

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Analisi di video con visione binoculare

Relatore:
Prof. Armando Bazzani

Presentata da:
Daniele Pucci

Correlatore:
Dott. Federico Bellisardi

Anno Accademico 2021/2022

Indice

Introduzione	3
1 Strumenti utilizzati per l'analisi di immagini digitali	4
1.1 Computer vision	5
1.1.1 Visione binoculare	6
1.1.2 OpenCV	6
2 Risoluzione proposta	7
2.1 Operazioni preliminari	9
2.1.1 Calcolare gli FPS	9
2.1.2 Impostare il fattore di scala	9
2.1.3 Impostare l'origine	10
2.2 Object segmentation	12
2.2.1 Calcolo dello sfondo	12
2.2.2 Rilevamento oggetti	16
2.3 Rimozione delle ombre	17
2.3.1 Metodo cromatico	18
2.4 Individuazione della <i>feature</i> testa	20
2.4.1 Interpolazione della dinamica	21
2.5 Calcolo dell'altezza della ROI	23
2.6 Algoritmo finale di detection	26
3 Risultati	27
Conclusione	28

Sommario

Nel *TCR - Termina container Ravenna*, è importante che nel momento di scarico del container sul camion non siano presenti persone nell'area. In questo elaborato si descrive la realizzazione e il funzionamento di un sistema di allarme automatico, in grado di rilevare persone ed eventualmente interrompere la procedura di scarico del container.

Tale sistema si basa sulla tecnica della *object segmentation* tramite rimozione dello sfondo, a cui viene affiancata una classificazione e rimozione delle eventuali ombre con un metodo cromatico. Inoltre viene identificata la possibile testa di una persona e avendo a disposizione due telecamere, si mette in atto una *visione binoculare* per calcolarne l'altezza.

Infine, viene presa in considerazione anche la dinamica del sistema, per cui la classificazione di una persona si può basare sulla grandezza, altezza e velocità dell'*oggetto* individuato.

Introduzione

Nei terminal container, le gru a ponte muovono i container tra imbarcazioni, zone di stoccaggio e mezzi di trasporto come camion. La sicurezza di tali operazioni è importante e delicata al tempo stesso, dati il peso e le dimensioni dei container. In particolare, nel *TCR - Terminal Container Ravenna*, mentre una gru scarica i container sul camion, un operatore responsabile controlla che non siano presenti persone nell'area. Questo elaborato descrive la realizzazione e il funzionamento di un sistema di allarme automatico, che sia in grado di rilevare persone nell'area.

In letteratura esistono vari approcci per rilevare persone con analisi di immagini [1][2]. Alcuni di questi sono i metodi basati sulla sottrazione dello sfondo [3][4] e le più recenti *RCNN* (dall'inglese *Region-Convolutional Neural Network*) [5] che sono in grado di localizzare per poi classificare oggetti in un'immagine, tra cui le due recenti architetture di *YOLO* (*You only look once*)[6] e *SSD* (*Single Shot Multi-Box Detector*)[7].

Nella situazione portuale affrontata in questo elaborato, non è possibile posizionare le telecamere a terra e queste sono state quindi fissate sulla gru, dal Laboratorio di *City Scienze* del Dipartimento di Fisica e Astronomia (DIFA) dell'Università di Bologna in collaborazione con il porto (*TCR - Terminal Container Bologna*). Si ha così un'inquadratura da un'altezza di 22 metri, perpendicolare alla zona interessata: ciò rende impossibile l'uso di tecniche di riconoscimento di persone basate su un addestramento di rete neurale, spesso in grado di identificare le persone frontalmente ma non dall'alto. Inoltre le operazioni portuali continuano sia nelle ore serali che in diverse condizioni meteo, per cui il riconoscimento delle persone deve essere efficace anche in tali condizioni.

La risoluzione proposta si basa dunque su un semplice ma robusto algoritmo di *object detection* basato sulla sottrazione dello sfondo, che viene poi affiancato da una classificazione e rimozione di eventuali ombre tramite un metodo cromatico. Per avere un'ulteriore criterio per classificare gli *oggetti* individuati, si è deciso di includere anche una visione binoculare, che permette di ricavare le altezze degli oggetti individuati. Inoltre, considerando la dinamica del sistema si possono mettere in relazione frame consecutivi e ricavare un terzo fattore con cui classificare l'oggetto: la sua velocità di movimento.

L'elaborato è strutturato nel seguente modo: il Capitolo 1 descrive gli strumenti utilizzati per effettuare l'analisi di immagini, il Capitolo 2 espone la risoluzione proposta, il Capitolo 3 illustra i risultati ottenuti e infine ci sono le Conclusioni.

Capitolo 1

Strumenti utilizzati per l'analisi di immagini digitali

Un'*immagine digitale* è la rappresentazione numerica di un'immagine bidimensionale, composta da una matrice di punti o celle, detti pixel [8]. In un'immagine in scala di grigi, ogni pixel rappresenta l'intensità del grigio, che varia dal nero al bianco.

In un'immagine a colori invece, per ogni pixel vengono memorizzati i livelli di intensità delle *componenti* (o *canali*), che variano a seconda del modello di colore usato. Tra questi modelli vi sono:

- RGB (Red, Green, Blue). L'acronimo RGB indica l'iniziale dei tre colori primari rosso (*red*), verde (*green*) e blu (*blue*) che vengono sommati per creare il colore desiderato. (Figura 1.1a)
- HSV (Hue, Saturation, Value). I tre parametri principali sono tonalità (*hue*), saturazione (*saturation*), intensità (*value*). HSV è una geometria cilindrica: l'altezza del cilindro indica il valore dell'intensità, la distanza dall'asse la saturazione e l'angolo la tonalità (Figura 1.1b). I colori primari e le loro combinazioni lineari sono quindi disposti nel bordo estremo del cilindro con saturazione massima e valore massimo, il nero si trova alla base del cilindro con intensità nulla, mentre il bianco è disposto nell'asse del cilindro, con saturazione nulla. Per convenzione, l'angolo della tonalità parte da 0° per il rosso, passando per il verde a 120°, per il blu a 240°, per poi tornare al rosso a 360°.

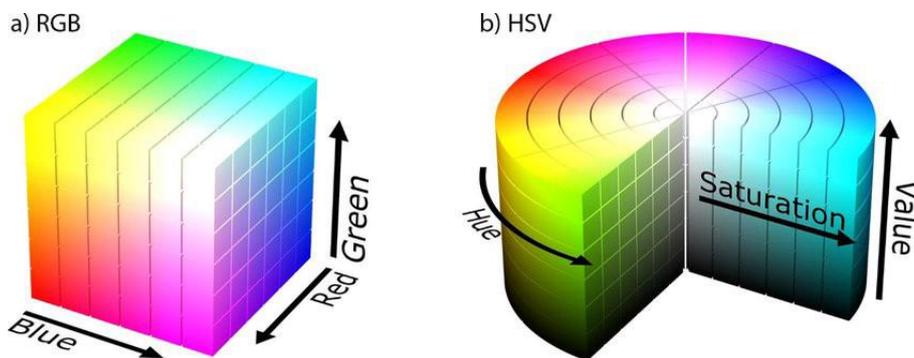


Figura 1.1: a) Rappresentazione grafica a cubo del modello RGB: ogni asse rappresenta l'intensità di uno dei tre colori primari (rosso, verde, blu). Nell'origine, vertice del cubo non visibile, l'intensità dei tre colori è zero, dunque il colore è nero. b) Rappresentazione grafica a cilindro del modello HSV: l'altezza del cilindro indica l'intensità, la distanza dall'asse la saturazione, l'angolo la tonalità (per convenzione, partendo dal rosso a 0°). (Fonte immagini: [9][10])

Un esempio di immagine digitale divisa nei tre canali RGB è visibile in figura 1.2.

Un'immagine digitale in scala di grigi è quindi rappresentata matematicamente da una matrice di numeri (spesso interi, da 0 a 255):

$$Immagine_{grigi} \equiv \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1y} \\ a_{21} & a_{22} & \dots & a_{2y} \\ \vdots & \vdots & \ddots & \vdots \\ a_{x1} & a_{x2} & \dots & a_{xy} \end{pmatrix} \quad (1.1)$$

mentre per un'immagine a colori si hanno tre matrici, una per ognuna dei tre canali. In un'immagine RGB, si avrà ad esempio una matrice per il canale rosso, una per il canale verde e una per il canale blu:

$$Immagine_{colori} \equiv \sum_{i=1}^3 \mathbf{A}_i \stackrel{(RGB)}{=} \mathbf{R} + \mathbf{G} + \mathbf{B} \quad (1.2)$$

1.1 Computer vision

Computer vision (noto in italiano anche come *visione artificiale* [11]) è l'insieme dei processi che mirano a creare una descrizione esplicita e significativa del mondo reale partendo da immagini digitali [12][13]. Vedere è inteso non solo come l'acquisizione di una fotografia, ma anche e soprattutto come l'interpretazione del contenuto di quell'area.



Figura 1.2: Immagine a colori insieme alle sue componenti R,G,B. (Fonte: [9])

1.1.1 Visione binoculare

Per *visione stereoscopica* (o *stereo vision*) si intende quella branca di visione artificiale che riguarda l'estrazione di informazioni tridimensionali a partire da immagini digitali. In particolare, la *visione binoculare* è un tipo di visione stereoscopica che utilizza due immagini digitali, simulando direttamente ciò che accade negli occhi umani, che osservano una scena da due diversi punti di vista [14].

Ovviamente, per ottenere una visione binoculare, occorre avere due video in ingresso e quindi due telecamere, che in questo caso sono state installate dal Laboratorio di *City Science* del Dipartimento di Fisica e Astronomia (DIFA) dell'Università di Bologna, in collaborazione con il *TCR - Terminal Container Ravenna*.

