



UNIVERSITÀ  
degli STUDI  
di CATANIA

CORSO DI LAUREA MAGISTRALE (LM-18) IN DATA SCIENCE

---

# Identificazione e classificazione dei rifiuti

TESI DI LAUREA MAGISTRALE

**Relatore**

*Chiar.mo prof.* Sebastiano Battiato

**Laureando**

*dott.* Pierpaolo Gumina

**Correlatore**

*prof.* Alessandro Ortis

**Tutor Aziendale**

*ing.* Antonino Casella

---

Aprile 27, 2023

*"Innanzitutto, un grande ringraziamento al mio relatore Prof. Sebastiano Battiato e al mio correlatore Prof. Alessandro Ortis, sempre pronti a guidarmi in ogni fase della realizzazione della tesi. Grazie a voi ho accresciuto le mie conoscenze e le mie competenze".*

*"La mia gratitudine va all'Ing. Casella, mio tutor presso l'azienda TIM, per avermi guidato durante il mio tirocinio formativo e per aver arricchito le mie conoscenze. Porterò sempre con me il bagaglio culturale che mi ha trasmesso".*

*"Un ringraziamento speciale a Marta, che mi ha aiutato a superare i momenti più difficili, che mi ha trasmesso la forza di credere in se stessi".*

*"Infine, un ringraziamento alla mia famiglia, grazie al loro sostegno e al loro incoraggiamento se oggi sono riuscito a raggiungere questo traguardo."*

## Sommario

La gestione dei rifiuti solidi nei centri urbani di grandi dimensioni è diventata un problema di grande complessità a causa della crescente quantità di rifiuti generati ogni giorno da privati e aziende. Le attuali tecniche di Computer Vision e Deep Learning possono aiutare nel rilevamento automatico e nella classificazione dei tipi di rifiuti per le attività di riciclaggio.

Il Machine Learning è un'area con un enorme potenziale in grado di rivoluzionare molte aree sociali e scientifiche, tra cui l'informatica industriale. I metodi studiati includono il modello esistente progettato manualmente e la sua modifica, nonché l'algoritmo di apprendimento automatico. Questo progetto di tesi, svolto in collaborazione con la TIM presso i laboratori di Catania, presenta l'uso di tecniche di Computer Vision legate a tecniche di Machine Learning con lo scopo di risolvere un problema pratico: identificare e classificare i rifiuti da un flusso video.

## Elenco delle figure

1.1	Immagine che mostra le tonnellate di rifiuti prodotte in tutto il mondo dal 2016 ad oggi, con la previsione fino al 2050. . . . .	7
1.2	L'immagine mostra dei cassonetti intelligenti alimentati a energia solare. I sensori incorporati misurano il livello di riempimento del bidone in tempo reale e attivano la compattazione automatica dei rifiuti, aumentando di fatto la capacità del bidone fino a 6-8 volte. Utilizzando moduli 2G/3G, è possibile monitorare da remoto lo stato del bidone. È disponibile in tre diverse dimensioni: 100L, 120L e 240L.	9
1.3	I diversi layer di una CNN. . . . .	10
2.1	Alcune comuni illusioni ottiche e ciò che potrebbero dirci sul sistema visivo: <b>(A)</b> La classica illusione di Muller-Lyer, in cui le lunghezze delle due linee orizzontali appaiono diverse, probabilmente a causa degli effetti prospettici ipotizzati. <b>(B)</b> Il quadrato "bianco" B nell'ombra e il quadrato "nero" A nella luce hanno in realtà lo stesso valore assoluto di intensità. La percezione è dovuta alla costanza di luminosità, il tentativo del sistema visivo di non tener conto dell'illuminazione nell'interpretazione dei colori. . . . .	12
2.2	<b>(A)</b> Gli algoritmi di Structure From Motion possono ricostruire un modello 3D puntiforme di una scena complessa di grandi dimensioni a partire da centinaia di fotografie parzialmente sovrapposte. <b>(B)</b> Gli algoritmi di stereo matching possono costruire un modello 3D dettagliato di un edificio a partire da centinaia di fotografie diversamente esposte prese da Internet. . . . .	13
2.3	Processo di ricostruzione 3D a partire da un'immagine reale. . . . .	17
2.4	Rete neurale Alexnet. . . . .	18
2.5	Microsoft Kinect, un accessorio sviluppato da Microsoft per la console Xbox 360 e non solo, sensibile al movimento del corpo umano . . . . .	19
2.6	Diagramma a blocchi del processo di Background Subtraction. . . . .	22
2.7	Esempio di video con telecamera in movimento che illustra l'ambiguità della definizione di foreground/background. . . . .	22

2.8	Grafico della funzione $z = x \cdot \exp(-x^2 + y^2)$ che illustra la sua pendenza in vari punti della curva. . . . .	30
2.9	Minimi e massimi locali per la funzione $f(x)$ . . . . .	31
2.10	Operazione di convoluzione con l'operatore di Laplace su un'immagine.	33
2.11	In alto. A sinistra: Gli oggetti rappresentati come insiemi grafici. Al centro: Gli oggetti incorporati in uno sfondo per formare un'immagine grafica. A destra: L'oggetto e lo sfondo vengono digitalizzati per formare un'immagine digitale (notare la griglia). Seconda riga: Esempio di elemento strutturante (SE) rappresentato come insieme, immagine grafica e infine come SE digitale. . . . .	34
2.12	(A) Immagine $I$ , composta da un insieme (oggetto) $A$ e dallo sfondo. (B) Quadrato SE, $B$ (il punto è l'origine). (C) Erosione di $A$ da parte di $B$ (mostrato ombreggiato nell'immagine risultante). (D) SE allungato. (E) Erosione di $A$ da parte di $B$ . . . . .	36
2.13	Utilizzo dell'erosione per rimuovere i componenti dell'immagine. (A) Immagine binaria $486 \times 486$ di una maschera wire-bond in cui i pixel in primo piano sono mostrati in bianco. (B)-(D) Immagine erosa utilizzando elementi strutturanti quadrati di dimensioni $11 \times 11$ , $15 \times 15$ e $45 \times 45$ elementi, rispettivamente, tutti con valore 1. . . . .	37
2.14	(A) Immagine $I$ , composta dall'insieme (oggetto) $A$ e dallo sfondo. (B) Quadrato SE (il punto è l'origine). (C) Dilatazione di $A$ mediante $B$ (mostrato in ombra). (D) SE allungato. (E) Dilatazione di $A$ per tale elemento. La linea tratteggiata in (C) ed (E) è il confine di $A$ , indicato come riferimento. . . . .	39
2.15	(A) Immagine $I$ , composta dall'insieme (oggetto) $A$ e dallo sfondo. (B) Elemento strutturante, $B$ . (C) Traslazione di $B$ contenuto in $A$ ( $A$ è mostrato scuro per chiarezza). (D) Apertura di $A$ da parte di $B$ . . . . .	41
2.16	(A) Immagine $I$ , composta dall'insieme (oggetto) $A$ e dallo sfondo. (B) Elemento strutturante $B$ . (C) Traslazione di $B$ tale che $B$ non si sovrapponga a nessuna parte di $A$ . ( $A$ è mostrato scuro per chiarezza). (D) Chiusura di $A$ da parte di $B$ . . . . .	42
2.17	Apertura e chiusura morfologica. (A) Immagine $I$ , composta da un insieme (oggetto) $A$ e da uno sfondo; è mostrato anche un elemento strutturante solido e circolare. (Il punto è l'origine). (B) Elemento strutturante in varie posizioni. (C)-(I) Le operazioni morfologiche utilizzate per ottenere l'apertura e la chiusura. . . . .	44
2.18	[a sinistra] Esempio di apertura. [a destra] Esempio di chiusura. . . . .	45
3.1	Pipeline del progetto . . . . .	48
3.2	Roi . . . . .	50

