for Video Technology 21, 804–815.

[39]. Wang, Y., Velipasalar, S., Casares, M., 2010. Cooperative object tracking and composite event detection with wireless *embedded* smart cameras. IEEE Transactions on Image Processing 19, 2614–2633.

[40]. Casares, M., Velipasalar, S., 2008. Light-weight salient foreground detection for *embedded* smart cameras.

[41]. Casares, M., Velipasalar, S., 2009. Light-weight salient foreground detection with adaptive memory requirement, pp. 1245–1248.

[42]. K. Toyama, J. Krumm, B. Brumitt, B. Meyers, Wallflower: Principles and Practice of Background Maintenance. Seventh IEEE International Conference on Computer Vision. Vol. 1 (1999) 255 - 261.

[43]. B. Lo, S. Velastin, Automatic congestion detection system for underground platforms. 2001 International symposium on intelligent multimedia, video, and speech processing (2001) 158 - 161.

[44]. I. Haritaoglu, D. Harwood, L.S. Davis. “W4: real-time surveillance of people and their activities”. IEEE Transac. on PAMI. Vol. 22 - 8 (2000) 809 – 830.

[45]. C. R. Wren, A. Azarbayejani, T. Darrel, A. P. Pentland, Pfinder: Real-Time Tracking of the Human Body. IEEE Trans. PAMI. Vol. 19-7, (1997) 780-785.

[46]. J. Heikkilä, O. Silvén, A Real-Time System for Monitoring of Cyclists and Pedestrians. IEEE Workshop on Visual Surveillance, (VS'99) (1999) 74 - 81.

[47]. Y. Matsushita, K. Nishino, K. Ikeuchi, and M. Sakauchi, Illumination Normalization with Time-Dependent Intrinsic

Image for Video Surveillance IEEE Trans. on PAMI. Vol. 26 – 10 (2004) 1336 – 1347

[48]. Apewokin, S., Valentine, B., Wills, L., Wills, S., Gentile, A., 2007. Multimodal mean adaptive backgrounding for *embedded* real-time video surveillance.

[49]. Dowson, D., Landau, B., 1982. The frchet distance between multivariate normal distributions. Journal of Multivariate Analysis 12, 450–455.

[50]. Suhr, J., Jung, H., Li, G., Kim, J., 2011. Mixture of gaussians-based background subtraction for bayer-pattern image sequences. IEEE Transactions on Circuits and Systems for Video Technology 21, 365–370.

[51]. Adams, Jr., J.E., 1995. Interactions between color plane interpolation and other image processing functions in electronic photography, in: Anagnostopoulos, C.N., Lesser, M.P. (Eds.), Cameras and Systems for Electronic Photography and Scientific Imaging, pp. 144–151.

[52]. Guo, J.M., Hsu, C.S., 2010. Cascaded background subtraction using block-based and *pixel*-based codebooks, pp. 1373–1376.

[53]. Axis camera application platform reference web page. http://www.axis.com/techsup/cam_servers/dev/application_platform/index.htm

[54]. Carletti, V., Del Pizzo, L., Percannella, G., & Vento, M. (2014, November). Foreground detection optimization for SoCs embedded on smart cameras. In Proceedings of the International Conference on Distributed Smart Cameras (p. 31). ACM.

[55]. Libreria RAPP. http://www.nongnu.org/rapp/doc/rapp/

[56]. Options That Control Optimization https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

[57]. CUDA C Best Practices Guide http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/

[58]. Chen, C.H., Chen, T.Y., Wang, D.J., Chen, T.J., 2012. A cost-effective people-counter for a crowd of moving people based on two-stage segmentation. JInform Hiding Multimedia Signal Process 3, 2073–4212

[59]. Barandiaran, J., Murguia, B., Boto, F., 2008. Real-time people counting using multiple lines, in: Image Analysis for Multimedia Interactive Services, 2008. WIAMIS'08. Ninth International Workshop on, IEEE. pp. 159–162.

[60]. Chen, T.H., Chen, T.Y., Chen, Z.X., 2006. An intelligent people-flow counting method for passing through a gate, in: Robotics, Automation and Mechatronics, 2006 IEEE Conference on, IEEE. pp. 1–6.

[61]. Antic, B., Letic, D., Culibrk, D., Crnojevic, V., 2009. K-means based segmentation for real-time zenithal people counting, in: Image Processing (ICIP), 2009 16th IEEE International Conference on, IEEE. pp. 2565–2568.

[62]. Mukherjee, S., Saha, B., Jamal, I., Leclerc, R., Ray, N., 2011. A novel *frame*work for automatic passenger counting, in: Image Processing (ICIP), 2011 18th IEEE International Conference on, IEEE. pp. 2969–2972.

[63]. Terada, K., Yoshida, D., Oe, S., Yamaguchi, J., 1999. A method of counting the passing people by using the stereo images, in: Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on, IEEE. pp. 338–342.

[64]. Yahiaoui, T., Meurie, C., Khoudour, L., Cabestaing, F., 2008. A people counting system based on dense and close stereovision, in: Image and Signal Processing. Springer, pp. 59–66.

[65]. Van Oosterhout, T., Bakkes, S., Kr¨ose, B.J., 2011. Head detection in stereo data for people counting and segmentation., in: VISAPP, pp. 620–625.

[66]. Zhang, X., Yan, J., Feng, S., Lei, Z., Yi, D., Li, S.Z., 2012. Water filling: Unsupervised people counting via vertical

Kinect sensor, in: Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on, IEEE. pp. 215–220.

[67]. Vera, P., Zenteno, D., Salas, J., 2013. Counting pedestrians in bidirectional scenarios using zenithal depth images, in: Pattern Recognition. Springer, pp. 84–93.

[68]. Del Pizzo, L., Foggia, P., Greco, A., Percannella, G., & Vento, M. (2015, June). A versatile and effective method for counting people on either RGB or depth overhead cameras. In Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on (pp. 1-6). IEEE.

[69]. Del Pizzo, L., Foggia, P., Greco, A., Percannella, G., & Vento, M. (2016). Counting people by RGB or depth overhead cameras. Pattern Recognition Letters, 81, 41-50.