1.1.2 OpenCV

Il sistema di allarme è stato realizzato tramite un codice [15] in C++ che utilizza largamente OpenCV [16], una libreria open source scritta in C e C++ per ambienti Linux, Windows e Mac OS X. È stato scelto di usare OpenCV per le sue performance, per il suo focus nelle applicazioni in tempo reale e per i vari metodi di elaborazione di immagini implementati nella libreria.

Capitolo 2

Risoluzione proposta

Il rilevamento di *oggetti* e il calcolo della relativa altezza sono sviluppati in più fasi. In particolare, per calcolare l'altezza di una persona è necessario individuarne la posizione della testa rispetto a un punto di riferimento comune nelle immagini sincronizzate provenienti dalle due telecamere.

Dunque il codice si compone di più step principali. Dapprima vengono svolte alcune operazioni preliminari per determinare un corretto fattore di scala tra pixel e centimetri e tra numero di frame e secondi. Inoltre viene scelto un punto che sia l'origine del sistema di riferimento comune alle due telecamere. Viene quindi applicata la segmentazione di oggetti (*object segmentation*) basata sulla rimozione dello sfondo e vengono rimosse eventuali ombre. A questo punto si ricava la posizione della eventuale *feature* 'testa' dell'oggetto individuato e tramite una visione binoculare si ricava la sua altezza. Infine, con queste informazioni viene deciso se l'oggetto rilevato possa essere una persona o no.

Uno schema a blocchi riassuntivo del funzionamento del codice è visibile in Figura ?? e l'intero codice è disponibile su GitHub [15].

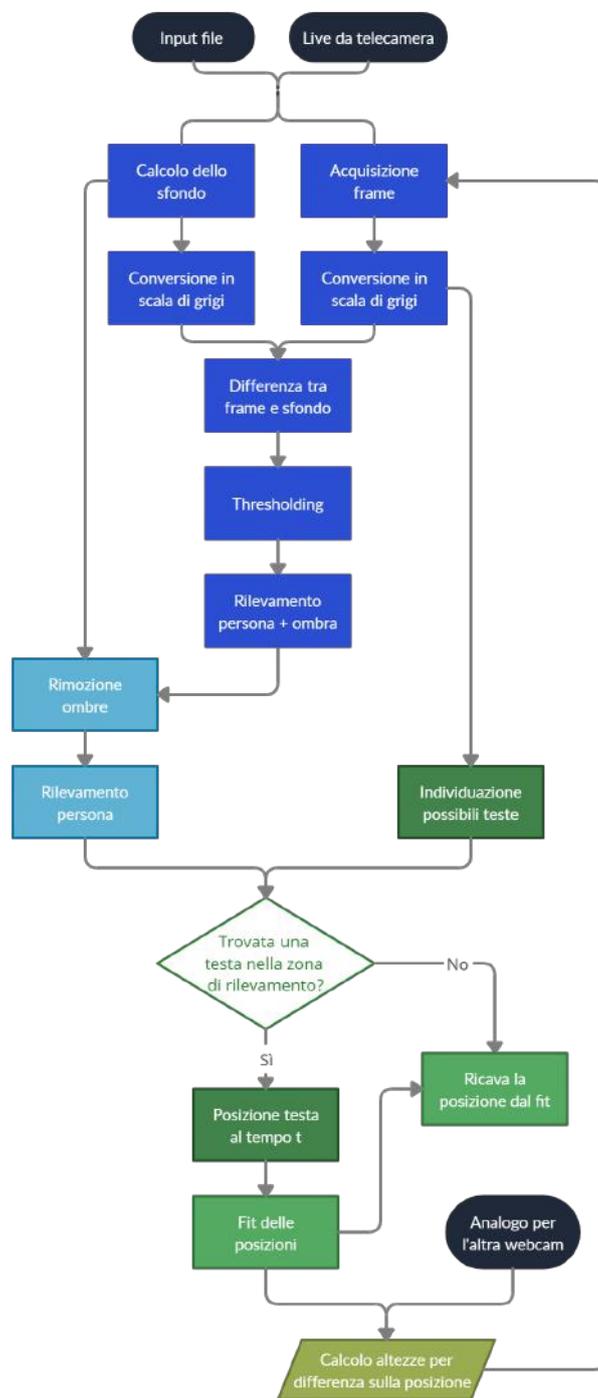


Figura 2.1: Schema riassuntivo del funzionamento dell’algoritmo implementato

2.1 Operazioni preliminari

Almeno la prima volta che il programma viene eseguito con nuove telecamere e/o con le stesse telecamere ma in una posizione diversa, occorre eseguire alcune operazioni preliminari in modo da garantire un sistema di riferimento comune alle due telecamere e determinare un corretto fattore di scala tra lunghezze e pixel e tra tempi e numero di frame.

In particolare occorrono tre operazioni, che nel codice sono implementate come tre metodi: `calculateFps`, `SetScale`, `SetOrigin`.

Ovviamente, una volta montate e impostate le telecamere, i valori ottenuti da questi tre metodi non cambiano, quindi queste operazioni possono essere eseguite una sola volta e i valori ottenuti possono essere salvati (ad esempio in un file di testo) e ricaricati ad una nuova esecuzione del programma.

2.1.1 Calcolare gli FPS

Al fine di avere una corretta conversione tra numero di frame e tempi, occorre conoscere quanto rapidamente ogni telecamera acquisisce un frame. Il numero dei frame acquisiti in un secondo viene detto *fps* (dall'inglese *frames per second*) e rappresenta quindi un diretto fattore di conversione numero di frame \leftrightarrow tempo.

Qualora il valore degli fps di ogni telecamera (nel caso di telecamere non identiche può essere diverso tra le due) non fosse accessibile tramite il metodo implementato in OpenCV `VideoCapture::get(CAP_PROP_FPS)`, questo valore viene ricavato semplicemente acquisendo alcuni frame (~ 60) e misurando il tempo impiegato per acquisirli, da cui si trova il valore di fps tramite un semplice rapporto.

2.1.2 Impostare il fattore di scala

Il codice implementato contiene un metodo che consente di ricavare il fattore di scala pixel \leftrightarrow centimetri. Come si può vedere in figura 2.2, questo fa selezionare all'utente due punti in un frame casuale del video (o nel frame corrente in caso di acquisizione in tempo reale da una telecamera) per poi chiedere la distanza in centimetri tra gli stessi. Calcolando la distanza (in numero di pixel) tra i due punti selezionati si può ottenere direttamente un fattore di scala tra pixel e centimetri come un semplice rapporto. Nel caso in esame, è stata usata una sbarra come riferimento, essendo nota la sua lunghezza (90cm).

Come per gli fps, anche in questo caso il valore di questo fattore di scala può essere diverso dalle due telecamere, qualora queste non siano identiche o abbiano un diverso valore di zoom.

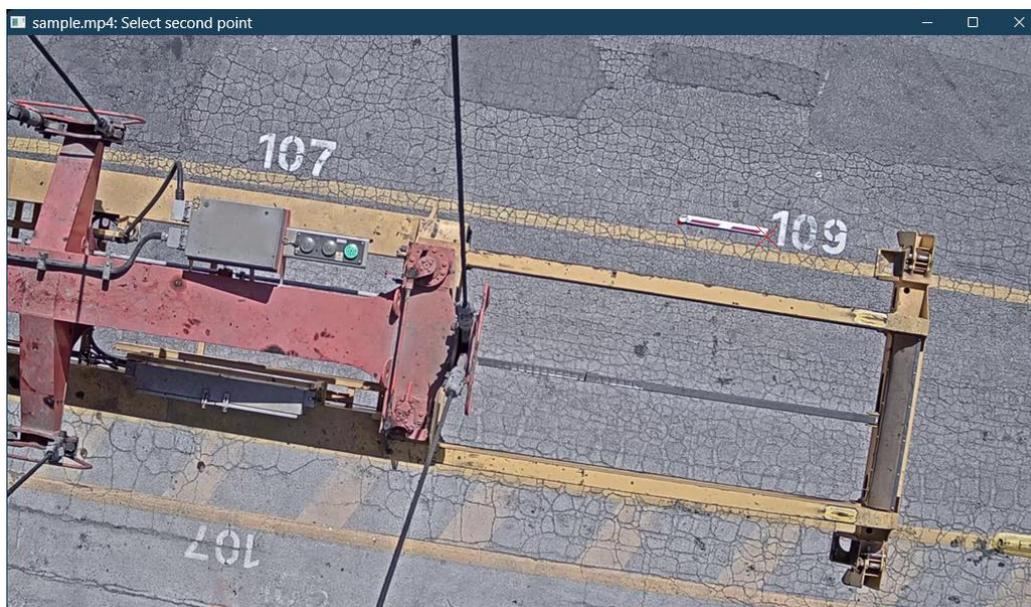


Figura 2.2: Selezione del secondo punto per impostare il fattore di scala. Il primo punto già fissato è contrassegnato con un piccolo cerchio rosso, il secondo punto con una sottile X rossa. I punti vengono scelti agli estremi della sbarra posta a terra, che in questo caso viene usata come riferimento, essendo nota la sua lunghezza.

2.1.3 Impostare l'origine

Analogamente a quanto accade per il fattore di scala, il codice ha un metodo anche per scegliere un punto che sia l'origine del sistema di riferimento. L'utente dovrà quindi selezionare lo stesso punto nelle due telecamere, in modo da poter ottenere le coordinate di un punto rispetto ad un'origine comune (Figura 2.3).

In teoria, andrebbe determinato anche l'eventuale angolo di rotazione tra le due telecamere, necessario per ottenere le coordinate assolute di un punto con cui calcolare l'altezza di un oggetto (vedi paragrafo 2.5). Tuttavia, lavorando con la distanza di un punto rispetto all'origine (che è un invariante per rotazioni), anziché con le sue coordinate, non è necessario conoscere l'angolo di rotazione tra le due telecamere. Questo approccio però si può applicare solamente qualora le coordinate dei punti siano tutte positive (o comunque, siano concordi), dunque è utile impostare l'origine in un angolo (convenzionalmente in alto a sinistra).