3.3	[A] Mostra il risultato dell'applicazione della maschera di primo piano. [B] Mostra l'applicazione dell'operatore di apertura. [C] Mostra l'applicazione dell'operatore di chiusura. . . . .	51
3.4	A sinistra l'immagine i cui bordi sono specificati tramite il parametro cv.CHAIN_APPROX_NONE. A destra l'immagine i cui bordi sono specificati tramite il parametro cv.CHAIN_APPROX_SIMPLE. . . . .	53
3.5	<b>[a sinistra]</b> Immagine sfocata con valore di varianza basso, 8.32. <b>[a destra]</b> Immagine non sfocata con valore di varianza alto, 62.16. L'immagine di sinistra verrà scartata dall'algoritmo. . . . .	54
3.6	Immagine che mostra l'architettura della rete ResNet 18. . . . .	56
3.7	Immagini che mostrano un campione rappresentativo della classe di appartenenza. . . . .	58
3.8	<b>SoC:</b> Broadcom BCM2711B0 quad-core A72 (ARMv8-A) 64-bit @ 1.5GHz; <b>GPU:</b> Broadcom VideoCore VI; <b>Networking:</b> 2.4 GHz and 5 GHz 802.11b/g/n/ac wireless LAN; <b>RAM:</b> 2GB LPDDR4 SDRAM; <b>Bluetooth:</b> Bluetooth 5.0, Bluetooth Low Energy (BLE); <b>GPIO:</b> 40-pin GPIO header, populated; <b>Memoria:</b> microSD; <b>Porte:</b> 2 x micro-HDMI 2.0, 3.5mm analogue audio-video jack, 2 x USB 2.0, 2 x USB 3.0, Gigabit Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI); <b>Dimensioni:</b> 88mm x 58mm x 19.5mm, 46g. . . . .	60
3.9	[A] Figura che mostra la scheda di espansione GrovePi+. [B] Grove LCD RGB Backlight . . . . .	61
3.10	[A] Scheda GrovePi+ che evidenzia i 16 connettori disponibili: porte digitali (8), porte analogiche (4) e porte I2C (4). [B] Diagramma che illustra la struttura di un bus I2C. . . . .	62

# Indice

<b>Elenco delle figure</b>	2
<b>1 Introduzione</b>	7
<b>2 Metodi</b>	11
2.1 Computer Vision . . . . .	11
2.1.1 Una breve descrizione storica . . . . .	14
2.2 OpenCV . . . . .	20
2.3 Background Subtractor . . . . .	20
2.3.1 Foreground vs. Background . . . . .	21
2.3.2 Definizione . . . . .	23
2.4 Laplaciano e il suo utilizzo nel Blur Detection . . . . .	29
2.5 Elaborazione morfologica delle immagini . . . . .	33
2.5.1 Erosione . . . . .	34
2.5.2 Dilatazione . . . . .	38
2.5.3 Apertura e Chiusura . . . . .	39
<b>3 Sistema Proposto</b>	47
3.1 Aspetti Software . . . . .	47
3.1.1 Algoritmo di Background Subtractor . . . . .	48
3.1.2 ROI - Region Of Interest . . . . .	49
3.1.3 Opening (apertura) e Closing (chiusura) . . . . .	50
3.1.4 Contorni . . . . .	52
3.1.5 Blur detection . . . . .	54
3.1.6 Metodo di classificazione . . . . .	55
3.2 Aspetti Hardware . . . . .	59
3.2.1 Single Board Computer - Raspberry Pi . . . . .	59
3.2.2 GrovePi+ . . . . .	60
<b>4 Risultati ottenuti</b>	65
<b>5 Conclusioni</b>	67
5.1 Sviluppi futuri . . . . .	67



# Capitolo 1

## Introduzione

L'aumento del tasso di urbanizzazione e dello sviluppo economico in tutto il mondo ha portato a una maggiore produzione di rifiuti. Nell'ultimo secolo la produzione di rifiuti è decuplicata e si prevede che entro il 2025 raddoppierà ancora, Figura 1.1.

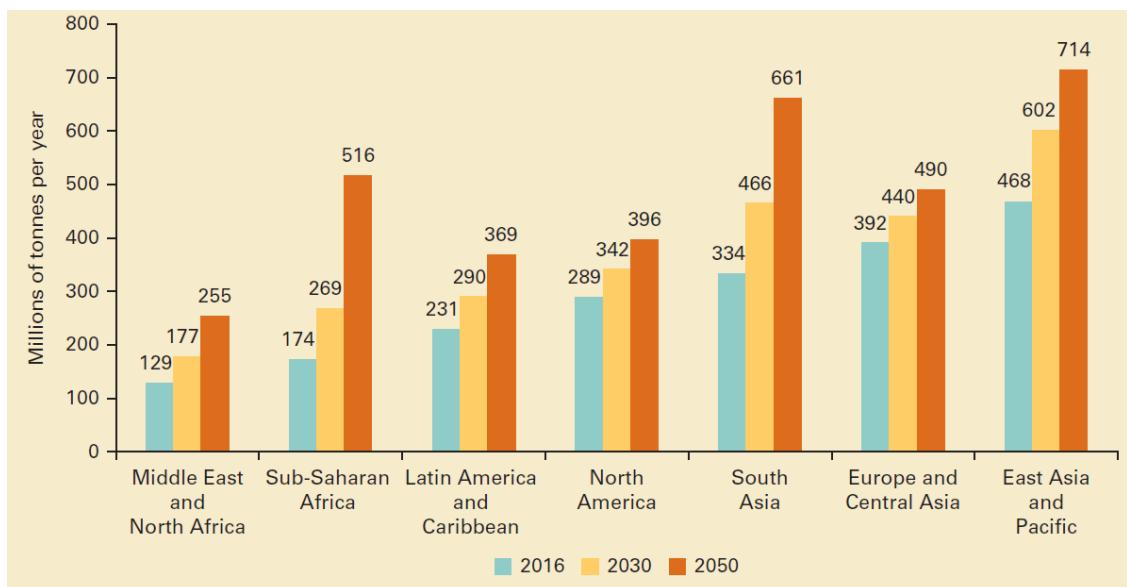


Figura 1.1: Immagine che mostra le tonnellate di rifiuti prodotte in tutto il mondo dal 2016 ad oggi, con la previsione fino al 2050.

Questi rifiuti spesso non vengono riciclati in modo adeguato e vengono solitamente gettati in discarica o inceneriti. Per risolvere il problema dei rifiuti si ricorre agli inceneritori, ma ci sono diverse criticità legate alle emissioni di polveri sottili, che possono comportare una serie di rischi per la salute e contribuire al surriscaldamento globale. Questo può portare all'accumulo di materiali inquinanti che rappresentano una minaccia per l'ambiente nonché per la salute umana e animale. I prodotti di

scarto possono essere classificati in base a vari fattori, come il consumo, la produzione, le proprietà chimiche e fisiche per consentire un efficace riutilizzo e riciclo. La capacità di ridurre e riciclare i rifiuti in modo più efficace non solo riduce l'impatto sull'ambiente, ma è anche più economicamente vantaggioso.

Il compito più semplice e dispendioso in termini di tempo nello smaltimento dei rifiuti è la classificazione dei rifiuti, quali vetro, carta, cartone, plastica e metallo, il che determina la percentuale di riciclaggio dei rifiuti riciclabili e influisce in modo significativo sull'efficienza del successivo trattamento. La precedente classificazione dei rifiuti si basava su una vasta quantità di selezione manuale dei rifiuti che non solo era costosa e inefficiente, ma poteva anche danneggiare la salute degli operatori a causa dei rifiuti tossici. Inoltre, con il rapido aumento della quantità di rifiuti urbani, vengono reclutati operatori di smistamento inesperti per sopperire alla carenza di manodopera. La combinazione di questi fattori rende il riciclaggio dei rifiuti una sfida sempre più impegnativa e prioritaria. Pertanto, un metodo di classificazione dei rifiuti che possa essere applicato automaticamente, rapidamente e facilmente alle stazioni di smistamento dei rifiuti è richiesto con urgenza dai gestori delle città, dai governi e dalle organizzazioni no-profit.

La classificazione dei rifiuti può avvenire in qualsiasi fase del processo di trattamento dei rifiuti, ma è opportuno che avvenga quando i rifiuti vengono inizialmente smaltiti per evitare che i potenziali materiali riciclabili vengano contaminati. I casonetti a energia solare, Figura 1.2, sono diventati sempre più popolari nelle smart city e nei campus, in quanto possono consentire una capacità otto volte superiore a quella di un casonetto tradizionale grazie alla compattazione e possono anche inviare un avviso quando il casonetto è pieno.

Tuttavia, questi bidoni non hanno un meccanismo di classificazione dei rifiuti, il che può portare alla contaminazione del bidone del riciclaggio. I contaminanti più comuni in un ambiente urbano sono i tovaglioli, gli involucri di plastica per alimenti, i sacchetti di plastica, le tazze da caffè, i manicotti da caffè, i guanti di gomma e i rifiuti medici in plastica.

Sono stati sperimentati diversi schemi per informare ed educare meglio il pubblico sulle categorie e sui vantaggi della classificazione dei rifiuti in base alle diverse fasce demografiche. Un meccanismo di classificazione automatizzato risolverebbe questo problema e incoraggerebbe l'impegno degli utenti, che non sarebbero preoccupati di inserire in un contenitore per il riciclaggio un oggetto che potrebbe contaminarlo.