[70]. VIPER program http://viper-toolkit.sourceforge.net/

[71]. Galčík, F., & Gargalík, R. (2013, October). Real-time depth map based people counting. In International Conference on Advanced Concepts for Intelligent Vision Systems (pp. 330-341). Springer International Publishing.

[72]. Michael Rauter. Reliable human detection and tracking in top-view depth images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 529-534, 2013.

[73]. Lei Zhu and Kin-Hong Wong. Human tracking and counting using the *Kinect* range sensor based on adaboost and kalman filter. In International Symposium on Visual Computing, pages 582-591. Springer, 2013.

[74]. Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In European conference on computational learning theory, pages 23-37. Springer, 1995.

[75]. Daw-Tung Lin and Dong-Han Jhuang. A novel layer-scanning method for improving real-time people counting. In

International Conference on Human-Computer Interaction, pages 661-665. Springer, 2013.

[76]. Ting-En Tseng, An-Sheng Liu, Po-Hao Hsiao, Cheng-Ming Huang, and Li-Chen Fu. Real-time people detection and tracking for *indoor* surveillance using multiple top-view depth cameras. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4077{4082. IEEE, 2014.

[77]. Andres Burbano, Samir Bouaziz, and Marius Vasiliu. 3d-sensing distributed *embedded* system for people tracking and counting. In 2015 International Conference on Computational Science and Computational Intelligence (CSCI), pages 470-475. IEEE, 2015.

[78]. Jakub Nalepa, Janusz Szymanek, and Michal Kawulok. Real-time people counting from depth images. In International Conference: Beyond Databases, Architectures and Structures, pages 387-397. Springer, 2015.

[79]. Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. Cambridge university press, 2000.

[80]. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 886{893. IEEE, 2005.

[81]. Luna, C. A., Losada-Gutierrez, C., Fuentes-Jimenez, D., Fernandez-Rincon, A., Mazo, M., & Macias-Guarasa, J. (2017). Robust people detection using depth information from an overhead Time-of-Flight camera. Expert Systems with Applications, 71, 240-256.

[82]. Macias-Guarasa, J., Losada-Gutierrez, C., Fuentes-Jimenez, D., Garcia-Jimenez, R., Luna, C. A., Fernandez-Rincon, A., &

Mazo, M. (2016). Geintra overhead tof people detection (gotpd1) database

[83]. Stahlschmidt, C., Gavriilidis, A., Velten, J., & Kummert, A. (2014). Applications for a people detection and tracking algorithm using a time-of-flight camera. Multime- dia Tools and Applications (pp. 1–18). doi: 10.1007/s11042- 014- 2260-3

[84]. Li, G., Ren, P., Lyu, X., & Zhang, H. (2016, December). Real-Time Top-View People Counting Based on a Kinect and NVIDIA Jetson TK1 Integrated Platform. In Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on (pp. 468-473). IEEE

[85]. C. Wateosot, N. Suvonvorn et al., "top-view based people counting using mixture of depth and color information.," in The second asian conference on information systems, ACIS. Citeseer, 2013

[86]. H.-C. Chen, Y.-C. Chang, N.-J. Li, C.-F. Weng, and W.-J. Wang, "Real-time people counting method with surveillance cameras implemented on embedded system," in Proceedings of the World Congress on Engineering and Computer Science, vol. 1, 2013.

[87]. C. Wateosot, N. Suvonvorn et al., "top-view based people counting using mixture of depth and color information.," in The second asian conference on information systems, ACIS. Citeseer, 2013.

[88]. Kuo, J. Y., Fan, G. D., & Lai, T. Y. (2016, September). People counting base on head and shoulder information. In Knowledge Engineering and Applications (ICKEA), IEEE International Conference on (pp. 52-55). IEEE.

[89]. Stan Birehfield. (1998). Elliptieal Head Traeking Using Intensity Gradients and Color Histograms. iEEE Conference on Computer Vision and Pattern Recognition. USA.

[90]. Kocak, Y. P., & Sevgen, S. (2017). Detecting and counting people using real-time directional algorithms implemented by compute unified device architecture. Neurocomputing, 248, 105-111.

[91]. Vera, P. , Monjaraz, S. , & Salas, J. (2016). Counting pedestrians with a zenithal arrangement of depth cameras. Machine Vision and Applications, 27 (2), 303–315

[92]. Stahlschmidt, C., Gavriilidis, A., Velten, J., & Kummert, A. (2016). Applications for a people detection and tracking algorithm using a time-of-flight camera. Multimedia Tools and Applications, 75(17), 10769-10786.

[93]. Mallat S (1998) A wavelet tour of signal processing. Academic Press, Elsevier Oxford

[94]. Perng, J. W., Wang, T. Y., Hsu, Y. W., & Wu, B. F. (2016, July). The design and implementation of a vision-based people counting system in buses. In System Science and Engineering (ICSSE), 2016 International Conference on (pp. 1-3). IEEE.

[95]. Kim, H. M., Kim, D., Kim, Y. S., & Kim, K. H. (2016, November). Intuitive Pointing Position Estimation for Large Scale Display Interaction in Top-View Depth Images. In Asian Conference on Computer Vision (pp. 227-238). Springer, Cham.

[96]. Tang, S., Wang, X., Lv, X., Han, T.X., Keller, J., He, Z., Skubic, M., Lao, S.:Histogram of oriented normal vectors for object recognition with a depth sensor. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012. LNCS, vol. 7725, pp. 525–538. Springer, Heidelberg (2013).

[97]. Liciotti, D., Paolanti, M., Frontoni, E., Mancini, A., & Zingaretti, P. (2016, December). Person Re-identification Dataset with RGB-D Camera in a Top-View Configuration. In International Workshop on Face and Facial Expression Recognition from Real World Videos (pp. 1-11). Springer, Cham.

[98]. Saleh, S. A. M., Suandi, S. A., & Ibrahim, H. (2015). Recent survey on crowd density estimation and counting for visual surveillance. Engineering Applications of Artificial Intelligence, 41, 103-114.

[99]. Loy, C. C., Chen, K., Gong, S., & Xiang, T. (2013). Crowd counting and profiling: Methodology and evaluation. In Modeling, Simulation and Visual Analysis of Crowds (pp. 347-382). Springer New York.