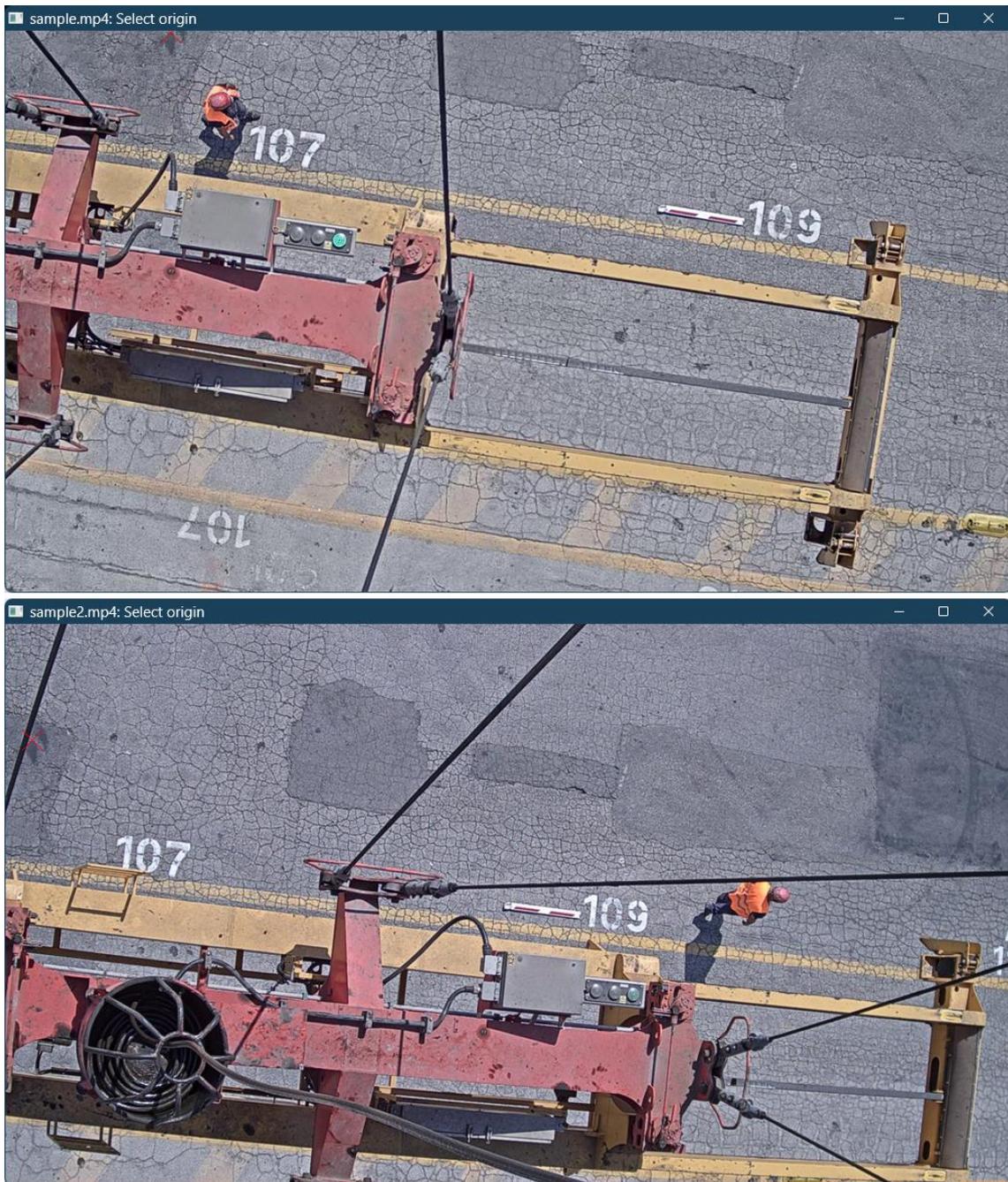


Figura 2.3: Selezione del punto di origine comune alle due telecamere. L'origine viene fissata in un punto facilmente identificabile (in questo caso all'estremo di una macchia di asfalto più scura) e il più possibile in un angolo, in modo che le coordinate abbiano tutte lo stesso segno, così da poter utilizzare le distanze dall'origine (e non le coordinate) nel calcolo delle altezze.

2.2 Object segmentation

In generale per segmentazione di un'immagine si intende il processo di partizione di un'immagine in regioni significative. Viene utilizzata per ottenere una rappresentazione più compatta o per estrarre degli oggetti dall'analisi di immagini [13].

Esistono almeno quattro diversi approcci per segmentare un'immagine [17] (sottrazione dello sfondo, differenza temporale, metodi statistici e *flussi ottici*); nel codice elaborato è stato implementato il primo perché più veloce, robusto e di facile implementazione.

La sottrazione dello sfondo è la classe di metodi più comunemente utilizzata [18][17] per rilevare regioni in movimento in un'immagine. Per *sfondo* (*background*) di una scena si intende tutto ciò che è statico nel tempo, mentre per *primo piano* (*foreground*) si intende ciò che si muove sopra lo sfondo. Se ad esempio la scena è composta da una persona che cammina in strada, il primo piano è dato dalla persona, lo sfondo è dato da tutto il resto (strada, cielo ecc...).

La segmentazione di immagini tramite sottrazione di sfondo consiste essenzialmente nel ricavare lo sfondo tramite un modello per poi ottenere il primo piano di un frame tramite sottrazione¹ di quest'ultimo dallo sfondo. È un metodo semplice e leggero a livello computazionale [17], quindi perfetto per applicazioni in tempo reale.

2.2.1 Calcolo dello sfondo

Esistono vari metodi per ricavare lo sfondo partendo da un video o da una serie di immagini (ad esempio: [19][20][21]). Nel corso dello sviluppo del codice ne sono stati implementati due.

Il primo metodo consiste nel prendere alcuni (~ 50) frame casuali dal video, convertirli in scala di grigi e calcolare ogni pixel dell'immagine di sfondo come mediana delle intensità dello stesso pixel nei diversi frame:

$$s_{xy} = Me(f_{xy}^i) \quad (2.1)$$

dove $Me(x^i)$ indica la mediana di una serie di valori generici x^i ; s_{xy} e f_{xy}^i sono rispettivamente l'intensità del pixel nella posizione (x, y) dello sfondo e del frame casuale i -esimo.

L'immagine di sfondo ottenuta con questo metodo è visibile in figura 2.4.

Il metodo è applicabile anche per ottenere immagini di sfondo a colori, dividendo ognuno dei frame nei tre canali RGB e ricavando così i tre canali dello sfondo, che sommati formeranno l'immagine di sfondo a colori (Figura 2.5).

¹Ricordando che le immagini digitali sono in realtà matrici con i pixel come elementi di matrice, qui e in tutto l'elaborato quando si parla di operazioni tra immagini si intendono operazioni tra matrici, quindi pixel per pixel. In particolare per la sottrazione, si dà per scontato che sia assoluta, ovvero date due immagini a_{xy} , b_{xy} , per loro differenza si intende $|a_{xy} - b_{xy}|$



Figura 2.4: Modello dello sfondo in scala di grigi. Il valore di ogni pixel (x, y) dell'immagine è ottenuto calcolando la mediana del valore dello stesso pixel (x, y) in 50 frame casuali del video.

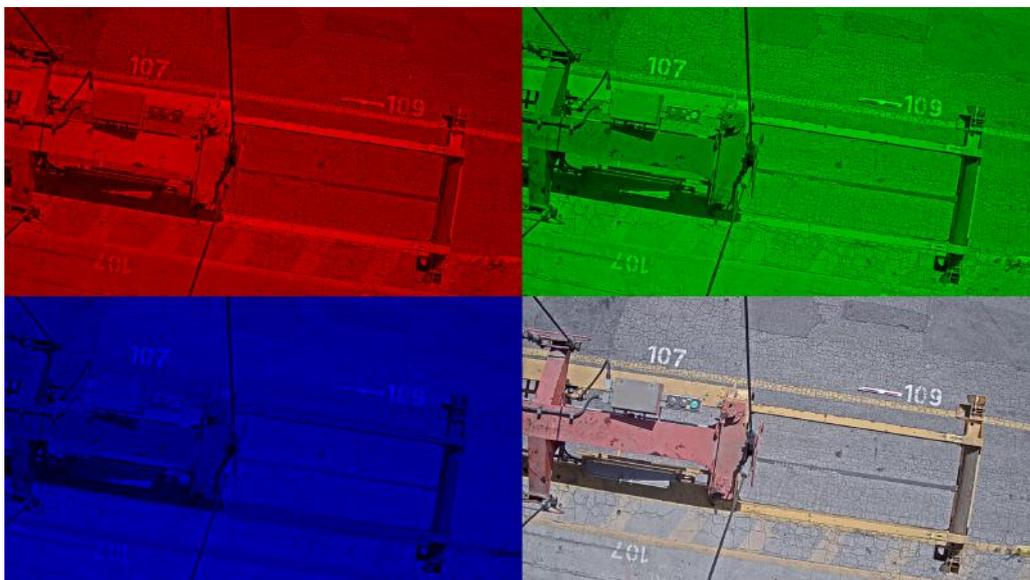


Figura 2.5: I tre canali R,G,B del modello di sfondo e lo sfondo a colori ottenuto sommandoli tra loro. Ognuno dei tre canali dello sfondo è stato ottenuto calcolando pixel per pixel la mediana delle intensità (relative allo stesso canale) di 50 frame casuali del video.