Le reti neurali profonde hanno dimostrato una maggiore accuratezza in una serie di recenti sfide di confronto nell'ambito del machine learning e del pattern recognition. Possiamo inquadrare il problema della classificazione dei rifiuti come un problema di classificazione di immagini, in cui abbiamo una telecamera che scatta un'immagine dei rifiuti e utilizza una rete neurale profonda per identificare la classe di appartenenza dei rifiuti. Per i compiti basati sulle immagini, le reti neurali profonde standard sono difficili da addestrare e incontrano problemi quando



Figura 1.2: L’immagine mostra dei casonetti intelligenti alimentati a energia solare. I sensori incorporati misurano il livello di riempimento del bidone in tempo reale e attivano la compattazione automatica dei rifiuti, aumentando di fatto la capacità del bidone fino a 6-8 volte. Utilizzando moduli 2G/3G, è possibile monitorare da remoto lo stato del bidone. È disponibile in tre diverse dimensioni: 100L, 120L e 240L.

cercano di scalare. Le connessioni dense tradizionali non hanno la proprietà di invarianza alla traslazione, per cui ogni minimo cambiamento nelle dimensioni o nel posizionamento di un’immagine sarebbe difficile da rilevare.

Le reti neurali convoluzionali (CNN), Figura 1.3, risolvono questo problema utilizzando più livelli convoluzionali nascosti che estraggono caratteristiche di alto livello. Gli strati convoluzionali sono costituiti da parametri apprendibili, chiamati filtri che attraversano l’intera immagine, cioè i canali di larghezza, lunghezza e colore dell’immagine e calcola il prodotto interno dell’ingresso e del filtro.

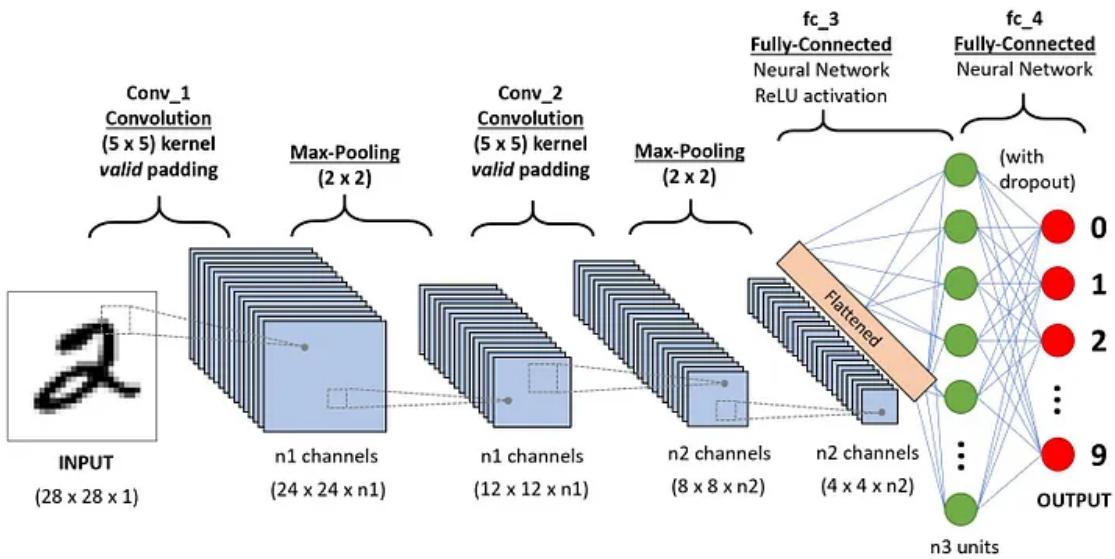


Figura 1.3: I diversi layer di una CNN.

Un altro elemento costitutivo delle reti di convoluzione è il livello di pooling, che opera sulle feature map. Lo strato di pooling riduce la dimensione spaziale della rappresentazione per ridurre la possibilità di overfitting e diminuisce il numero di parametri. Il max pooling è un approccio che divide lo spazio in singole regioni e sceglie il valore massimo per ogni regione. Le CNN utilizzano anche la funzione di attivazione ReLU della forma  $f(x) = \max(0, x)$ , che consente un addestramento più rapido senza compromettere l'accuratezza della rete. La combinazione di questi recenti progressi ha permesso di utilizzare le CNN in una serie di compiti di computer vision, come il rilevamento di volti, il riconoscimento di targhe, la classificazione multi-label, noise recognition e object detection.

# Capitolo 2

## Metodi

La detection di oggetti basati sulla Computer Vision e sul Machine Learning sono due approcci distinti utilizzati per eseguire il task di identificazione di oggetti all'interno di immagini o video. La detection di oggetti basati sulla Computer Vision sfrutta una serie di tecniche di elaborazione delle immagini, quali ad esempio il rilevamento di contorni e il filtraggio dei colori, per rilevare e segmentare gli oggetti dallo sfondo circostante. Quest'approccio risulta efficace nella rilevazione di oggetti di diverse forme e dimensioni, ma può richiedere l'intervento manuale per identificare correttamente gli oggetti in situazioni complesse.

D'altro canto, la detection di oggetti basata sul Machine Learning utilizza algoritmi di apprendimento automatico per identificare gli oggetti all'interno di immagini o video. In tal caso, un dataset di immagini di addestramento contenenti oggetti di interesse viene impiegato per addestrare un algoritmo di machine learning al fine di riconoscere gli oggetti di interesse. Questo algoritmo, una volta addestrato, è in grado di identificare gli oggetti all'interno di nuove immagini sulla base delle conoscenze acquisite durante la fase di addestramento.

### 2.1 Computer Vision

Come esseri umani, percepiamo la struttura tridimensionale del mondo che ci circonda con apparente facilità. Basti pensare a quanto siano vivide le percezioni tridimensionali degli oggetti che ci circondano. È possibile distinguere la forma, la traslucenza, luci, ombre che si manifestano sulle superfici riuscendo a separare correttamente l'oggetto in primo piano dallo sfondo. Gli psicologi percettivi hanno trascorso decenni a cercare di capire come funziona il sistema visivo e, anche se sono in grado di creare illusioni ottiche per svelare alcuni dei suoi meccanismi, vedi la figura 2.1.

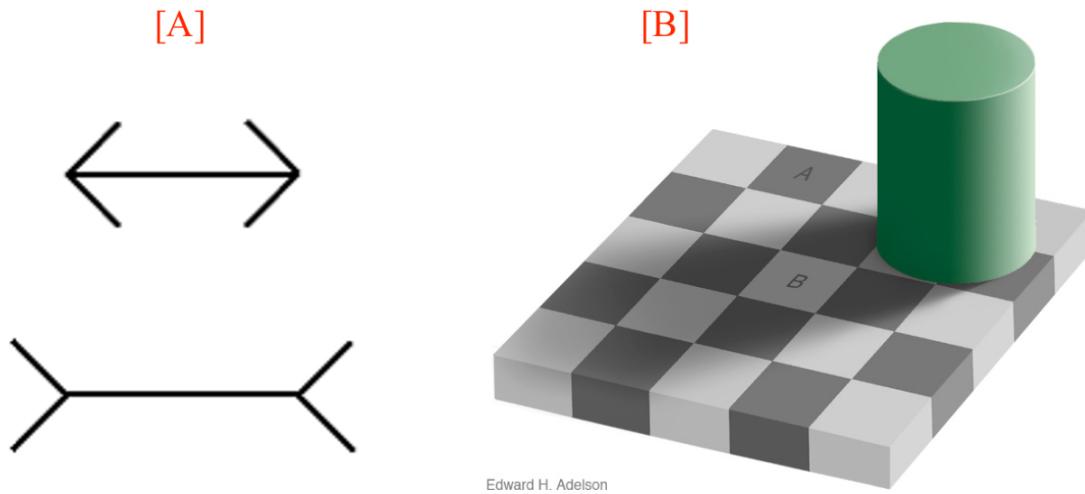


Figura 2.1: Alcune comuni illusioni ottiche e ciò che potrebbero dirci sul sistema visivo: (A) La classica illusione di Muller-Lyer, in cui le lunghezze delle due linee orizzontali appaiono diverse, probabilmente a causa degli effetti prospettici ipotizzati. (B) Il quadrato "bianco" B nell'ombra e il quadrato "nero" A nella luce hanno in realtà lo stesso valore assoluto di intensità. La percezione è dovuta alla costanza di luminosità, il tentativo del sistema visivo di non tener conto dell'illuminazione nell'interpretazione dei colori.

I ricercatori nel campo della computer vision hanno sviluppato, in parallelo, tecniche matematiche per recuperare la forma e l'aspetto tridimensionale degli oggetti nelle immagini. In questo campo, i progressi degli ultimi due decenni sono stati rapidi. Oggi disponiamo di tecniche affidabili per calcolare con precisione un modello 3D di un ambiente a partire da migliaia di fotografie parzialmente sovrapposte (Figura 2.2A). Dato un insieme sufficientemente ampio di viste di un particolare oggetto o di una facciata, possiamo creare modelli 3D densi e accurati di superfici utilizzando la corrispondenza stereo (Figura 2.2B). Possiamo anche, con moderato successo, delineare la maggior parte delle persone e degli oggetti in una fotografia. Tuttavia, nonostante tutti questi progressi, il sogno di avere un computer che spieghi un'immagine con lo stesso livello di dettaglio e causalità di un bambino di due anni rimane inafferrabile.