[100]. Conte, D., Foggia, P., Percannella, G., & Vento, M. (2013). Counting moving persons in crowded scenes. Machine vision and applications, 24(5), 1029-1042.

[101]. Erickson, V. L., Lin, Y., Kamthe, A., Brahme, R., Surana, A., Cerpa, A. E., ... & Narayanan, S. (2009, November). Energy efficient building environment control strategies using real-time occupancy measurements. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (pp. 19-24). ACM.

[102]. Karpagavalli, P., & Ramprasad, A. V. (2013, April). Estimating the density of the people and counting the number of people in a crowd environment for human safety. In Communications and Signal Processing (ICCSP), 2013 International Conference on (pp. 663-667). IEEE.

[103]. Ma, J., Dai, Y., & Hirota, K. (2017). A Survey of Video-Based Crowd Anomaly Detection in Dense Scenes. Journal Of Advanced Computational Intelligence And Intelligent Informatics, 21(2), 235-246.

[104]. Prati, A., Mikic, I., Trivedi, M. M., & Cucchiara, R. (2003). Detecting moving shadows: algorithms and evaluation. IEEE transactions on pattern analysis and machine intelligence, 25(7), 918-923.

[105]. Conte, D., Foggia, P., Percannella, G., & Vento, M. (2012). Removing object reflections in videos by global optimization.

IEEE Transactions on Circuits and Systems for Video Technology, 22(11), 1623-1633.

[106]. T.G. Dietterich, Approximate statistical test for comparing supervised classification learning algorithms, Neural Computation 10(7) (1998), 1895–1923.

[107]. T. Kohonen, Self-organization and associative memory: 3rd edition, Springer-Verlag New York, Inc., 1989.

[108]. L. Xu, A. Krzyzak and C.Y. Suen, Several methods for combining multiple classifiers and their applications in handwritten character recognition, IEEE Trans. on System, Man and Cybernetics SMC-22(3) (1992), 418–435.

[109]. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110

[110]. Matas J, Chum O, Urban M, Pajdla T (2002) Robust widebaseline stereo from maximally stable extremal regions. Br Mach Vis Conf 1:384–393

[111]. Bay H, Ess A, Tuytelaars T, Gool LV (2008) Surf: speeded up robust features. Comput Vis Image Underst 110(3):346–359

[112]. Tulyakov S, Jaeger S, Govindaraju V, Doermann D (2008) Review of classifier combination methods. Springer, Berlin, Hiedelberg

[113]. Ranawana R, Palade V (2006) Multi-classifier systems: Review and a roadmap for developers. Int J Hybrid Int Syst 3(1):35–61

[114]. L. Rokach, Taxonomy for characterizing ensemble methods in classification tasks: are view and annotated bibliography, Comput.Stat.DataAnal.53 (2009)4046–4072

[115]. M. Woźniak, M.Graña, E.Corchado, A survey of multiple classifier systems as hybrid systems, Inf.Fusion16(2014)3–17.

[116]. Atrey PK, Hossain MA, El Saddik A, Kankanhalli MS (2010) Multimodal fusion for multimedia analysis: a survey. Multimed Syst 16(6):345–379

[117]. Sinha A, Chen H, Danu D, Kirubarajan T, Farooq M (2008) Estimation and decision fusion: A survey. Neurocomputing 71(13):2650–2656

[118]. Kittler J, Hatef M, Duin RP, Matas J (1998) On combining classifiers. IEEE Trans Pattern Anal Mach Intell 20(3):226–239

[119]. L. Lam and C. Y. Suen, "Optimal combinations of pattern classifiers,"Pattern Recognit. Lett., vol. 16, no. 9, pp. 945–954, Sep. 1995

[120]. Ho, T.K., Hull, J.J., Srihari, S.N. Decision Combination in Multiple Classifier Systems (1994) IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 (1)

[121]. Farrell KR (1995) Text-dependent speaker verification using data fusion. In: 1995 International Conference on Acoustics, Speech, and Signal Processing, 1995. ICASSP-95, vol 1. IEEE, pp 349–352

[122]. Farrell KR, Ramachandran RP, Sharma M, Mammone RJ (1997) Sub-word speaker verification using data fusion methods. In: Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop. IEEE, pp 531–540

[123]. Huenup´an F, Yoma NB, Molina C, Garret´on C (2008) Confidence based multiple classifier fusion in speaker verification. Pattern Recogn Lett 29(7):957–966

[124]. Xiang B, Berger T (2003) Efficient text-independent speaker verification with structural Gaussian mixture models and neural network. IEEE Trans Speech Audio Process 11(5):447–456

[125]. Dash JK, Mukhopadhyay S, Garg MK, Prabhakar N, Khandelwal N (2014) Multi-classifier framework for lung tissue classification. In: 2014 IEEE Students' Technology Symposium (TechSym). IEEE, pp 264–269

[126]. Dash JK, Mukhopadhyay S, Prabhakar N, Garg M, Khandelwal N (2013) Content-based image retrieval for interstitial lung diseases using classification confidence. In: SPIE Medical Imaging, pp. 86,702Y, International Society for Optics and Photonics

[127]. Ma L, Liu X, Song L, Liu Y, Zhou C, Zhao X, Zhao Y (2014) A new classifier fusion method based on confusion matrix and classification confidence for recognizing common ct imaging signs of lung diseases. In: SPIE Medical Imaging, pp. 90,351H–90,351H. International Society for Optics and Photonics

[128]. P. Soda, G. Iannello, and M. Vento, "A multiple expert system for classifying fluorescent intensity in antinuclear autoantibodies analysis,"Pattern Anal. Appl., vol. 12, no. 3, pp. 215–226, Sep. 2009

[129]. M. D. Santo, M. Molinara, F. Tortorella, and M. Vento, "Automatic classification of clustered microcalcifications by a multiple expert system,"Pattern Recognit., vol. 36, no. 7, pp. 1467–1477, Jul. 2003

[130]. Jain AK, Ross A, Prabhakar S (2004) An introduction to biometric recognition. IEEE Trans Circ Syst Video Technol 14(1):4–20

[131]. Maltoni D, Maio D, Jain AK, Prabhakar S (2009) Handbook of fingerprint recognition. Springer