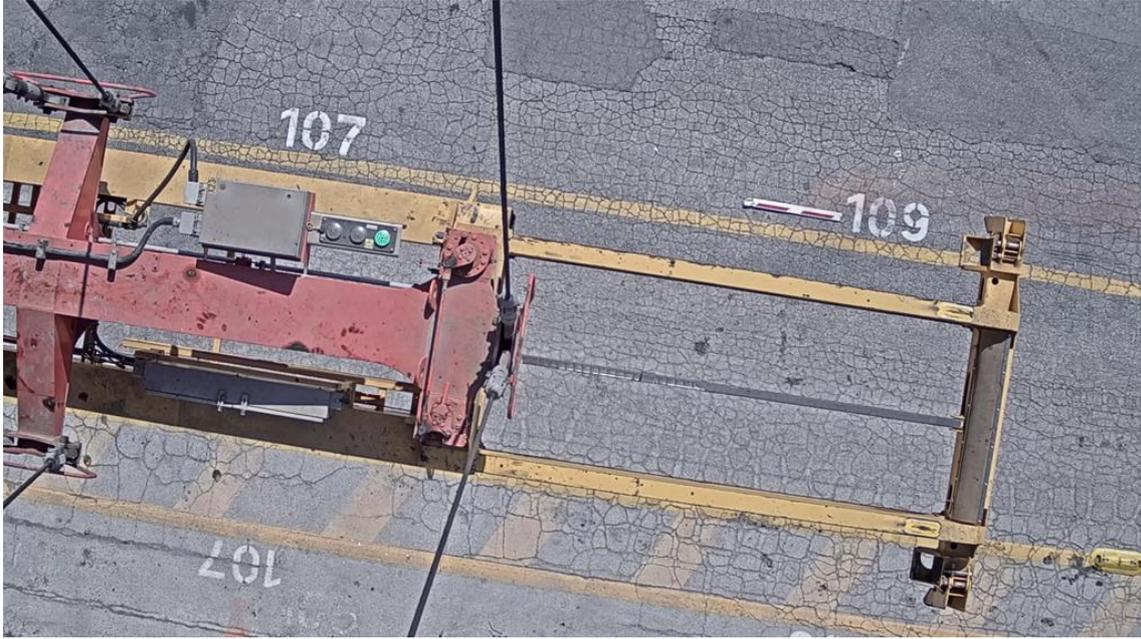


Figura 2.6: Modello di sfondo calcolato per iterazione: la nuova immagine di sfondo è data dalla somma pesata tra lo sfondo attuale e il frame corrente con un peso $\alpha = 0.005$. In questo caso si può vedere una leggera sfumatura arancione nella zona vicino alla sbarra, che comunque non ha effetti nell'identificazione di oggetti.

Il secondo metodo invece parte da un'immagine di sfondo inizialmente vuota (nera). Ogni frame acquisito viene sommato all'immagine di sfondo con un certo peso $\alpha < 1$, per formare una nuova immagine di sfondo:

$$\mathbf{S}_{i+1} = \mathbf{S}_i + \alpha \mathbf{F}_i \quad \text{con } \mathbf{S}_0 = \mathbf{0} \quad (2.2)$$

dove \mathbf{S}_i e \mathbf{F}_i indicano rispettivamente l'immagine (matrice) di sfondo e del frame acquisito nello step temporale discreto i .

Iterando il processo un numero sufficiente di volte, si ottiene il modello visibile in figura 2.6, in cui si può notare una leggera sfumatura arancione, che comunque non ha effetti significativi nel rilevamento di oggetti.

Il parametro α determina quanto lo sfondo è sensibile a cambiamenti: come si può vedere in figura 2.7, più grande è il peso con cui il frame acquisito viene sommato allo sfondo, più lo sfondo è dipendente dal frame stesso e varia velocemente con questo, ma necessita di meno frame (e quindi meno tempo) per essere calcolato.

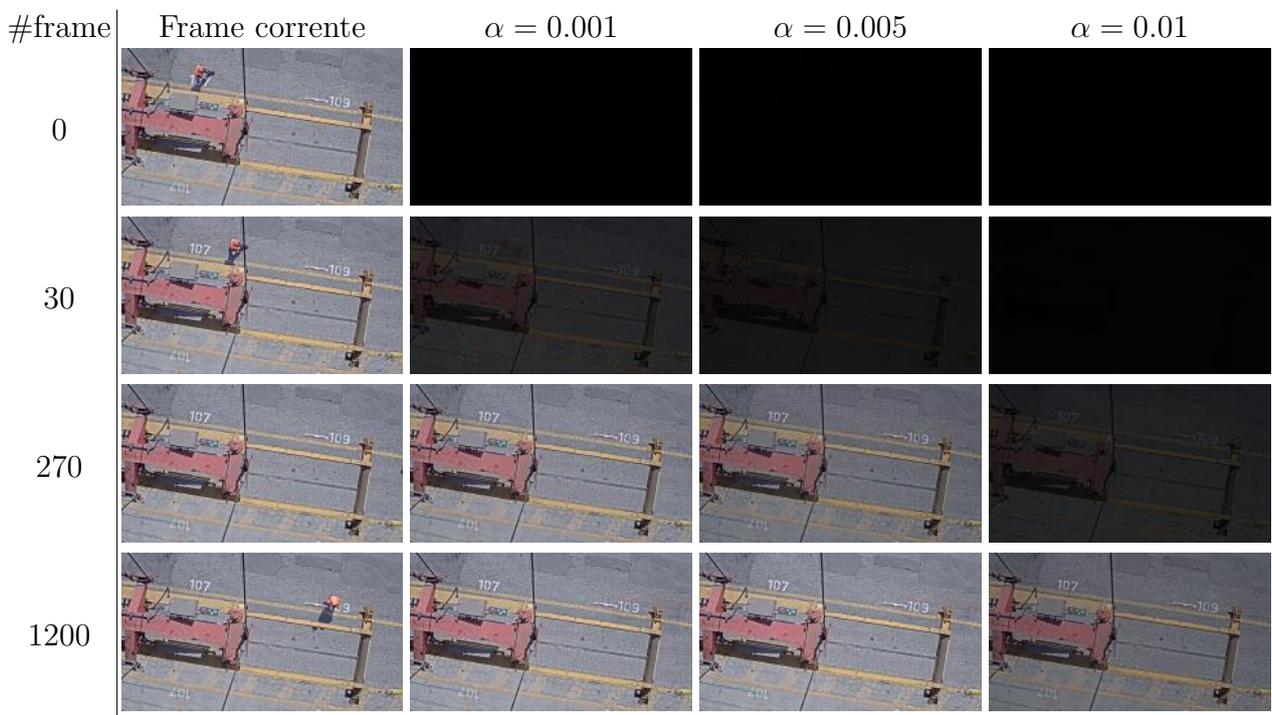


Figura 2.7: Confronto tra sfondi ottenuti con un diverso numero di frame acquisiti e un diverso valore del peso α . Si può notare come per un più grande valore di α , lo sfondo sia più robusto e meno sensibile ai cambiamenti (nell'ultima riga si vede una leggera ombra arancione per α piccoli), ma impiega anche più frame, e quindi più tempo, per essere calcolato.

Il secondo modello, già implementato nella libreria OpenCV tramite la funzione `accumulateWeighted`, è quello che è stato utilizzato nell'elaborato, perché offre la possibilità di ottenere uno sfondo dinamico, che si aggiorni in tempo reale al variare della scena. Ciò è essenziale nell'applicazione portuale su cui questo elaborato si è concentrato: data la diversa condizione di luce a diversi orari del giorno, non è possibile ricavare un modello di sfondo una sola volta.

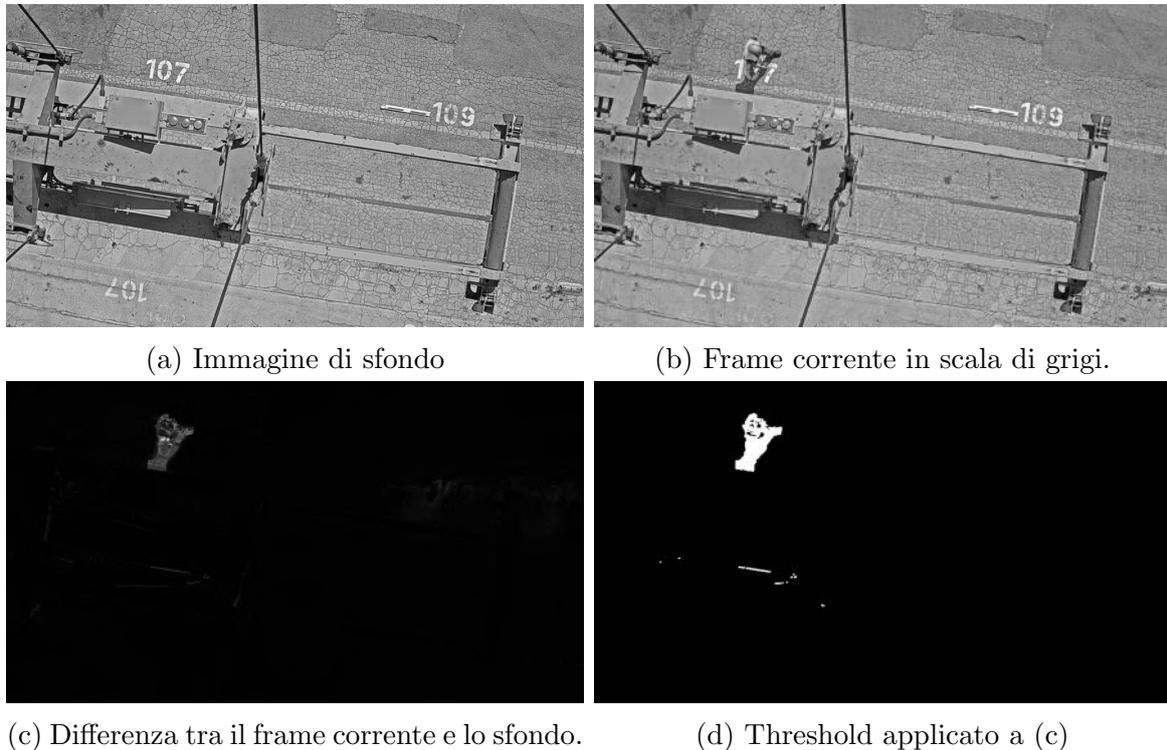


Figura 2.8: Prima fase: rilevamento oggetti per differenza del frame corrente con lo sfondo. Si può notare che l'*oggetto* rilevato in questo modo include anche l'ombra della persona.