Perché la visione è così difficile? In parte, perché si tratta di un problema inverso, in cui cerchiamo di recuperare alcune incognite a fronte di informazioni insufficienti per specificare completamente la soluzione. Dobbiamo quindi ricorrere a modelli fisici e probabilistici, o all'apprendimento automatico da grandi insiemi di esempi, per disambiguare tra le potenziali soluzioni. Tuttavia, modellare il mondo visivo in tutta la sua complessità è molto più difficile che, ad esempio, modellare il tratto vocale che produce i suoni parlati.

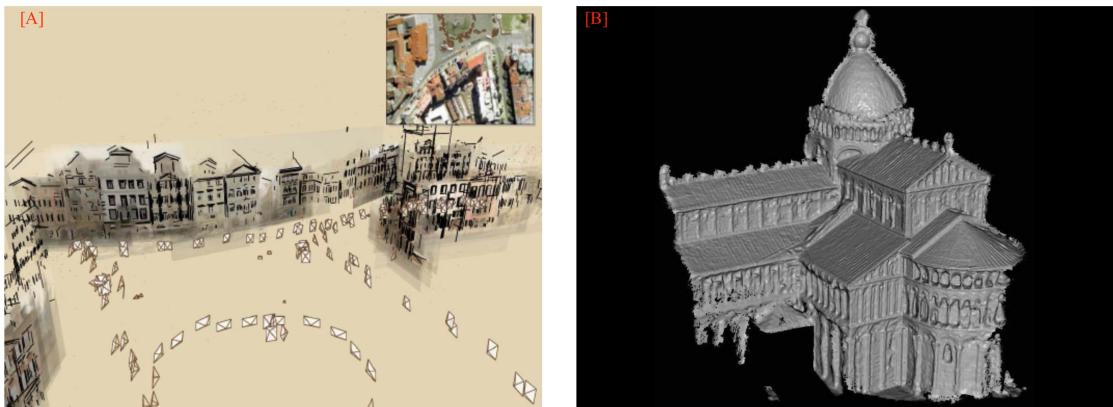


Figura 2.2: (A) Gli algoritmi di Structure From Motion possono ricostruire un modello 3D puntiforme di una scena complessa di grandi dimensioni a partire da centinaia di fotografie parzialmente sovrapposte. (B) Gli algoritmi di stereo matching possono costruire un modello 3D dettagliato di un edificio a partire da centinaia di fotografie diversamente esposte prese da Internet.

I modelli previsionali che utilizziamo nella computer vision sono di solito sviluppati in fisica (radiometria, ottica e progettazione di sensori) e nella computer grafica. Entrambi questi campi modellano il modo in cui gli oggetti si muovono e si animano, il modo in cui la luce si riflette sulle loro superfici, viene dispersa dall'atmosfera, rifratta attraverso le lenti delle telecamere (o gli occhi umani) e infine proiettata su un piano d'immagine piatto (o curvo). Sebbene la grafica computerizzata non sia ancora perfetta, in molti ambiti, come il rendering di una scena fissa composta da oggetti di uso quotidiano o l'animazione di creature estinte come i dinosauri, l'illusione della realtà è essenzialmente presente. Nella computer vision cerchiamo di fare l'inverso, cioè di descrivere il mondo che vediamo in una o più immagini e di ricostruirne le proprietà, come la forma, l'illuminazione e la distribuzione dei colori. È sorprendente che gli esseri umani e gli animali riescano a farlo con tanta facilità, mentre gli algoritmi di visione computerizzata sono così inclini all'errore. Le persone che non hanno lavorato nel campo spesso sottovalutano la difficoltà del problema. Questa errata percezione che la visione debba essere facile risale agli albori dell'intelligenza artificiale, quando inizialmente si riteneva che le parti cognitive (dimostrazione logica e pianificazione) dell'intelligenza fossero intrinsecamente più difficili delle componenti percettive.

La buona notizia è che oggi la computer vision viene utilizzata in un'ampia varietà di applicazioni del mondo reale, tra cui:

- **Riconoscimento ottico dei caratteri (OCR):** lettura dei codici postali scritti a mano sulle lettere e riconoscimento automatico delle targhe (ANPR);

- **Ispezione meccanica:** ispezione rapida delle parti per la garanzia della qualità, utilizzando la visione stereoscopica con illuminazione specifica per misurare le tolleranze sulle ali degli aerei o sulle parti della carrozzeria o per cercare i difetti nelle fusioni di acciaio utilizzando la visione a raggi X;
- **Retail:** riconoscimento degli oggetti per casse automatiche e negozi completamente automatizzati;
- **Logistica di magazzino:** consegna autonoma dei pacchi e "unità" di trasporto dei pallet e prelievo di pezzi da parte di manipolatori robotici;
- **Imaging medico:** registrazione di immagini pre-operatorie e intra-operatorie o studi a lungo termine sulla morfologia cerebrale delle persone che invecchiano;
- **Veicoli a guida autonoma:** in grado di guidare da punto a punto tra le città.
- **Costruzione di modelli 3D (fotogrammetria):** costruzione interamente automatizzata di modelli 3D da fotografie aeree e da droni;
- **Match move:** unione di immagini generate al computer (CGI) con filmati live action tracciando i punti caratteristici del video di partenza per stimare il movimento della telecamera 3D e la forma dell'ambiente. Queste tecniche sono ampiamente utilizzate a Hollywood, inoltre, richiedono l'uso di un matting preciso per inserire nuovi elementi tra quelli in primo piano e quelli sullo sfondo.
- **Motion capture (mocap):** utilizzo di marcatori retro-riflettenti rilevati da più telecamere o altre tecniche basate sulla visione per catturare soggetti per l'animazione al computer;
- **Sorveglianza:** monitoraggio di intrusi, analisi del traffico autostradale e monitoraggio delle piscine per le vittime di annegamento;
- **Riconoscimento delle impronte digitali e biometria:** per l'autenticazione automatica degli accessi e per applicazioni forensi.

### **2.1.1 Una breve descrizione storica**

In questa sezione si fornisce una breve sintesi dei principali sviluppi della computer vision negli ultimi cinquant'anni.

## 1970

Quando la computer vision fece la sua comparsa, all'inizio degli anni Settanta, era vista come la componente della percezione visiva di un ambizioso programma volto a imitare l'intelligenza umana e a dotare i robot di un comportamento intelligente. All'epoca, alcuni dei primi pionieri dell'intelligenza artificiale e della robotica ritenevano che la soluzione del problema degli "input visivi" sarebbe stata un passo facile verso la soluzione di problemi più difficili, come il ragionamento e la pianificazione di livello superiore. Ciò che distingueva la computer vision dal campo già esistente dell'elaborazione digitale delle immagini era il desiderio di recuperare la struttura tridimensionale del mondo dalle immagini e di utilizzarla come trampolino di lancio verso la comprensione completa della scena. I primi tentativi di comprensione della scena prevedevano l'estrazione dei bordi e la successiva deduzione della struttura 3D di un oggetto o di un "mondo a blocchi" dalla struttura topologica delle linee 2D (Roberts 1965). All'epoca furono sviluppati diversi algoritmi di etichettatura delle linee e del rilevamento dei bordi. Un approccio qualitativo per comprendere le variazioni di intensità e ombreggiatura e spiegarle con gli effetti dei fenomeni di formazione dell'immagine, come l'orientamento della superficie e le ombre, è stato promosso da Barrow e Tenenbaum (1981) nel loro articolo sulle immagini intrinseche. In quel periodo furono sviluppati anche approcci più quantitativi alla computer vision, tra cui il primo di molti algoritmi di corrispondenza stereo basati su caratteristiche.

## 1980

Negli anni '80, l'attenzione si è concentrata su tecniche matematiche più sofisticate per l'analisi quantitativa di immagini e scene. Le immagini piramidali<sup>1</sup> hanno iniziato a essere ampiamente utilizzate per eseguire compiti come la fusione di immagini e la ricerca di corrispondenza da grande a piccola.