[132]. Bahrololoum, A., & Nezamabadi-pour, H. (2017). A multi-expert based framework for automatic image annotation. Pattern Recognition, 61, 169-184

[133]. Roli F, Kittler J, Fumera G, Muntoni D (2002) An experimental comparison of classifier fusion rules for multimodal personal identity verification systems. In: Multiple Classifier Systems. Springer, pp 325–335

[134]. L.-L. Huang and A. Shimizu, “A multi-expert approach for robust face detection,” Pattern Recognit., vol. 39, no. 9, pp. 1695–1703, Sep. 2006

[135]. L. P. Cordelia, M. D. Santo, G. Percannella, C. Sansone, and M. Vento, “A multi-expert system for movie segmentation,” in Multiple Classifier Systems. London, U.K.: Springer-Verlag, 2002, pp. 304–313

[136]. De Santo, M., Percannella, G., Sansone, C., & Vento, M. (2007). Segmentation of news videos based on audio-video information. Pattern analysis and applications, 10(2), 135-145.

[137]. Tran, T. P., Tsai, P., & Jan, T. (2008, December). A Multi-expert Classification Framework with Transferable Voting for Intrusion Detection. In Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on (pp. 877-882). IEEE.

[138]. Xu L, Krzyzak A, Suen CY (1992) Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans Syst Man Cybern 22(3):418–435

[139]. Foggia, P., Sansone, C., Tortorella, F., Vento, M. Multiclassification: Reject criteria for the Bayesian combiner (1999) Pattern Recognition, 32 (8), pp. 1435-1447

[140]. De Stefano, C., Sansone, C., Vento, M. To reject or not to reject: that is the question - an answer in case of neural classifiers (2000) IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 30 (1), pp. 84-94

[141]. Kittler, J., & Alkoot, F. M. (2003). Sum versus vote fusion in multiple classifier systems. IEEE transactions on pattern analysis and machine intelligence, 25(1), 110-115.

[142]. Fumera, G., & Roli, F. (2005). A theoretical and experimental analysis of linear combiners for multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(6), 942-956.

[143]. Mi, A., & Huo, Z. (2011). Experimental comparison of six fixed classifier fusion rules. Procedia Engineering, 23, 429-433.

[144]. Kuncheva, L. I. (2002). A theoretical study on six classifier fusion strategies. IEEE Transactions on pattern analysis and machine intelligence, 24(2), 281-286.

[145]. F. Roli and G. Giacinto, Hybrid Methods in Pattern Recognition, chapter Design of multiple classifier systems,Worldwide Scientific Publishing, 2002, 199–226.

[146]. Quost B, Masson MH, Denoeux T (2011) Classifier fusion in the dempster–shafer framework using optimized t-norm based combination rules. Int J Approx Reason 52(3):353–374

[147]. Kuncheva LI, Bezdek JC, Duin RP (2001) Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recogn 34(2):299–314

[148]. Huang Y, Suen C (1993) The behavior-knowledge space method for combination of multiple classifiers. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Institute of Electrical Engineers Inc (IEEE), pp 347–347

[149]. Y. Lu, Knowledge integration in a multiple classifier system, Appl. Intell.6(2) (1996), 75–86

[150]. Canuto, A. M. D. P. (2001). Combining neural networks and fuzzy logic for applications in character recognition (Doctoral dissertation, University of Kent at Canterbury).

[151]. Z. Ghahramani and H. Kim, Bayesian classifier combination, Gatsby Technical Report, 2003.

[152]. K. Woods, W.P. Kegelmeyer Jr., K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4) (1997) 405–410

[153]. Grover, J., & Hanmandlu, M. (2017). Personal identification using the rank level fusion of finger-knuckle-prints. Pattern Recognition and Image Analysis, 27(1), 82-93.

[154]. Hou, J., Xu, E., Xia, Q., & Qi, N. M. (2015). Evaluating classifier combination in object classification. Pattern Analysis and Applications, 18(4), 799-816.

[155]. Nguyen, T. T., Nguyen, T. T. T., Pham, X. C., & Liew, A. W. C. (2016). A novel combining classifier method based on Variational Inference. Pattern Recognition, 49, 198-212

[156]. Sharma, R., Das, S., & Joshi, P. (2015, December). Rank level fusion in multibiometric systems. In Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2015 Fifth National Conference on (pp. 1-4). IEEE

[157]. Foggia, P., Saggese, A., & Vento, M. (2015). Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion. IEEE TRANSACTIONS on circuits and systems for video technology, 25(9), 1545-1556

[158]. Kuncheva, L. I. (2014). Combining Pattern Classifiers: Methods and Algorithms. John Wiley & Sons.

[159]. De Marsico, M., Nappi, M., Riccio, D., & Wechsler, H. (2016). Leveraging implicit demographic information for face recognition using a multi-expert system. Multimedia Tools and Applications, 1-29

[160]. Gehler P, Nowozin S (2009) On feature combination for multiclass object classification. In: IEEE international conference on computer vision, pp 221–228

[161]. Altmann A, Rosen-Zve M, Prosperi M, Aharoni E, Neuvirth H et al (2008) Comparison of classifier fusion methods for predicting response to anti hiv-1 therapy. PLoS One 3(10):1–9

[162]. Heerden CV, Barnard E (2009) Combining multiple classifiers for age classification. In: 20th annual symposium of the Pattern Recognition Association of South Africa, pp 59–64