2.2.2 Rilevamento oggetti

Una volta ricavato lo sfondo e convertito in scala di grigi (figura 2.8a), si acquisisce un frame del video in scala di grigi (2.8b), che viene sottratto dall'immagine di sfondo (2.8c). All'immagine differenza viene poi applicato un *threshold*², producendo così un'immagine *binaria* (2.8d).

Vengono poi rilevati i contorni delle aree bianche dell'immagine binaria, visibili in blu in figura 2.9. Eventuali piccoli rumori possono essere rimossi considerando l'area delle regioni e ingorandole quelle la cui area è inferiore ad un valore di soglia prefissato A_{min} .

A questo punto è stata compiuta la segmentazione dell'immagine così come descritto in [17] e implementato in [22].

²Il *threshold* è un semplice metodo di elaborazione di immagini che partendo da un'immagine in scala di grigi produce un'immagine binaria[13]. Il metodo di *threshold* più semplice consiste nel sostituire un pixel nero o bianco al posto di ogni pixel, a seconda che il valore di questo sia rispettivamente minore o maggiore di un certo valore di soglia τ .

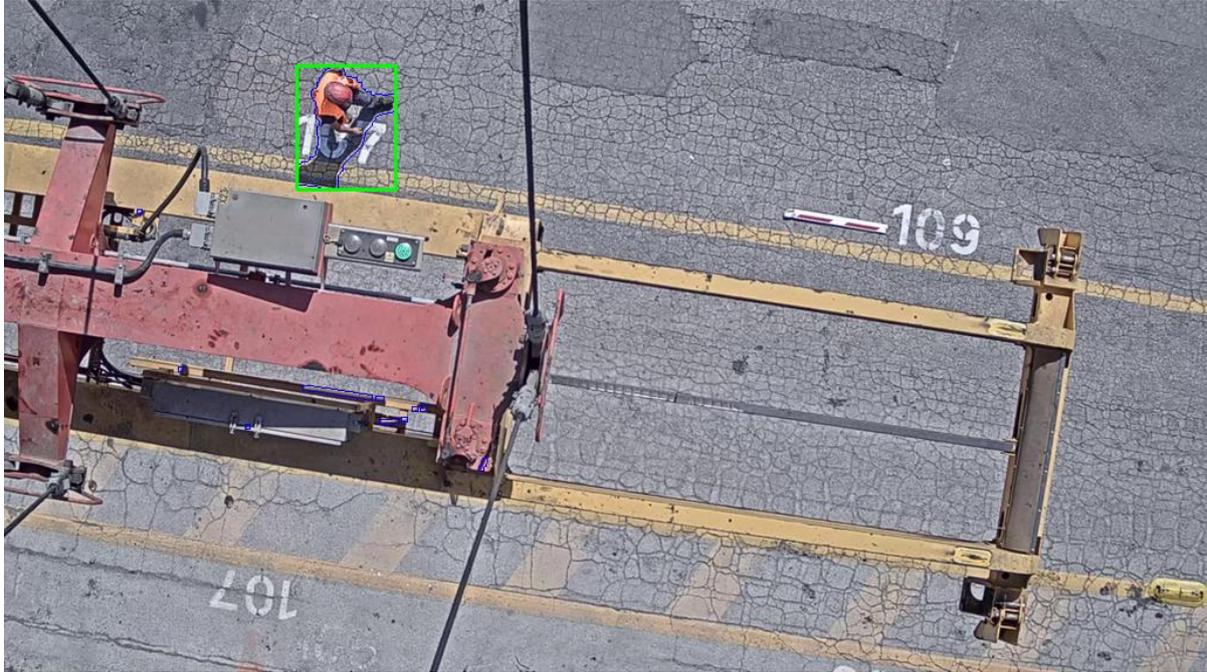


Figura 2.9: Contorni dell'immagine di threshold 2.8d riportati nel frame originale. I piccoli rumori sono rimossi considerando solamente le regioni la cui area è maggiore di una certa soglia A_{min} .

2.3 Rimozione delle ombre

La segmentazione di immagini basata sulla rimozione dello sfondo, identifica gli oggetti in modo efficiente e robusto, ma non prende in considerazione le ombre, che vengono considerate come oggetti in movimento [17].

Questo è un problema in quanto la decisione finale si basa sulla grandezza dell'oggetto rilevato e la sua altezza, valori che possono cambiare anche di molto qualora l'ombra risulti di dimensioni comparabili con l'oggetto stesso.

Esistono tuttavia vari metodi per rilevare le ombre, molti dei quali sono elencati in [23]. I metodi si possono basare su diverse caratteristiche, tra cui:

- Intensità. La più semplice assunzione che può essere fatta sull'ombra è che le regioni in ombra diventano più scure, quindi l'intensità dei pixel diminuisce [24].
- Cromaticità. I modelli che usano questo metodo si basano sull'ipotesi che le regioni in ombra diventano più scure, ma mantengono la loro *cromaticità*, che è una misura del colore indipendente dall'intensità. Per esempio un pixel verde in ombra diventerà verde scuro: un colore più scuro ma con la stessa cromaticità (verde).

- Geometria. In teoria le dimensioni, l'orientazione, e la forma dell'ombra possono essere calcolate, se si ha la giusta conoscenza della sorgente di illuminazione, della forma dell'oggetto e del terreno su cui l'ombra è proiettata.

Nella situazione portuale analizzata in questo elaborato, non si sono potuti utilizzare metodi geometrici, che richiedono una conoscenza accurata delle fonti di luce. Infatti di giorno la posizione del Sole non è fissa e le ombre si muovono quindi nel tempo, mentre nelle ore serali ci sono fonti di luce artificiali anche non fisse, come ad esempio i fanali dei camion.

Pertanto è stato implementato un metodo basato sulla cromaticità, che oltre a non richiedere conoscenze a priori sugli oggetti né sulla fonte di luce, risulta anche il più veloce in termini di esecuzione [23], anche rispetto ad altri metodi più complessi qui non citati.

2.3.1 Metodo cromatico

Nei metodi cromatici, il fattore più importante è scegliere un modello di colore che separi l'intensità dalla cromaticità. Questo non accade nel modello RGB, ma modelli come l'HSV risultano ottimi per tale scopo [25]. Il metodo cromatico implementato in questo caso è quello proposto da Cucchiara et al. [26], largamente usato in casi di sorveglianza e ben descritto in [23].

Una volta convertito un frame in HSV, le ombre degli oggetti avranno intensità (V) inferiore rispetto allo sfondo, la stessa tonalità (H) e spesso una inferiore saturazione (S) [26][23]. Per cui un pixel di un frame f_{xy} è considerato parte di un'ombra se le seguenti tre condizioni sono soddisfatte:

$$\beta_1 \leq f_{xy}^V / s_{xy}^V \leq \beta_2 \quad f_{xy}^S / s_{xy}^S \leq \tau_S \quad |f_{xy}^H / s_{xy}^H| \leq \tau_H \quad (2.3)$$

dove f_{xy}^C indica la componente C del pixel (x, y) del frame \mathbf{F} , mentre $\beta_1, \beta_2, \tau_S, \tau_H$ sono parametri di threshold ottimizzati empiricamente.

In figura 2.10a sono evidenziati i pixel classificati come ombra, mentre in figura 2.10b e 2.11 si può vedere l'oggetto rilevato privo della sua ombra (cioè la sottrazione tra 2.8d e 2.10a).



(a) Pixel dell'*oggetto* classificati come ombra

(b) *Oggetto* privato della sua ombra

Figura 2.10: Seconda fase: rimozione delle ombre dagli oggetti rilevati.

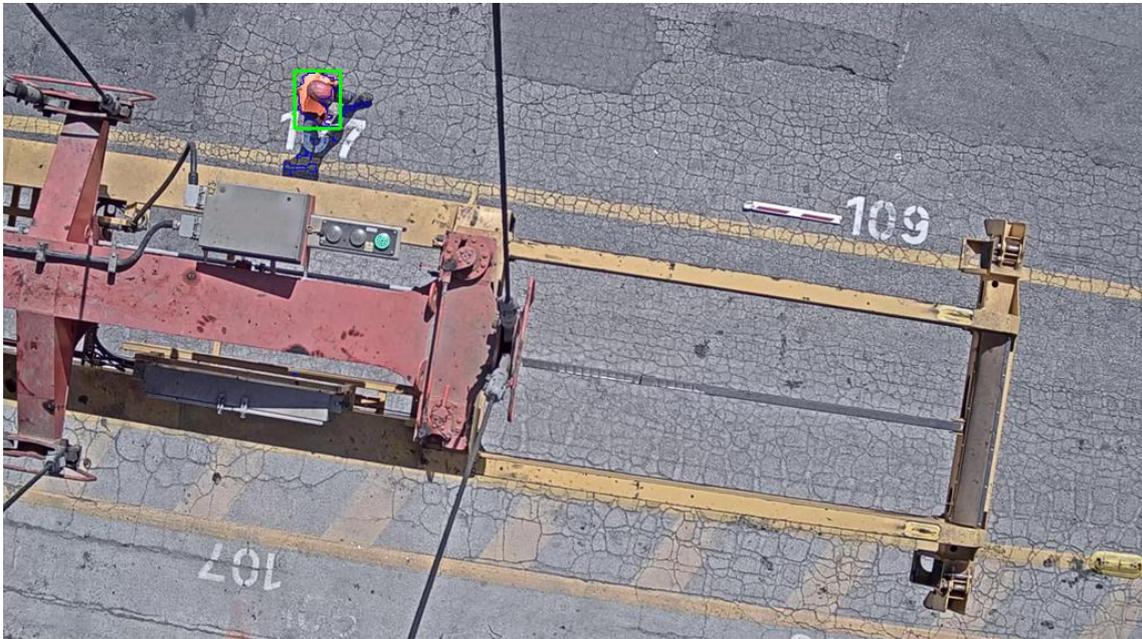


Figura 2.11: Contorno dell'*oggetto* senza ombra 2.10b riportato nel frame originale.