## 1990

Mentre molti degli argomenti precedentemente menzionati hanno continuato a essere esplorati, alcuni di essi sono diventati significativamente più attivi. Un'ondata di attività sull'uso degli invarianti proiettivi per il riconoscimento si è evoluta

---

<sup>1</sup>Normalmente si lavora con immagini con risoluzione predefinita, ma spesso è necessario modificare la risoluzione (abbassarla) o ridimensionare l'immagine originale: in questo caso le piramidi di immagini sono utili. La funzione `pyrUp()` aumenta la dimensione al doppio della dimensione originale e la funzione `pyrDown()` la riduce alla metà. Se manteniamo l'immagine originale come immagine di base e continuiamo ad applicare la funzione `pyrDown` su di essa e manteniamo le immagini in una pila verticale, l'aspetto sarà quello di una piramide. Lo stesso vale per il ridimensionamento dell'immagine originale mediante la funzione `pyrUp`.

in uno sforzo congiunto per risolvere il problema del structure from motion<sup>2</sup>. Anche gli algoritmi di tracciamento sono migliorati molto, compreso il tracciamento dei contorni utilizzando contorni attivi. La segmentazione delle immagini, un argomento attivo fin dagli albori della computer vision, è stato anche un argomento di ricerca attivo, che ha prodotto tecniche basate sulla minima energia e sulla minima lunghezza di descrizione, tagli normalizzati e mean shift.

Le tecniche di apprendimento statistico hanno iniziato a fare la loro comparsa, dapprima nell'applicazione dell'analisi degli autogeni delle componenti principali per il riconoscimento dei volti e dei sistemi dinamici lineari per il tracciamento delle curve. Forse lo sviluppo più significativo della computer vision in questo decennio è stata la maggiore interazione con la computer grafica, soprattutto nell'area interdisciplinare della modellazione e del rendering basati sulle immagini. L'idea di manipolare direttamente le immagini del mondo reale per creare nuove animazioni è emersa per la prima volta con le tecniche di morphing delle immagini ed è stata successivamente applicata all'interpolazione delle viste, allo stitching di immagini panoramiche e al rendering. Contemporaneamente, sono state introdotte anche tecniche di modellazione basate sulle immagini per la creazione automatica di modelli 3D realistici a partire da collezioni di immagini, vedi Figura 2.3.

---

<sup>2</sup>Structure from motion o SfM (in italiano: Struttura dal movimento) è una tecnica di range imaging della computer vision e della percezione visiva, con cui il processo di stima di strutture tridimensionali da sequenze di immagini bidimensionali che può essere accoppiato con segnali di movimento locali. Nella visione biologica la SfM fa riferimento al fenomeno per cui gli esseri umani (e altre creature viventi) posso ricostruire strutture 3D da un campo in movimento in proiezione 2D (retinale) di un oggetto o una scena in movimento.

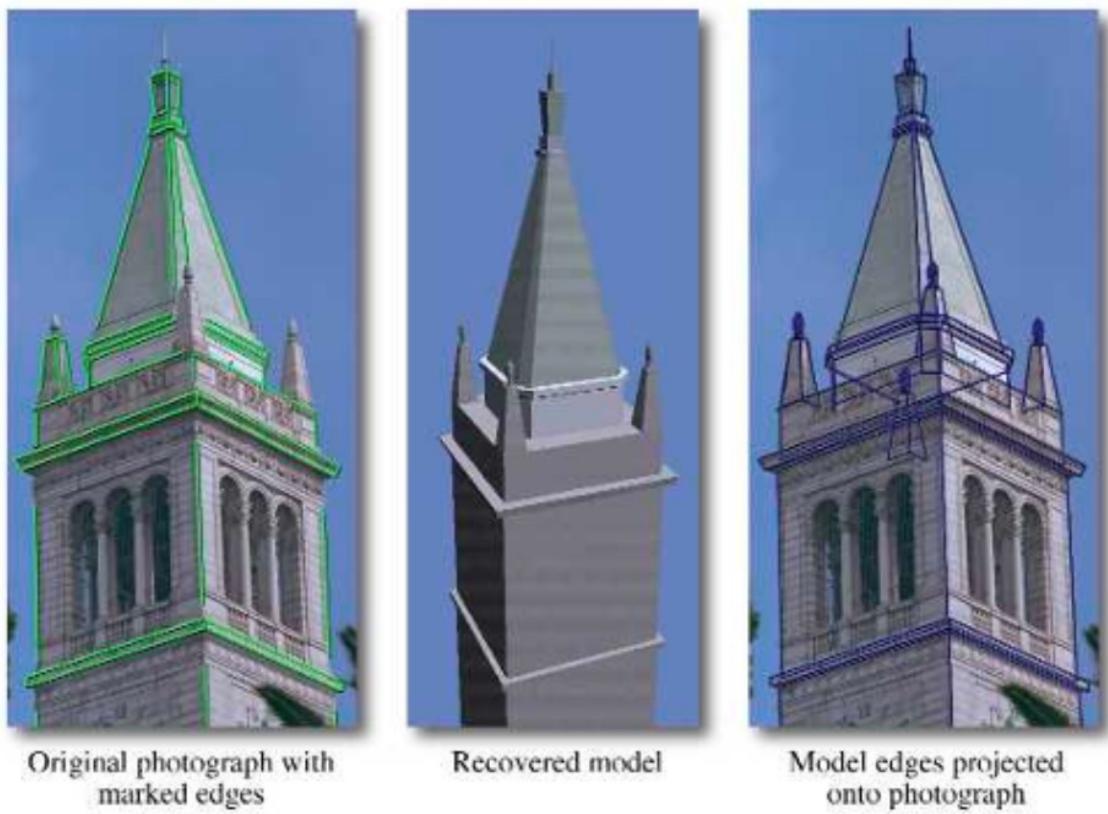


Figura 2.3: Processo di ricostruzione 3D a partire da un’immagine reale.

## 2000

Questo decennio ha continuato ad approfondire l’interazione tra i campi della visione e della grafica, ma soprattutto ha accolto gli approcci basati sui dati e sull’apprendimento come componenti fondamentali della visione. La tendenza più rilevante di questo decennio, è sul riconoscimento visivo. È stata l’applicazione di sofisticate tecniche di apprendimento automatico ai problemi di computer vision. Questa tendenza ha coinciso con la maggiore disponibilità di immense quantità di dati parzialmente etichettati su Internet, nonché con un significativo aumento della potenza di calcolo, che rende più fattibile l’apprendimento di categorie di oggetti senza l’uso di un’attenta supervisione umana.

## 2010

La tendenza a utilizzare grandi insiemi di dati etichettati (e anche auto-supervisionati) per sviluppare algoritmi di apprendimento automatico è diventata un’onda anomala che ha completamente rivoluzionato lo sviluppo di algoritmi di riconoscimento delle immagini e di altre applicazioni, come il denoising e il flusso ottico, che in

precedenza utilizzavano tecniche bayesiane e di ottimizzazione globale. Questa tendenza è stata favorita dallo sviluppo di dataset annotati di alta qualità su larga scala, come ImageNet, Microsoft COCO (Common Objects in Context) e LVIS. Questi set di dati hanno fornito non solo metriche affidabili per seguire i progressi degli algoritmi di riconoscimento e segmentazione semantica, ma soprattutto dati etichettati sufficienti per sviluppare soluzioni complete basate sull'apprendimento automatico.

Un'altra tendenza importante è stata l'enorme aumento della potenza di calcolo disponibile grazie allo sviluppo di algoritmi general purpose (data-parallel) su unità di elaborazione grafica (GPGPU). La rivoluzionaria rete neurale profonda SuperVision ("AlexNet") (Figura 2.4), che è stata la prima rete neurale a vincere la sfida annuale ImageNet per il riconoscimento di immagini su larga scala, si è basata sull'addestramento tramite GPU, oltre che su una serie di progressi tecnici, per ottenere prestazioni straordinarie. Dopo la pubblicazione di questo articolo, i progressi nell'uso delle architetture convoluzionali profonde hanno subito una forte accelerazione, al punto che oggi sono l'unica architettura considerata per i compiti di riconoscimento e segmentazione semantica.

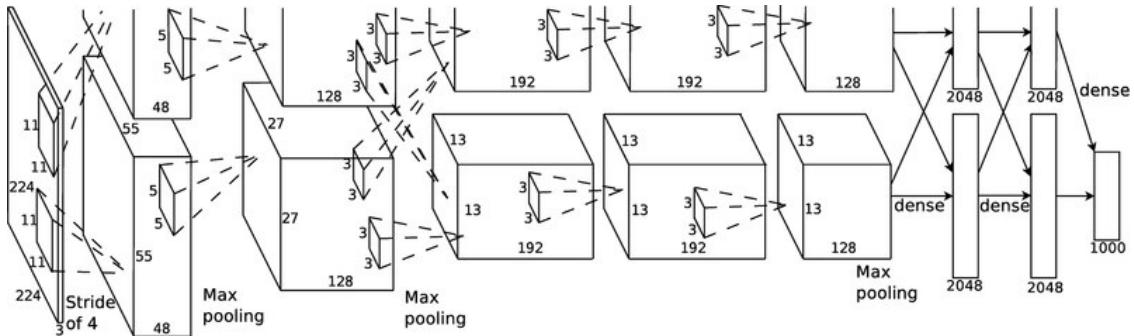


Figura 2.4: Rete neurale Alexnet.