[163]. http://archive.ics.uci.edu/ml/datasets.html
[164]. http://ganymed.imib.rwthaachen.de/irma/datasets_en.php
[165]. NIST: Biometric scores set (2004). [Online]. Available: www.itl.nist.gov/iad/894.03/biometricscores
[166]. P. Phillips, P. Flynn, T. Scruggs, K. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 947–954
[167]. LG 4000 iris image database." [Online]. Available: http://biometrics.idealtest.org/
[168]. Brodatz P (1966) Textures: a photographic album for artists and designers, vol 66. Dover New York University of Sourthern California, Signal and Image Processing Institute. Rotated Textures, [Online]. Available: http://sipi.usc.edu/services/database/Databaese.html
[169]. Ojala T, Maenpaa T, Pietikainen M, Viertola J, Kyllonen J, Huovinen S (2002) Outex-new framework for empirical evaluation of texture analysis algorithms. In: Proceedings 16th International Conference on Pattern Recognition, 2002, vol 1. IEEE, pp 701–706
[170]. V. Carletti, L. Del Pizzo, G. Percannella, and M. Vento. Benchmarking two algorithms for people detection from topview depth cameras. In International Conference on Pattern Recognition and Image Analysis. Springer, 2017.
[171]. V. Carletti, L. Del Pizzo, G. Percannella, and M. Vento. An efficient and effective method for people detection from top-view depth cameras. In 14th IEEE International Conference on Advanced Video and Signal based Surveillance, 2017.
[172]. Zhu, Y., & Wang, Z. (2016, October). Real-Time Abnormal Behavior Detection in Elevator. In Chinese Conference on Intelligent Visual Surveillance (pp. 154-161). Springer, Singapore

[173]. Dash, J. K., Mukhopadhyay, S., & Gupta, R. D. (2016). Multiple classifier system using classification confidence for texture classification. Multimedia Tools and Applications, 1-22

Appendice A: Progettazione di un Sistema per la Misurazione dei Flussi Antropici in Entrata e in Uscita ad un Complesso Archeologico

Di seguito si descrive un sistema di analisi video intelligente per la misurazione dei flussi antropici in entrata e in uscita ad un complesso archeologico. Tale sistema è progettato, realizzato e messo in esercizio presso il complesso archeologico di San Pietro a Corte e il palazzo Fruscione (Figura 67), siti nel Vicolo Adelberga del Comune di Salerno (SA). Il sistema rientra in un innovativo complesso di azioni rivolte all'utilizzo delle nuove tecnologie al fine di salvaguardare e valorizzare il patrimonio paesaggistico e culturale del nostro Paese attraverso soluzioni tecnologiche avanzate.

Figura 67 - Vista del complesso archeologico di San Pietro a Corte del palazzo Fruscione

Il sistema deve essere in grado di rilevare in tempo reale gli attraversamenti di persone in ingresso e in uscita ai suddetti siti, nonché di determinare istante per istante il numero di visitatori presenti all'interno di ogni struttura (aggiornamento in *real-time*).

In particolare, il sistema, fornendo in ogni istante all'operatore il numero di visitatori presenti nella struttura d'interesse, risulta essere molto utile per limitare o inibire temporaneamente l'accesso di nuovi visitatori in particolari situazioni, ad esempio in caso di sovraffollamento o per limitare l'innalzamento della temperatura interna nei periodi dell'anno più caldi. Inoltre, il sistema di conteggio deve fornire statistiche sugli accessi alla struttura al fine di consentire agli operatori di analizzare il flusso passato e pianificare al meglio le azioni future.

Esistono numerose tecnologie di mercato che consentono il conteggio delle persone in ingresso/uscita da un locale chiuso. Ai fini della realizzazione del presente progetto si adotta una tecnologia basata sull'impiego di algoritmi di analisi video intelligente in grado di elaborare in tempo reale i flussi video proveniente da telecamere di rete IP, in accordo ai vantaggi di tale soluzione rispetto a quelle tradizionali. Un altro aspetto considerato è relativo alla possibilità di elaborare il flusso video direttamente a bordo camera (soluzione *edge-side*).

Di seguito si riportano i dettagli delle singole strutture di interesse e le scelte progettuali per l'installazione delle telecamere.

Il complesso archeologico di San Pietro a Corte presenta un unico punto di accesso per i visitatori (Figura 68), le dimensioni di tale ingresso sono *130*208 cm*. Superato l'ingresso, la visita della struttura inizia su una corsia di larghezza *133 cm* (Figura 69).

Figura 68 - Punto di ingresso/uscita del complesso archeologico di San Pietro a Corte

Figura 69 - Altro vista del punto di ingresso/uscita del Complesso archeologico di San Petro a Corte

Dall'analisi dell'ingresso è stato individuato il punto d'installazione della telecamera. Vista la necessità di ridurre il rumore dovuto a cambiamenti di luminosità per effetto di variazioni meteorologiche o apertura e chiusura della porta, si è scelto il punto d'installazione a circa *4 metri* dal varco ad un altezza di *367 cm* (Figura 70).

Figura 70 - Installazione telecamera nel complesso archeologico di San Pietro a Corte

Definito il punto d'installazione, la zona da monitorare è quella riferita alla corsia larga *133 cm.* Tali dimensioni consentono di monitorare l'intera area di interesse utilizzando un unico sensore: considerando la Figura 71, con la telecamera posizionata a *366 cm* ed angolo di vista di *76°* è possibile rilevare correttamente i visitatori da un'altezza di *110 cm.*

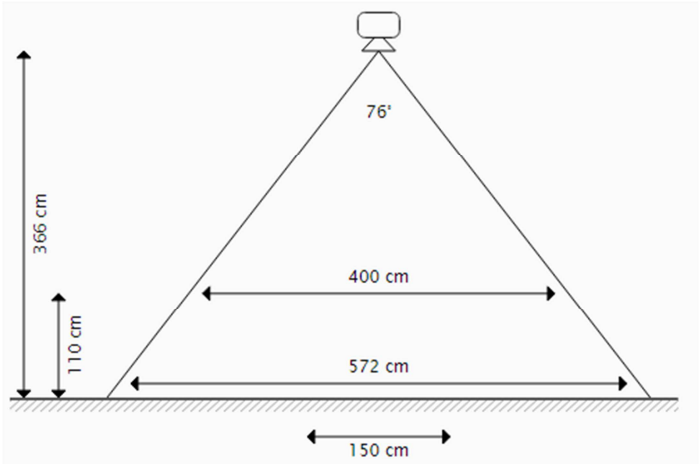


Figura 71 - Configurazione dell'installazione nel complesso di San Pietro a Corte

Il secondo edificio d'interesse è Palazzo Fruscione. La struttura presenta un unico punto di ingresso/uscita (Figura 72). Anche per questo edificio si è preferito definire un punto d'installazione non in prossimità del varco. Difatti, a pochi metri dall'ingresso è presente una seconda porta (Figura 73) ed effettuando l'installazione in prossimità di quest'ultima si riducono gli effetti dovuti all'illuminazione naturale e all'apertura/chiusura della porta. In Figura 74 si mostra il punto di installazione scelto. La porta ha un ingresso di *113x194 cm* e il punto d'installazione della telecamera è ad un altezza di *276 cm.* Con questa configurazione è possibile rilevare correttamente i visitatori da un'altezza di *110 cm* con angolo di vista di *77°* (Figura 75).