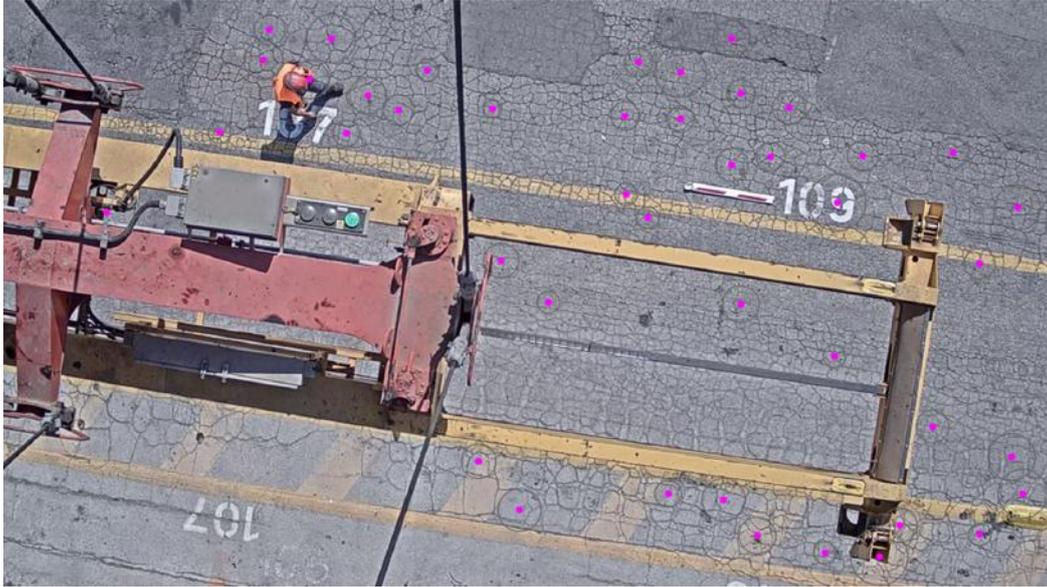


Figura 2.12: Individuazione di zone circolari tramite il metodo della trasformata di Hough applicato a tutta l'immagine a titolo di esempio. Sono visualizzati solo i cerchi con raggio compreso tra 8 e 30 pixel.

2.4 Individuazione della *feature* testa

Come anticipato, per avere un ulteriore criterio oltre alla dimensione dell'oggetto per determinare se questo possa essere o no una persona, se ne vuole determinare l'altezza. Per farlo, è necessario individuare il punto più alto dell'oggetto rilevato: nel caso di una persona, la testa.

L'algoritmo che individua la testa si basa sul presupposto che questa sia tonda e usa la tecnica della *trasformata di Hough* per estrarre cerchi dall'immagine. La trasformata di Hough è una tecnica di estrazione di linee da un'immagine digitale brevettata nel 1962 da P. Hough [27][28] e successivamente generalizzata da R.Duda e P.Hart nel 1972 per forme arbitrarie come cerchi o ellissi [29]. La tecnica della trasformata di Hough è implementata in OpenCV [16] sotto il nome di `HoughCircles` e richiede due parametri: r_{min} e r_{max} , rispettivamente il minimo e il massimo raggio del cerchio trovato.

Il risultato dell'applicazione del metodo `HoughCircles` in un frame di esempio è visibile in 2.12. In questo caso il metodo è stato applicato in tutta l'immagine a titolo di esempio, ma nel codice finale viene applicato solo nella regione interessata (*ROI*, dall'inglese *region of interest*), in verde in figura 2.13.

Si può notare come spesso rilevare le forme circolari nella ROI permette di ottenere in buona sostanza la posizione della testa.



Figura 2.13: Zona circolare individuata nella ROI rettangolare della persona determinata in 2.11. Questa approssima discretamente la posizione della testa.

2.4.1 Interpolazione della dinamica

Il rilevamento della testa tramite riconoscimento di cerchi non funziona sempre in modo perfetto: in alcuni frame viene rilevata una testa in una posizione sbagliata, in altri non viene trovata affatto, mentre in altri ancora ne vengono trovate due. Per questo motivo si è scelto di tenere in considerazione la dinamica del sistema, per mettere in relazione più frame consecutivi.

Ipotizzando che in un tempo sufficientemente breve una persona si muova di moto rettilineo uniforme con velocità $\vec{v} = (v_x, v_y)$, possono essere ricavate tre relazioni lineari ($y(x)$, $x(t)$, $y(t)$) e per ciascuna di queste si può ricavare la pendenza e l'ordinata all'origine della retta effettuando un fit lineare:

$$\begin{aligned}
 y = a + bx &\implies b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} & a = \bar{y} - b\bar{x} \\
 x = x_0 + v_x t &\implies v_x = \frac{\sum_{i=1}^n (t_i - \bar{t})(x_i - \bar{x})}{\sum_{i=1}^n (t_i - \bar{t})^2} & x_0 = \bar{x} - v_x \bar{t} \\
 y = y_0 + v_y t &\implies v_y = \frac{\sum_{i=1}^n (t_i - \bar{t})(y_i - \bar{y})}{\sum_{i=1}^n (t_i - \bar{t})^2} & y_0 = \bar{y} - v_y \bar{t}
 \end{aligned} \tag{2.4}$$

dove con una linea sopra una grandezza (ad esempio \bar{x}) si indica la media di quella grandezza.

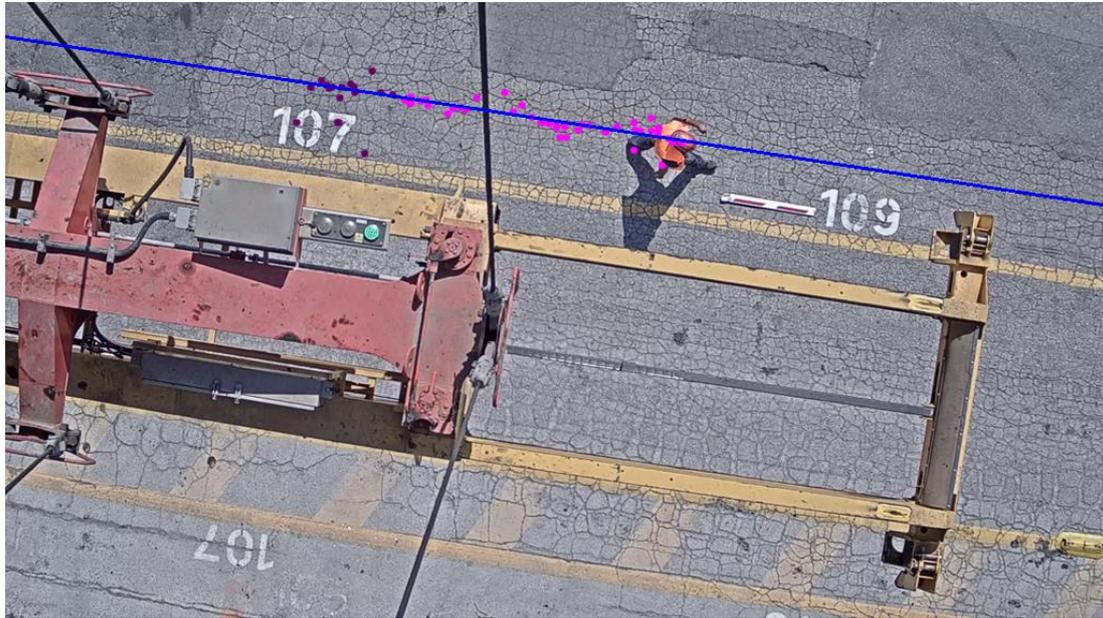


Figura 2.14: Esito del fit sulla traiettoria ($y(x)$) riportato sul frame corrente. In blu la linea di fit, in fucsia i punti usati per calcolare il fit. I punti più scuri sono stati trovati più di 3 secondi prima del frame corrente e vengono pertanto ignorati nel calcolo del fit.

Nel caso in esame, che mira a ricavare la posizione della testa di una persona, (x_i, y_i) rappresenta la posizione (in pixel) della testa trovata con il metodo della trasformata di Hough, mentre t_i indica il tempo (in numero di frame dall'inizio del video) nel quale questa è stata trovata.

Come si è detto, l'ipotesi di moto rettilineo uniforme può essere valida al massimo per qualche secondo: una persona potrebbe fermarsi, girare o addirittura tornare indietro. Per ovviare a questo problema, si è scelto quindi di considerare nel calcolo del fit solamente i punti più recenti, trovati non più di 3 secondi fa.

Nella figura 2.14 è visibile l'esito del fit sulla traiettoria riportato direttamente sul frame corrente: si noti l'ottima accuratezza del fit con la posizione della testa

Una volta effettuati i fit, la posizione della testa (x, y) in un istante t può essere ricavata dalle equazioni del moto:

$$x = x_0 + v_x t \quad y = y_0 + v_y t \quad (2.5)$$

Il risultato di questo calcolo è visibile come un punto verde nel frame di esempio in figura 2.15.

Operando le giuste conversioni tra numero di frame e secondi e tra pixel e centimetri, si ottiene la posizione della testa in un sistema di riferimento reale e comune tra le due telecamere, con cui è possibile calcolare l'altezza.