Anche i sensori e l'hardware specializzati per le attività di computer vision hanno continuato a progredire. La telecamera di profondità Microsoft Kinect (Figura 2.5), rilasciata nel 2010, è diventata rapidamente un componente essenziale di molti sistemi di modellazione 3D e di tracciamento delle persone. Nel corso del decennio, i sistemi di modellazione e tracciamento della forma del corpo in 3D hanno continuato a evolversi, al punto che oggi è possibile dedurre il modello 3D di una persona con gesti ed espressioni da una singola immagine.



Figura 2.5: Microsoft Kinect, un accessorio sviluppato da Microsoft per la console Xbox 360 e non solo, sensibile al movimento del corpo umano

Infine, le applicazioni mobile di realtà aumentata che eseguono in tempo reale la stima della posa e l'aumento dell'ambiente utilizzando combinazioni di tracciamento delle caratteristiche e misurazioni inerziali sono comuni e sono attualmente in fase di estensione per includere effetti di occlusione di profondità accurati al pixel.

In sintesi, nell'ultimo decennio si sono registrati incredibili progressi nelle prestazioni e nell'affidabilità degli algoritmi di visione artificiale, in parte dovuti al passaggio all'apprendimento automatico e all'addestramento su serie molto ampie di dati reali. Ha visto anche l'applicazione degli algoritmi di visione in una miriade di scenari commerciali e di consumo, nonché le nuove sfide generate dal loro uso diffuso.

## 2.2 OpenCV

OpenCV (Open Source Computer Vision) è una libreria open source di computer vision e machine learning sviluppata principalmente in C++ ma che supporta anche Python e Java. È stata creata per fornire una piattaforma comune per gli sviluppatori di visione artificiale e viene utilizzata in una vasta gamma di applicazioni, tra cui riconoscimento di oggetti, rilevamento facciale, tracciamento dei movimenti, analisi video e molto altro.

OpenCV offre un'ampia gamma di funzionalità, tra cui il rilevamento di contorni, il rilevamento di linee e cerchi, il riconoscimento di forme, la correzione dell'illuminazione e la segmentazione delle immagini. Inoltre, la libreria supporta anche l'apprendimento automatico, che consente di creare modelli di intelligenza artificiale per eseguire compiti complessi come la classificazione di immagini e la rilevazione di oggetti.

OpenCV viene utilizzato in molte applicazioni in cui la visione artificiale è un elemento chiave, tra cui la sicurezza, la sorveglianza, la produzione, la medicina, la robotica, l'automazione industriale, l'agricoltura, l'analisi delle immagini satellitari e molto altro ancora. Ad esempio, in campo medico, OpenCV viene utilizzato per l'analisi delle immagini radiografiche e per il rilevamento di tumori. Nell'industria, viene utilizzato per il controllo qualità dei prodotti e per l'ispezione dei circuiti stampati. Nella robotica, viene utilizzato per la navigazione e il riconoscimento di oggetti.

Inoltre, OpenCV viene utilizzato anche in applicazioni per il riconoscimento facciale, come le applicazioni di sorveglianza. In queste applicazioni, il software può rilevare volti umani all'interno di un'immagine e identificare le caratteristiche facciali come gli occhi, il naso e la bocca. Questo può essere utilizzato per rilevare i movimenti delle persone all'interno di un'area monitorata o per identificare persone specifiche.

In sintesi, OpenCV è una libreria di visione artificiale potente ed estremamente utile, che può essere utilizzata in molte applicazioni diverse. Grazie alla sua flessibilità, OpenCV può essere utilizzato con una vasta gamma di linguaggi di programmazione, il che lo rende una scelta popolare per gli sviluppatori che lavorano in diversi ambienti.

## 2.3 Background Subtractor

Il costante aumento del volume di dati video acquisiti ogni giorno offre l'opportunità e la necessità di varie soluzioni applicative reali che richiedono il riconoscimento

di alto livello di oggetti, scene, attività ed eventi contenuti nei dati video. Esempi di queste applicazioni sono la videosorveglianza automatizzata, l’archiviazione e il recupero di video, la guida autonoma, l’analisi sportiva, la "motion capture" senza marcatori, ecc. Il rilevamento e la segmentazione di oggetti di interesse, come persone e veicoli, sono le prime fasi di queste applicazioni di rilevamento automatico di eventi visivi. È sempre auspicabile ottenere un’accuratezza molto elevata nel rilevamento dei target con un tasso di falsi positivi il più basso possibile.

Possiamo caratterizzare i domini dei video attraverso due dimensioni ortogonali: il movimento della telecamera (video catturati da una telecamera statica o da una piattaforma mobile) e il requisito di elaborazione per il processo decisionale (video offline vs. video in streaming online). Negli ultimi tre decenni, sono stati sviluppati diversi algoritmi per affrontare il rilevamento e la segmentazione degli oggetti di interesse, sfruttando le caratteristiche di ciascuna di queste categorie di video per ottenere soluzioni specifiche ed efficienti.

Nelle applicazioni di videosorveglianza, le telecamere fisse o le telecamere pan-tilt-zoom (PTZ) vengono utilizzate per monitorare le attività in luoghi esterni o interni. In alcune applicazioni di sorveglianza è necessaria l’elaborazione online in tempo reale dei video, mentre in altre applicazioni i video vengono compressi e registrati e l’elaborazione viene eseguita offline. Poiché le telecamere sono ferme, il rilevamento di oggetti in movimento può essere ottenuto confrontando ogni nuovo fotogramma con una rappresentazione dello sfondo della scena. Questo processo è chiamato "background subtraction" (sottrazione dello sfondo) e la rappresentazione della scena è chiamata "background model" (modello dello sfondo). In questo caso si assume che la scena sia statica o pressoché statica. La Figura 2.6 illustra un diagramma a blocchi di un tipico processo di background subtraction.

Indipendentemente dal fatto che l’elaborazione sia richiesta online o offline per questo tipo di video, le tecniche tradizionali di background subtraction forniscono soluzioni efficaci per il rilevamento e la segmentazione dei target in questi scenari.

### 2.3.1 Foreground vs. Background

Capire cosa costituisca al foreground (primo piano) e cosa al background (sfondo) della scena è essenziale per poter definire cosa sia il "background subtraction". Nel caso di una telecamera fissa, la definizione di foreground e background è chiara: un oggetto in primo piano è tutto ciò che non assomiglia allo sfondo fisso, in termini di aspetto o di movimento locale. Questa definizione presuppone che si conosca a priori, o si possa stimare online, quale sia lo sfondo della scena. Pertanto, è sufficiente modellare la distribuzione statistica delle caratteristiche dello sfondo della scena, e l’individuazione del primo piano è principalmente un processo di individuazione dei valori anomali di tale distribuzione. Una definizione alternativa, nel

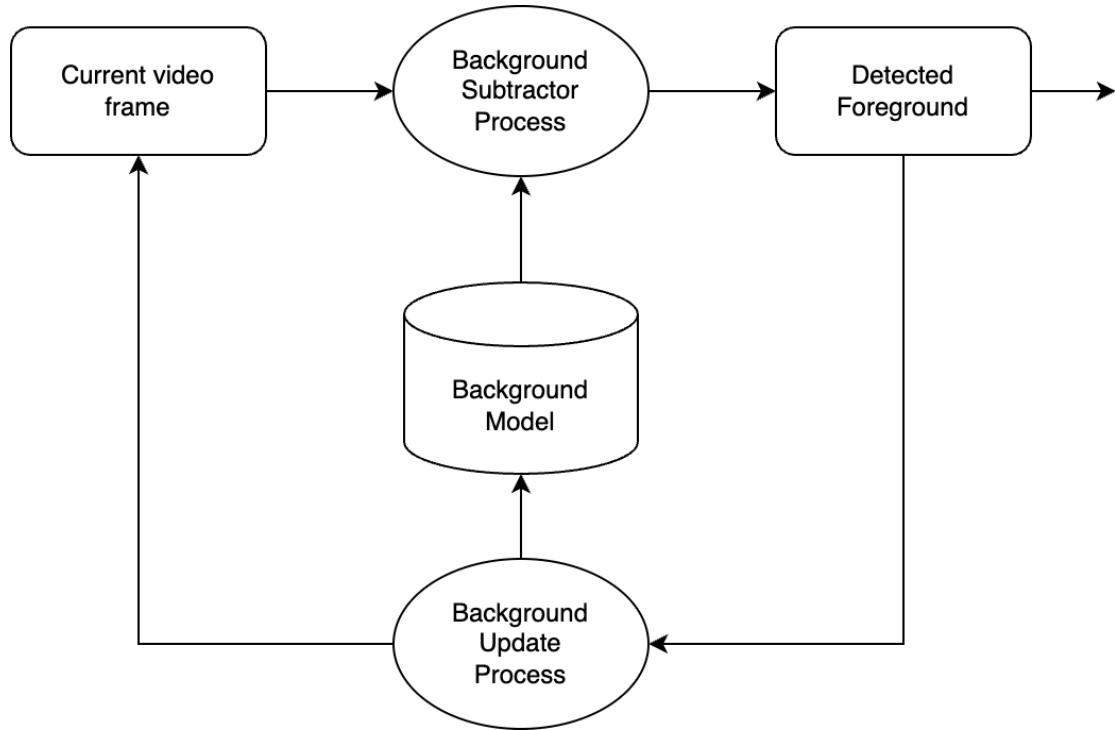


Figura 2.6: Diagramma a blocchi del processo di Background Subtraction.

caso di una telecamera e di una scena stazionarie, è: qualsiasi oggetto "significativamente" in movimento è in primo piano, mentre qualsiasi oggetto stazionario fa parte dello sfondo della scena. Ovviamenete, le due definizioni di cui sopra non sono equivalenti.