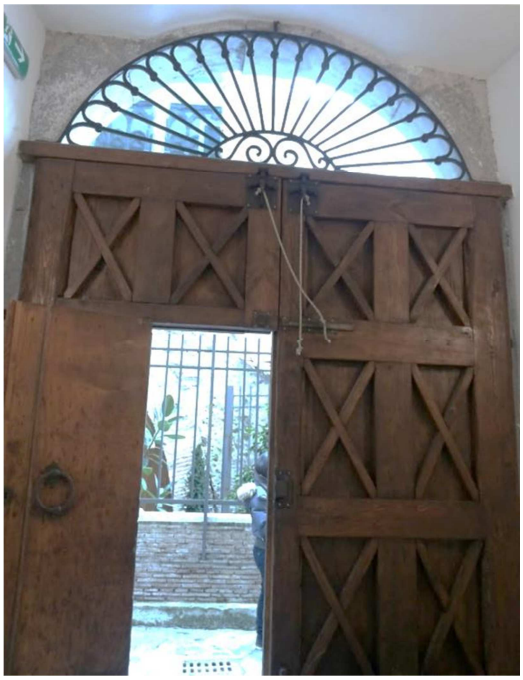


Figura 72 - Porta di ingresso/uscita del Palazzo Fruscione

Figura 73 - Porta in prossimità del punto di ingresso/uscita del Palazzo Fruscione

Figura 74 - Installazione della telecamera nel Palazzo Fruscione

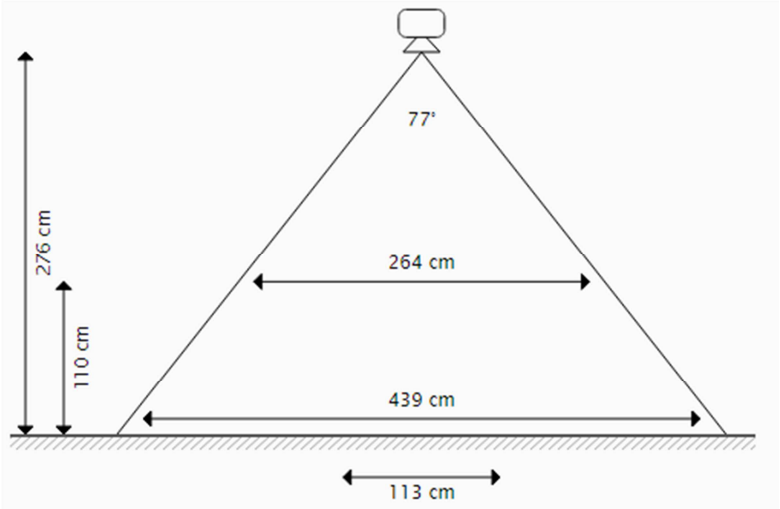


Figura 75 - Configurazione dell'installazione nel Palazzo Fruscione

Come indicato in precedenza, i sensori considerati sono telecamere IP in grado di elaborare il flusso video direttamente a bordo camera e di

fornire le statistiche sul numero di visitatori. Questa scelta consente di evitare i problemi connessi all'utilizzo di un server centrale per la raccolta ed elaborazione dei flussi, riducendo i costi di realizzazione e aumentando la scalabilità della soluzione. Ogni telecamera è connessa ad un *router* che presenta porte PoE (*Power over Ethernet*), in grado di alimentare la telecamera, e un antenna Wi-Fi per la comunicazione con altri dispositivi.

L'applicazione deve offrire, in accordo con quanto richiesto, le seguenti funzionalità:

- ✓ costi di installazione e di esercizio bassi;
- ✓ bassa intrusività nell'ambiente da monitorare;
- ✓ elaborazione del software di analisi video per il conteggio delle persone, con accuratezza (*f-index*) superiore all'80%;
- ✓ possibilità di riconfigurazione all'occorrenza i parametri di funzionamento;
- ✓ lettura dei conteggi direttamente a bordo camera e su dispositivi esterni.

L'algoritmo impiegato per il conteggio degli attraversamenti e quello descritto nel paragrafo 3.2. Inoltre, vista la necessità di un'elaborazione a bordo camera e di un'accuratezza della soluzione superiore all'*80%*, e considerati i risultati delle sperimentazioni riportati nel paragrafo 3.3, si è selezionata una telecamera con architettura MIPS che presenta le caratteristiche riportate in Tabella 30. Difatti, lavorando con una risoluzione pari a *320*240* si ottiene un'accuratezza maggiore dell'*80%* come indicato nella Tabella 9 e un *frame rate* pari a 23 come riportato in Tabella 14.

Caratteristiche	Descrizione
Sensore	CMOS Progressive Scan da 1/3,6”
Lente	1,6 mm, visualizzazione 134°
Tempo di otturazione	da 1/30500s a 2s
Compressione video	H.264
Impostazioni immagine	Compressione, colore, luminosità, nitidezza, contrasto, bilanciamento del bianco, controllo dell'esposizione, compensazione della retroilluminazione, WDR a contrasto dinamico, sovrapposizione di testo su immagini, specularità delle immagini.
Sicurezza	Protezione mediante password, filtri per indirizzi IP, HTTPS crittografia, sistema di controllo degli accessi alla rete, autenticazione digest, log accesso utenti
Analisi Video Intelligente	Conteggio visitatori in ingresso e in uscita alla struttura
Eventi generati	Ingressi e uscite con incremento dei rispettivi contatori
Alimentazione	Power over Ethernet con max 4,5 W
Connettori	Maschio RJ45 10BASE-T/100BASE-TX PoE su un cavo di rete da 2 m (6,6 piedi)
Installazione fisica	Su parete o soffitto

Tabella 30 - Caratteristiche tecniche della telecamera selezionata

In Figura 76 si riporta lo schema dell’architettura dell’intero sistema. La telecamera elabora il flusso ed invia (tramite connessione con router *PoE*) le statistiche (informazioni sui contatori con *timestamp* di ciascun evento) ad un server centrale. La telecamera fornisce una intuitiva interfaccia web per visualizzare il flusso *live* e i contatori di ingresso e uscita aggiornati in tempo reale. Si ha la possibilità di accedere a questa interfaccia in prossimità del router Wi-Fi o utilizzando una connessione Internet per un accesso dall’esterno della

struttura con dispositivi quali *smartphone*, *tablet*, *smartTV* (Figura 76).