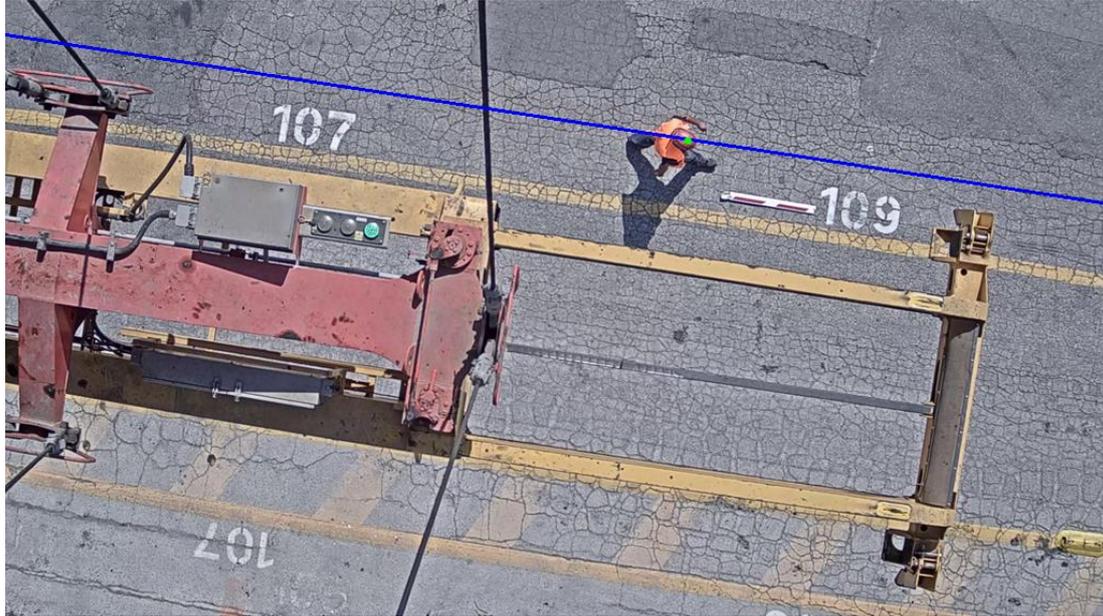


Figura 2.15: Terza fase: calcolo della posizione della testa nel frame corrente dal risultato del fit (punto verde). Si noti l'ottima accuratezza.

2.5 Calcolo dell'altezza della ROI

Una volta individuata la posizione della testa rispetto ad un sistema di riferimento comune alle due telecamere, il calcolo dell'altezza della persona è un processo di semplice trigonometria. Come si può capire dallo schema in figura 2.16, conoscendo la distanza d tra le due telecamere e la loro altezza da terra H , l'altezza h di un punto può essere ricavata da una semplice proporzione. Indicando con $\vec{x}_1 = (x_1, y_1)$ e $\vec{x}_2 = (x_2, y_2)$ la posizione della testa per le telecamere in un medesimo istante di tempo si ha:

$$\frac{d}{H - h} = \frac{|\vec{x}_1 - \vec{x}_2|}{h} \quad (2.6)$$

Da cui:

$$h = \frac{|\vec{x}_1 - \vec{x}_2| H}{|\vec{x}_1 - \vec{x}_2| + d} \quad (2.7)$$

Come già anticipato nel paragrafo 2.1.3, la formula 2.7 può essere applicata solo se si conosce l'eventuale angolo di rotazione tra le due telecamere, così da avere un sistema di riferimento correttamente orientato e quindi comune a entrambe. Il problema può essere tuttavia aggirato se si considerano le distanze di un punto dall'origine invece che le sue coordinate. Per cui, indicando con r_1 e r_2 la distanza della testa dall'origine nelle due

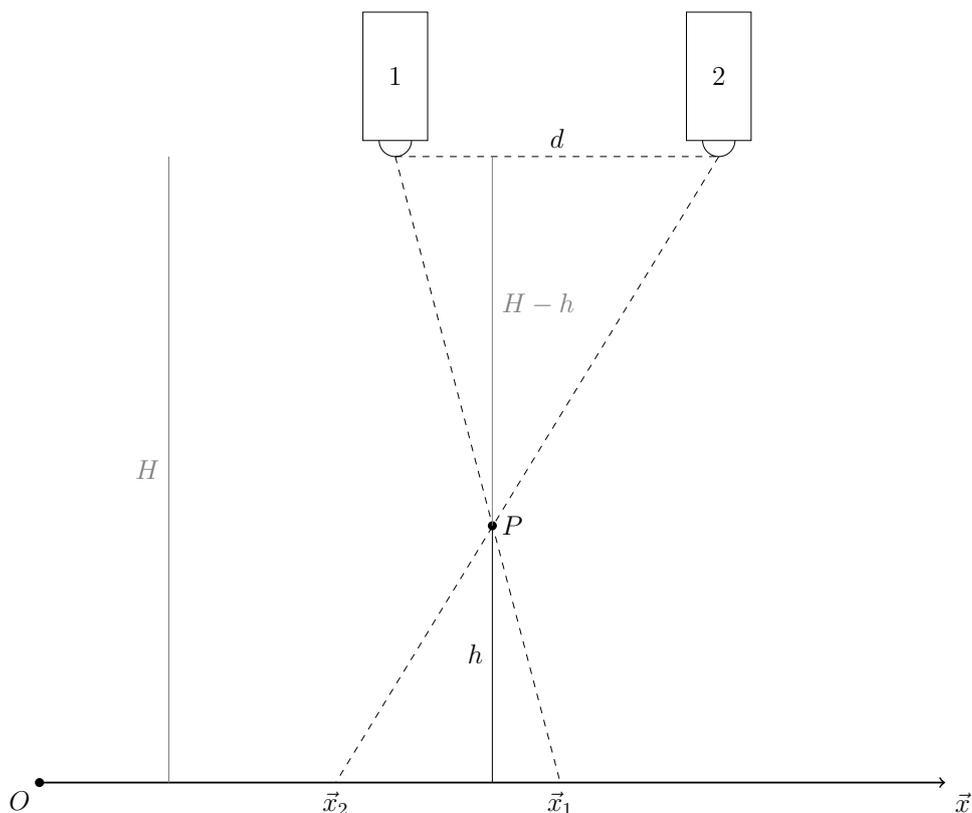


Figura 2.16: Schema per il calcolo dell'altezza h di un punto P . Se è stato fissato un sistema di riferimento comune, conoscendo la distanza tra le telecamere d e la loro altezza da terra H , si può osservare che i triangoli che si vengono a creare sono simili, per cui l'altezza h di un punto può essere ricavata dalla proporzione $d : (H-h) = |\vec{x}_1 - \vec{x}_2| : h$

telecamere ($r = \sqrt{x^2 + y^2}$), l'altezza si può ricavare dalla seguente equazione:

$$h = \frac{|r_1 - r_2| H}{|r_1 - r_2| + d} \quad (2.8)$$

supponendo che le coordinate di tutti i punti abbiano lo stesso segno, ovvero che l'origine del sistema di riferimento sia posta abbastanza vicino ad un angolo.

Il risultato del calcolo dell'altezza su alcuni frame di esempio è visibile in figura 2.17.

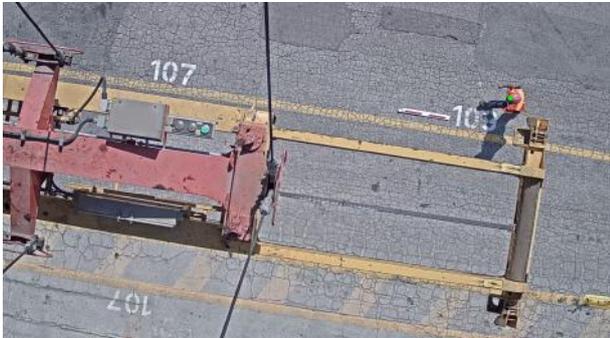
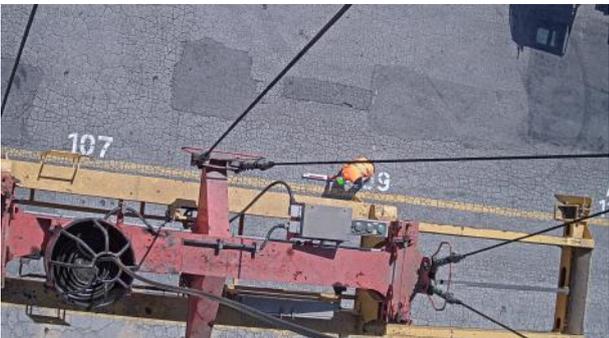
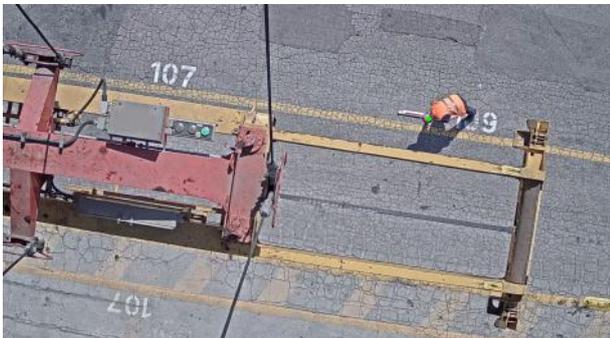
Camera sinistra	Camera destra	Altezza calcolata (cm)
		188.475
		166.842
		154.468
		63.791

Figura 2.17: Calcolo dell'altezza su alcuni frame di esempio. Nell'ultima riga la persona si abbassa a prendere la sbarra e l'altezza scende a 64cm.

2.6 Algoritmo finale di detection

Al termine dell'analisi video, si è a conoscenza della grandezza, altezza e velocità di un eventuale oggetto individuato: se questi valori rientrano in un intervallo prestabilito, l'oggetto viene classificato come persona e una variabile *flag* può ritornare il valore corrispondente. Installando il software nei computer di controllo del porto, sarà possibile collegare a questa variabile flag un qualche tipo di allarme, visivo e/o sonoro, che aiuti l'operatore responsabile nell'individuare possibili pericoli.

La stessa variabile può anche ritornare altri valori che segnalino la presenza di malfunzionamenti sospetti, come ad esempio una ROI che sparisce, che può essere dovuta ad una persona che si è fermata e che quindi il codice assorbe nel modello di sfondo.