Figura 2.7: Esempio di video con telecamera in movimento che illustra l'ambiguità della definizione di foreground/background.

Nel caso di una telecamera che si muove liberamente, la definizione di primo piano e sfondo è ambigua. Ad esempio, consideriamo i fotogrammi della Figura 2.7, tratti da un video catturato da una telecamera in movimento. Che cosa è il primo piano e che cosa è lo sfondo in questo caso? L'albero dovrebbe essere considerato il primo piano? Se abbiamo un modo per recuperare le informazioni sulla profondità e se siamo interessati agli oggetti più vicini alla telecamera, allora potremmo considerare l'albero come primo piano. Tuttavia, anche affidarsi alla profondità è ambiguo.

Qual è la soglia di profondità che possiamo utilizzare per decidere il primo piano? Che dire, ad esempio, della pianta del terreno, dove la profondità cambia in modo uniforme? Anche se consideriamo il movimento piuttosto che la profondità, la risposta non è chiara. Considerando il movimento apparente dell'immagine (movimento 2D), ogni elemento nella scena si muove. Considerando il movimento fisico (recupero del movimento 3D), l'albero, il terreno e l'edificio costituiscono un'unica entità rigida che ha lo stesso movimento.

### 2.3.2 Definizione

Una telecamera statica che osserva una scena è un caso comune di sistema di sorveglianza. L'individuazione di oggetti intrusi è un passo essenziale nell'analisi della scena. Un'ipotesi solitamente applicabile è che le immagini della scena senza gli oggetti intrusi presentino un comportamento regolare che può essere ben descritto da un modello statistico. Se si dispone di un modello statistico della scena, è possibile rilevare un oggetto intruso individuando le parti dell'immagine che non si adattano al modello. Questo processo, come detto in precedenza, è noto come “background subtraction” (sottrazione dello sfondo).

Nel caso della comune sottrazione dello sfondo a livello di pixel, il modello della scena ha una funzione di densità di probabilità per ogni pixel distinto. Un pixel di una nuova immagine è considerato un pixel di sfondo se il suo nuovo valore è ben descritto dalla sua funzione di densità. Per una scena statica, il modello più semplice potrebbe essere un'immagine della scena senza gli oggetti intrusi. Il passo successivo consisterebbe, ad esempio, nello stimare valori appropriati per le varianze dei livelli di intensità dei pixel dall'immagine, in quanto le varianze possono variare da pixel a pixel. Tuttavia, i valori dei pixel hanno spesso distribuzioni complesse e sono necessari modelli più elaborati. Nella sezione successiva, verrà esposto il problema della sottrazione dello sfondo basata sui pixel.

#### Background subtraction basato sui pixel

Il valore di un pixel al tempo  $t$  in RGB è indicato con  $\vec{x}^{(t)}$ . Si possono utilizzare anche altri spazi di colore o caratteristiche locali. Ad esempio, in (Mittal e Paragios, 2004 [1]) sono stati utilizzati colori normalizzati e stime del flusso ottico. La sottrazione dello sfondo basata sui pixel comporta la decisione se il pixel appartiene allo sfondo (BG) o a un oggetto in primo piano (FG). È più probabile che il pixel appartenga allo sfondo se

$$\frac{p(BG|\vec{x}^{(t)})}{p(FG|\vec{x}^{(t)})} = \frac{p(\vec{x}^{(t)}|BG)p(BG)}{p(\vec{x}^{(t)}|FG)p(FG)} \quad (2.1)$$

è maggiore di 1 e viceversa. I risultati della sottrazione dello sfondo vengono solitamente propagati ad alcuni moduli di livello superiore, ad esempio gli oggetti

rilevati vengono spesso tracciati. Durante il tracciamento di un oggetto si possono ottenere alcune conoscenze sull'aspetto dell'oggetto tracciato e queste conoscenze possono essere utilizzate per migliorare la sottrazione dello sfondo. Nel caso generale, non sappiamo nulla degli oggetti in primo piano che possono essere visti, né quando e quanto spesso saranno presenti. Pertanto, si assume una distribuzione uniforme per l'aspetto degli oggetti in primo piano  $p(\vec{x}^{(t)}|FG)$ . La decisione sull'appartenenza di un pixel allo sfondo viene presa se:

$$p(\vec{x}^{(t)}|BG) > c_{thr} (= p(\vec{x}^{(t)}|FG)p(FG)/p(BG)) \quad (2.2)$$

dove  $c_{thr}$  è un valore di soglia, mentre  $p(\vec{x}^{(t)}|BG)$  è il modello di sfondo. Il modello di sfondo è stimato da un set di allenamento  $\chi$ . Il modello stimato è indicato con  $\hat{p}(\vec{x}|\chi, BG)$  e dipende dal set di allenamento, come indicato esplicitamente. In pratica, l'illuminazione della scena può cambiare gradualmente (giorno o condizioni atmosferiche in una scena esterna) o improvvisamente (spegnimento o accensione della luce in una scena interna). Un nuovo oggetto può essere introdotto nella scena o un oggetto presente può essere rimosso da essa. Per adattarsi a questi cambiamenti, possiamo aggiornare il set di addestramento aggiungendo nuovi campioni e scartando quelli vecchi. Assumiamo che i campioni siano indipendenti e il problema principale è come stimare in modo efficiente la funzione di densità online. In letteratura esistono modelli che considerano l'aspetto temporale di una sequenza di immagini e quindi la decisione dipende anche dai valori dei pixel precedenti della sequenza. Ad esempio, in (Toyama et al., 1999 [2]; Monnet et al., 2003 [3]) la distribuzione dei valori dei pixel nel tempo è modellata come un processo autoregressivo. In (Stenger et al., 2001[4]; Kato et al., 2002 [5]) vengono utilizzati modelli di Markov nascosti. Tuttavia, questi metodi sono generalmente molto più lenti e l'adattamento ai cambiamenti della scena è difficile.

Gli approcci pixel-wise presuppongono che i pixel adiacenti non siano correlati. Il Markov random field può essere utilizzato per modellare la correlazione tra i valori dei pixel adiacenti (Kato et al., 2002 [5]), ma porta ad algoritmi lenti e complessi. Un filtraggio aggiuntivo delle immagini segmentate spesso migliora i risultati, poiché impone una certa correlazione (Elgammal et al., 2000 [6]; Cemgil et al., 2005 [7]). Un altro argomento correlato è il rilevamento delle ombre. L'oggetto intruso può proiettare ombre sullo sfondo. Di solito, siamo interessati solo all'oggetto e i pixel corrispondenti all'ombra devono essere rilevati (Prati et al., 2003 [8]).

## Gaussian mixture model

Per adattarsi ai possibili cambiamenti, il training set deve essere aggiornato. Scegliamo un periodo di adattamento ragionevole  $T$ . Al tempo  $t$  abbiamo  $\chi_T = \{x^{(t)}, \dots, x^{(t-T)}\}$ . Per ogni nuovo campione aggiorniamo l'insieme di dati di addestramento  $\chi_T$  e stimiamo nuovamente la densità. Questi campioni potrebbero

contenere valori che appartengono agli oggetti in primo piano. Pertanto, dovremmo indicare la densità stimata come  $\hat{p}(\vec{x}^{(t)}|\chi_T, BG + FG)$ . Utilizziamo una GMM con  $M$  componenti:

$$\hat{p}(\vec{x}^{(t)}|\chi_T, BG + FG) = \sum_{m=1}^M \hat{\pi}_m N(\vec{x}; \hat{\mu}_m, \hat{\sigma}_m^2 I) \quad (2.3)$$

dove  $\hat{\mu}_1, \dots, \hat{\mu}_M$  sono le stime delle medie e  $\hat{\sigma}_1^2, \dots, \hat{\sigma}_M^2$  sono le stime delle varianze che descrivono le componenti gaussiane. Per motivi computazionali, le matrici di covarianza sono mantenute isotrope. La matrice identità  $I$  ha dimensioni appropriate. I coefficienti di mixing stimati, indicati con  $\hat{\pi}_m$ , sono non negativi e sommano a uno.