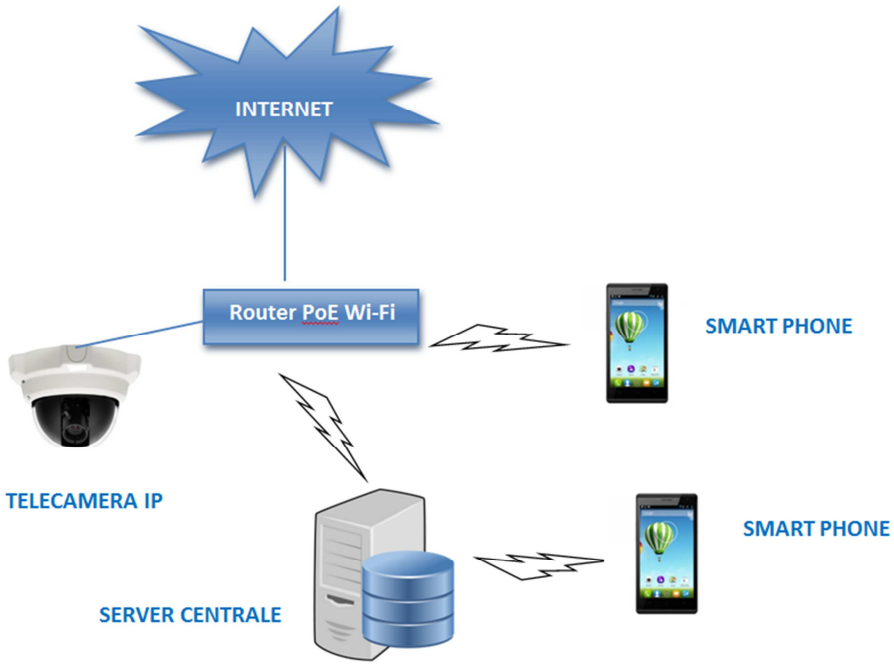


Figura 76 - Architettura del sistema di misurazione dei flussi antropici

L'architettura di riferimento per l'implementazione dell'interfaccia è quella presentata in Figura 77.

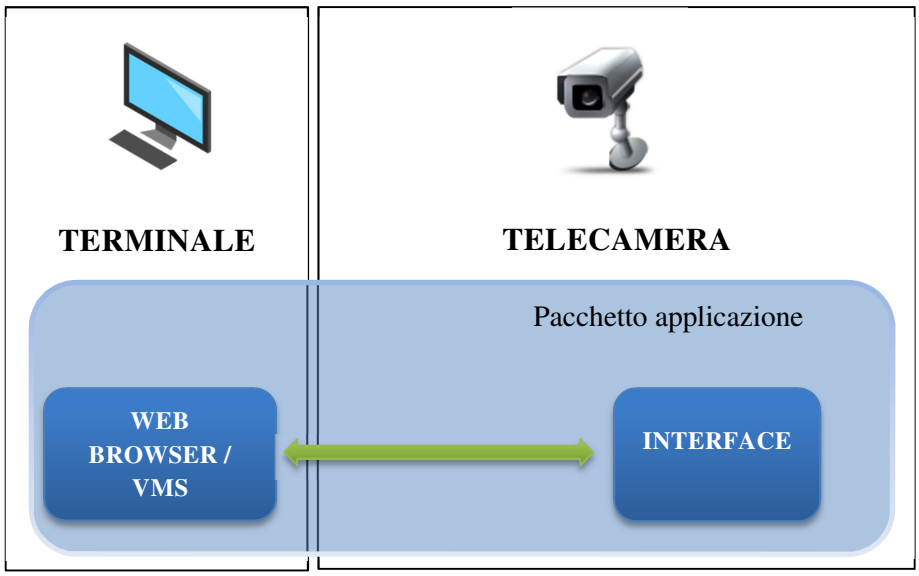


Figura 77 - Architettura dell'interfaccia web

I moduli che fanno parte dell'applicazione sono scritti in C e compilati ed installati con gli strumenti Axis (per dettagli si rimanda al Capitolo 4). In particolare, il modulo *INTERFACE* consente di generare un output formattato secondo le regole dell'HTML. Le pagine Web, incluse nel pacchetto dell'applicazione, sono sviluppate in HTML5, con l'utilizzo di comandi SSI (*Server Side Includes*) e *JavaScript*. Per realizzare la comunicazione tra il terminale e la telecamera, si utilizza l'interfaccia *net_http.h* fornita da *Axis*, che consente all'applicazione di gestire le richieste HTTP, facendo uso del web server presente nella telecamera (per dettagli si rimanda al paragrafo 5.1). In questo modo si ha la possibilità di gestire pagine dinamiche a cui associare delle funzioni di *callback* per la gestione delle richieste da parte dell'utente o delle applicazione in esecuzione sulla telecamera.

Il formato scelto per lo scambio dati tra pagine Web e interfacce CGI è quello JSON (*JavaScript Object Notation*), un formato strutturato di scambio dati, indipendente dal linguaggio di programmazione. JSON è supportato nativamente in *JavaScript*, mentre per utilizzarlo lato telecamera, quindi in linguaggio C, si utilizzano le librerie *open-source Jansson 2.6* che consentono di effettuare il *parsing* e la traduzione nei formati nativi del C.

Poiché la libreria di interfacciamento Web di *Axis* non consente di accedere alla richiesta HTTP POST completa, ma soltanto ai singoli campi, gli oggetti JSON si incapsulano in un parametro della richiesta POST; ciò non comporta problemi, in quanto gli oggetti JSON sono mantenuti in formato testuale.