Capitolo 3

Risultati

In questo elaborato è stato sviluppato un sistema di riconoscimento di persone semplice, veloce e robusto, che è risultato efficace nella soluzione del problema proposto. Infatti, tramite sottrazione dello sfondo si ottiene una perfetta segmentazione dell'immagine, che unita alla rimozione dell'ombra tramite un metodo cromatico permette di identificare con molta precisione una persona in movimento sotto la gru.

Anche se l'individuazione della testa in un singolo frame non risulta sempre perfetta, l'interpolazione della dinamica fa sì che si abbia assoluta precisione nella posizione della testa della persona individuata.

Il calcolo dell'altezza risulta invece meno preciso, ma comunque abbastanza da poter utilizzare la misura dell'altezza come un ulteriore fattore per classificare come persona un oggetto rilevato. L'imprecisione si può comunque migliorare, qualora venissero installate telecamere identiche, senza rotazione reciproca e magari apposite per un'inquadratura da oltre i 20 metri di distanza.

Il codice risulta inoltre estremamente veloce ed è in grado di analizzare il flusso video da una telecamera in tempo reale. Supponendo un flusso video di 20 fps da ogni telecamera, ciò significa che ogni frame viene analizzato in meno di 0.025 secondi.

Conclusioni

La soluzione implementata nel codice è risultata efficace nella soluzione del problema proposto. Il codice implementato permette infatti di effettuare una perfetta segmentazione di immagini che consente di separare il primo piano dallo sfondo. L'*oggetto* così identificato può anche contenere un'ombra, che viene rimossa con un metodo cromatico. Al classico e ordinario *object detection*, è stata quindi aggiunta una classificazione e rimozione delle ombre. A questa si affianca poi anche il riconoscimento della testa e il fit della sua posizione (con ottimi risultati in termini di accuratezza), includendo quindi nell'analisi un'ipotesi sulla dinamica e mettendo così in relazione frame consecutivi.

Buona, anche se non ottima, è la precisione che si ha nel calcolo delle altezze, che permette di avere un altro criterio per classificare l'oggetto e decidere se sia o no una persona. La precisione può essere tuttavia migliorata, qualora venissero montate telecamere identiche, fisse e senza una rotazione reciproca.

L'algoritmo può e deve essere migliorato nel caso di un'applicazione concreta come sistema di allarme (ad esempio supportare il riconoscimento di più persone simultaneamente, effettuando fit separati per ROI separate) e deve essere testato in situazioni diverse da quella ideale presa in esame, per valutare come cambia sia il riconoscimento di oggetti che la rimozione delle ombre. Ad esempio, è noto come il metodo di rimozione delle ombre basato sulla cromaticità sia veloce e di semplice implementazione, ma meno efficace in situazioni di scarsa luminosità [23].

Tuttavia, con soli pochi parametri ($\alpha, A_{min}, \beta_1, \beta_2, \tau_S, \tau_H$) si è in grado di ricavare con ottima precisione la posizione, l'altezza, la grandezza e la velocità di un oggetto nel video preso in esame. Inoltre il codice è estremamente rapido: è infatti possibile analizzare il flusso di frame in ingresso (da un video registrato o live da una telecamera) in tempo reale.

L'intero codice è disponibile su GitHub[15].

Bibliografia

- [1] Payal Panchal et al. “A Review on Object Detection and Tracking Methods”. In: 2015.
- [2] Mohammed Noman, Vladimir Stankovic e Ayman Tawfik. “Object Detection Techniques: Overview and Performance Comparison”. In: *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. 2019, pp. 1–5. DOI: [10.1109/ISSPIT47144.2019.9001879](https://doi.org/10.1109/ISSPIT47144.2019.9001879).
- [3] Deepjoy Das e Sarat Saharia. “Implementation And Performance Evaluation Of Background Subtraction Algorithms”. In: *International Journal on Computational Science & Applications* 4 (mag. 2014). DOI: [10.5121/ijcsa.2014.4206](https://doi.org/10.5121/ijcsa.2014.4206).
- [4] M. Piccardi. “Background subtraction techniques: a review”. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*. Vol. 4. 2004, 3099–3104 vol.4. DOI: [10.1109/ICSMC.2004.1400815](https://doi.org/10.1109/ICSMC.2004.1400815).
- [5] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: [10.1109/CVPR.2014.81](https://doi.org/10.1109/CVPR.2014.81).
- [6] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: giu. 2016, pp. 779–788. DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [7] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *ECCV (1)*. A cura di Bastian Leibe et al. Vol. 9905. Lecture Notes in Computer Science. Springer, 2016, pp. 21–37. ISBN: 978-3-319-46447-3. URL: <http://arxiv.org/abs/1512.02325>.
- [8] Rafael C. Gonzalez e Richard E. Woods. *Digital Image Processing*. Pearson College Div, 2018. ISBN: 978-0133356724.
- [9] Wikipedia contributors. *RGB color model* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 14-November-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=RGB_color_model&oldid=1121448584.
- [10] Wikipedia contributors. *HSL and HSV* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 14-November-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=1115874918.

- [11] Wikipedia. *Visione artificiale* — *Wikipedia, L'enciclopedia libera*. [Online; in data 11-novembre-2022]. 2021. URL: http://it.wikipedia.org/w/index.php?title=Visione_artificiale&oldid=121125693.
- [12] Dana H. Ballard e Christopher M. Brown. *Computer vision*. Prentice Hall, 1982. ISBN: 0131653164.
- [13] Linda Shapiro e George Stockman. *Computer vision*. Pearson, 2001. ISBN: 978-0130307965.
- [14] Ke Lu et al. “Binocular stereo vision based on OpenCV”. In: *IET International Conference on Smart and Sustainable City (ICSSC 2011)*. 2011, pp. 1–4. DOI: [10.1049/cp.2011.0312](https://doi.org/10.1049/cp.2011.0312).
- [15] Daniele Pucci. *Binocular vision for Ravenna port*. 2022. URL: <https://github.com/puccj/ravenna-port>.
- [16] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb's Journal of Software Tools* (2000).
- [17] Gopal Thapa, Kalpana Sharma e Mrinal Ghose. “Moving Object Detection and Segmentation using Frame Differencing and Summing Technique”. In: *International Journal of Computer Applications* 102 (set. 2014), pp. 20–25. DOI: [10.5120/17828-8647](https://doi.org/10.5120/17828-8647).
- [18] Joshua Migdal e W. Eric L. Grimson. “Background Subtraction Using Markov Thresholds”. In: *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*. Vol. 2. 2005, pp. 58–65. DOI: [10.1109/ACVMOT.2005.33](https://doi.org/10.1109/ACVMOT.2005.33).
- [19] Mohand Allili, Nizar Bouguila e Djemel Ziou. “A Robust Video Foreground Segmentation by Using Generalized Gaussian Mixture Modeling”. In: *giu.* 2007, pp. 503–509. ISBN: 0-7695-2786-8. DOI: [10.1109/CRV.2007.7](https://doi.org/10.1109/CRV.2007.7).
- [20] Jaime Gallego, Montse Pardas e Jose-Luis Landabaso. “Segmentation and tracking of static and moving objects in video surveillance scenarios”. In: *2008 15th IEEE International Conference on Image Processing*. 2008, pp. 2716–2719. DOI: [10.1109/ICIP.2008.4712355](https://doi.org/10.1109/ICIP.2008.4712355).
- [21] R. Girisha e S Murali. “Segmentation of motion objects from surveillance video sequences using partial correlation”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. 2009, pp. 1129–1132. DOI: [10.1109/ICIP.2009.5414526](https://doi.org/10.1109/ICIP.2009.5414526).
- [22] Sovit Ranjan Rath. *Moving Object Detection using Frame Differencing with OpenCV*. URL: <https://debuggercafe.com/moving-object-detection-using-frame-differencing-with-opencv/>. (08.02.2021).

- [23] Andres Sanin, Conrad Sanderson e Brian C. Lovell. “Shadow detection: A survey and comparative evaluation of recent methods”. In: *Pattern Recognition* 45.4 (2012), pp. 1684–1695. ISSN: 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2011.10.001>. URL: <https://www.sciencedirect.com/science/article/pii/S0031320311004043>.
- [24] Alessandro Leone e Cosimo Distanto. “Shadow detection for moving objects based on texture analysis”. In: *Pattern Recognition* 40 (apr. 2007), pp. 1222–1233. DOI: [10.1016/j.patcog.2006.09.017](https://doi.org/10.1016/j.patcog.2006.09.017).
- [25] Yong Shan, Fan Yang e Runsheng Wang. “Color Space Selection for Moving Shadow Elimination”. In: *Fourth International Conference on Image and Graphics (ICIG 2007)*. 2007, pp. 496–501. DOI: [10.1109/ICIG.2007.54](https://doi.org/10.1109/ICIG.2007.54).
- [26] R. Cucchiara et al. “Detecting moving objects, ghosts, and shadows in video streams”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.10 (2003), pp. 1337–1342. DOI: [10.1109/TPAMI.2003.1233909](https://doi.org/10.1109/TPAMI.2003.1233909).
- [27] P. V. C. Hough. “Machine Analysis of Bubble Chamber Pictures”. In: *Conf. Proc. C* 590914 (1959). A cura di L. Kowarski, pp. 554–558.
- [28] Paul VC Hough. *Method and means for recognizing complex patterns*. Dic. 1962. URL: <https://cir.nii.ac.jp/crid/1572543025125903232>.
- [29] Richard O. Duda e Peter E. Hart. “Use of the Hough Transformation to Detect Lines and Curves in Pictures”. In: *Commun. ACM* 15.1 (gen. 1972), pp. 11–15. ISSN: 0001-0782. DOI: [10.1145/361237.361242](https://doi.org/10.1145/361237.361242). URL: <https://doi.org/10.1145/361237.361242>.