Detto ciò, si è provveduto alla realizzazione di un'interfaccia per impostare i seguenti parametri dell'applicazione utili a:

- ✓ abilitare l'applicazione;
- ✓ disegnare le regioni di interesse per la rilevazione del movimento (sensori virtuali);
- ✓ definire i parametri dell'algoritmo di *background subtraction,* cioè la soglia T_h per ottenere la *foreground mask*, il coefficiente α per impostare la velocità di aggiornamento del modello del background;
- ✓ definire i parametri dell'algoritmo per il conteggio: W, H, θ_c, θ_k.

Questi parametri si gestiscono attraverso l'interfaccia *param.h* fornita da Axis. In particolare, nella pagina di configurazione si visualizza un flusso video MJPEG, sul quale si riporta il sensore linea virtuale tramite una combinazione di HTML5 e *Javascript*; le informazioni sul sensore sono passate al modulo *INTEFACE* dell'applicazione sviluppata attraverso il metodo HTTP POST. Tramite *mouse* si ha la possibilità di disegnare direttamente sul flusso video il sensore di

interesse, oppure impostare le coordinate utilizzando le caselle di testo. Si utilizzano inoltre dei comandi SSI per ottenere alcune informazioni relative alla configurazione generale della telecamera (rotazione, risoluzione, ecc.) utilizzando l'interfaccia VAPIX di Axis: API per richiamare le funzionalità della telecamera, leggere e impostare la configurazione.

In Figura 78 si presenta un estratto dell'interfaccia web che include il flusso video della telecamera su cui è riportato il sensore virtuale e i contatori di ingresso e uscita.

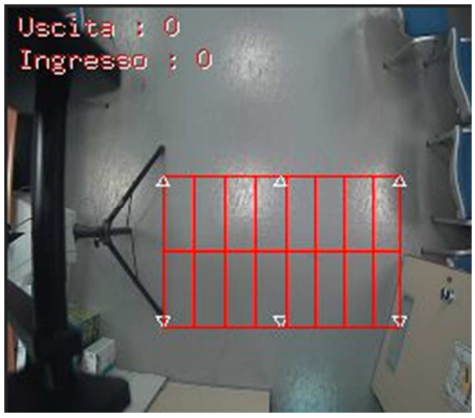


Figura 78 – Flusso video visualizzato nell'interfaccia web con contatori di ingresso/uscita

Il server centrale è connesso alla telecamera e ad altre tipologie di sensori come ad esempio quelli per rilevare la temperatura e l'umidità della sala. L'applicativo in esecuzione sul server (sviluppato da altri partner del progetto) gestisce l'archiviazione degli eventi inviati dalle telecamere e rileva condizioni di interesse come, ad esempio, il raggiungimento di una stato sfavorevole per una corretta conservazione delle opere architettoniche presenti all'interno della struttura. Tali condizioni sono segnalate all'operatore (sul videoterminale e tramite invio di email e sms), il quale può gestire la situazione mitigando il rischio (ad es. interrompendo

temporaneamente gli accessi dei visitatori). Oltre a situazioni di allerta, le statistiche sui visitatori sono rilevanti per una corretta gestione del personale e per accrescere il gradimento dei visitatori. Ad esempio, queste informazioni possono fornire un quadro generale e puntuale sulle fasce orarie e i periodi di maggiore o minore affluenza, in modo da indirizzare opportune campagne pubblicitarie e per aumentare la *customer satisfaction*.

Dai primi test effettuati sul sistema installato e configurato, si rileva che:

- ✓ l'accesso alla telecamera utilizzando dispositivi locali (uno *smartphone* connesso al router tramite Wi-Fi) avviene in pochi secondi con aggiornamento automatico della pagina riportando i valori correnti dei contatori di ingresso e uscita. Il collegamento tramite Internet viene assicurato da un ISP (Internet Service Provider) esterno;
- ✓ le prestazioni fornite dall'algoritmo nello scenario analizzato in precedenza sono compatibili con quelle riportare nelle sperimentazioni nel paragrafo 3.3 e sufficienti per gli scopi prefissati;
- ✓ le statistiche generate dalla telecamera sono inviate correttamente al server centrale. Quest'ultimo ricevute le informazioni, le elabora anche in accordo ai valori forniti dagli altri tipi di sensori per segnalare eventuali criticità. Inoltre, il server memorizza internamente i dati per successive analisi da parte dell'operatore, tramite l'utilizzo della *dashboard*, una pagina web dinamica scritta in *Javascript* che permette all'operatore di effettuare richieste sulle statistiche degli attraversamenti. In Figura 79 si presenta l'interfaccia del sistema per interrogare, visualizzare e stampare le statistiche.

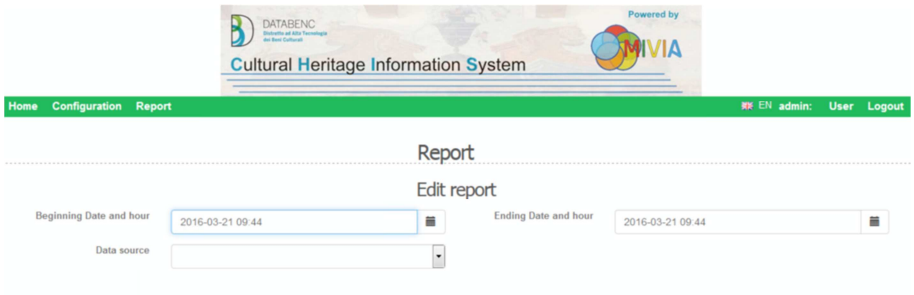


Figura 79 - Pagina web per generare richieste delle statistiche

Una soluzione così come descritta rientra in ambito IoT (*Internet of Things*), in cui oggetti intelligenti, anche di diversa tipologia, dialogano ed interagiscono tra loro per raggiungere un obiettivo comune. Difatti le telecamere IP si configurano come il presente e il futuro delle soluzioni tecnologiche presenti sul mercato (*smart building, smart city, smart car, smart health*) con indubbi vantaggi per la vita di tutti i giorni.