**Aggiornamento delle equazioni** Dato un nuovo campione di dati  $\vec{x}^{(t)}$  al tempo  $t$ , le equazioni di aggiornamento ricorsivo sono (Titterington, 1984 [9]):

$$\hat{\pi}_m \leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m), \quad (2.4)$$

$$\hat{\mu}_m \leftarrow \hat{\mu}_m + o_m^{(t)}(\alpha/\hat{\pi}_m)\vec{\delta}_m, \quad (2.5)$$

$$\hat{\sigma}_m^2 \leftarrow \hat{\sigma}_m^2 + o_m^{(t)}(\alpha/\hat{\pi}_m)(\vec{\delta}_m^T \vec{\delta}_m - \hat{\sigma}_m^2), \quad (2.6)$$

dove  $\vec{\delta}_m = \vec{x}^{(t)} - \hat{\mu}_m$ .

Invece dell'intervallo di tempo  $T$  menzionato in precedenza, qui la costante  $\alpha$  definisce un inviluppo a decadimento esponenziale che viene utilizzato per limitare l'influenza dei vecchi dati. Manteniamo la stessa notazione tenendo presente che effettivamente  $\alpha = 1/T$ . Per un nuovo campione la proprietà  $o_m^{(t)}$  è impostata a 1 per la componente “vicina” con  $\hat{\pi}_m$  maggiore e le altre sono impostate a zero. Diciamo che un campione è “vicino” a un componente se la distanza Mahalanobis dal componente è, per esempio, inferiore a tre. La distanza al quadrato dalla componente  $m$  viene calcolata come:  $D_m^2(\vec{x}^{(t)}) = \vec{\delta}_m^T \vec{\delta}_m / \hat{\sigma}_m^2$ . Se non ci sono componenti “vicine”, viene generata una nuova componente con  $\hat{\pi}_{M+1} = \alpha, \hat{\mu}_{M+1} = \vec{x}^{(t)}$  e  $\hat{\sigma}_{M+1} = \sigma_0$ , dove  $\sigma_0$  è una varianza iniziale appropriata. Se si raggiunge il numero massimo di componenti, si scarta la componente con il valore  $\hat{\pi}_m$  più basso.

L'algoritmo presentato è un algoritmo di clustering on-line. Di solito, gli oggetti in primo piano intrusivi sono rappresentati da alcuni cluster aggiuntivi con pesi piccoli  $\hat{\pi}_m$ . Pertanto, possiamo approssimare il modello di background dai primi  $B$  cluster più grandi:

$$\hat{p}(\vec{x}|\mathcal{X}_T, BG) \sim \sum_{m=1}^B \hat{\pi}_m \mathcal{N}(\vec{x}; \hat{\mu}_m, \hat{\sigma}_m^2 I). \quad (2.7)$$

Se i componenti sono ordinati in modo da avere pesi decrescenti  $\hat{\pi}_m$ , si ha:

$$B = \arg \min_b \left( \sum_{m=1}^b \hat{\pi}_m > (1 - c_f) \right), \quad (2.8)$$

dove  $c_f$  è una misura della porzione massima di dati che può appartenere agli oggetti in primo piano senza influenzare il modello di background. Ad esempio, se un nuovo oggetto entra in scena e rimane statico per un certo periodo di tempo, verrà presentato temporalmente come un cluster aggiuntivo. Poiché il vecchio sfondo è occluso, il peso  $\pi_{B+1}$  del nuovo cluster aumenterà costantemente. Se l'oggetto rimane statico abbastanza a lungo, il suo peso diventa maggiore di  $c_f$  e può essere considerato parte dello sfondo. Se osserviamo l'equazione (2.4), possiamo concludere che l'oggetto deve rimanere statico per circa  $\log(1 - c_f)/\log(1 - \alpha)$  fotogrammi. Ad esempio, per  $c_f = 0.1$  e  $\alpha = 0.001$  otteniamo 105 fotogrammi.

**Selezione del numero dei componenti** Il peso  $\pi_m$  è la frazione dei dati che appartiene alla componente  $m$ th del GMM. Può essere considerato come la probabilità che un campione provenga dalla componente  $m$ th e in questo modo  $\pi_m$ -s definisce una distribuzione multinomiale di base. Supponiamo di avere  $t$  campioni di dati e che ognuno di essi appartenga a una delle componenti della GMM. Assumiamo inoltre che il numero di campioni che appartengono alla  $m$ -esima componente sia  $n_m = \sum_{i=1}^t o_m^{(i)}$ , dove  $i_m^{(i)}$ -s è definito nella sezione precedente. La distribuzione multinomiale ipotizzata per ogni  $n_m$ -s fornisce la funzione di verosimiglianza (likelihood)  $\mathcal{L} = \prod_{m=1}^M \pi_m^{n_m}$ . I pesi di mixing sono vincolati ad avere una somma pari a uno. Prendiamo in considerazione questo aspetto introducendo il moltiplicatore di Lagrange  $\mathcal{L}$ . La stima di Maximum Likelihood (ML) segue la seguente formula:  $\frac{\delta}{\delta \hat{\pi}_m} (\log \mathcal{L} + \lambda(\sum_{m=1}^M \hat{\pi}_m - 1)) = 0$ . Dopo aver eliminato la  $\lambda$ , otteniamo:

$$\hat{\pi}_m^{(t)} = \frac{n_m}{t} = \frac{1}{t} \sum_{i=1}^t o_m^{(i)}. \quad (2.9)$$

La stima di  $t$  campioni è indicata come  $\hat{\pi}_m^{(t)}$  e può essere riscritta in forma ricorsiva come funzione della stima  $\hat{\pi}_m^{(t-1)}$  per  $t = 1$  campioni e della proprietà  $o_m^{(t)}$  dell'ultimo campione:

$$\hat{\pi}_m^{(t)} = \hat{\pi}_m^{(t-1)} + \frac{1}{t} (o_m^{(t)} - \hat{\pi}_m^{(t-1)}). \quad (2.10)$$

Se ora fissiamo l'influenza dei nuovi campioni fissando  $1/t$  a  $\alpha = 1/T$ , otteniamo l'aggiornamento mostrato dall'equazione (2.4). L'influenza fissa dei nuovi campioni significa che ci si affida maggiormente ai nuovi campioni e che il contributo dei vecchi campioni viene ridotto in modo esponenziale, come detto in precedenza. La conoscenza preventiva per la distribuzione multinomiale può essere introdotta

utilizzando il suo priore coniugato, il priore di Dirichlet  $\mathcal{P} = \prod_{m=1}^M \pi_m^{c_m}$ . I coefficienti  $c_m$  hanno un'interpretazione significativa. Per la distribuzione multinomiale,  $c_m$  presenta l'evidenza preventiva (nel senso di massima a posteriori (MAP)) per la classe  $m$  - il numero di campioni che appartengono a quella classe a priori. Come descritto in (Zivkovic e van der Heijden, 2004 [10]), utilizziamo coefficienti negativi  $c_m = -c$ . L'evidenza preventiva negativa significa che accetteremo l'esistenza della classe  $m$  solo se i dati dimostrano l'esistenza di questa classe. Questo tipo di priori è anche legato al criterio della lunghezza minima del messaggio, utilizzato per selezionare i modelli corretti per i dati (Zivkovic e van der Heijden, 2004 [10]). La soluzione MAP che include il priore citato si ottiene da  $\frac{\delta}{\delta \hat{\pi}_m} (\log \mathcal{L} + \log \mathcal{P} + \lambda(\sum_{m=1}^M \hat{\pi}_m - 1)) = 0$ , dove  $\mathcal{P} = \sum_{m=1}^M \pi_m^{-c}$ . Si ottiene:

$$\hat{\pi}_m^{(t)} = \frac{1}{K} \left( \sum_{i=1}^t o_m^{(i)} - c \right), \quad (2.11)$$

dove  $K = \sum_{m=1}^M \left( \sum_{i=1}^t o_m^{(i)} - c \right) = t - Mc$ . Riscriviamo l'equazione (2.11) come:

$$\hat{\pi}_m^{(t)} = \frac{\hat{\Pi}_m - c/t}{1 - Mc/t}, \quad (2.12)$$

dove  $\hat{\Pi}_m = \frac{1}{t} \sum_{i=1}^t o_m^{(i)}$  è la stima ML dall'equazione (2.9) e il bias dal priore è introdotto attraverso  $c/t$ . Il bias diminuisce per insiemi di dati più grandi ( $t$  più grande). Tuttavia, se un piccolo bias è considerato accettabile, possiamo mantenerlo costante fissando  $c/t$  a  $c_T = c/T$  con un certo valore grande di  $T$ . Ciò significa che il bias sarà sempre lo stesso che sarebbe stato per un insieme di dati con  $T$  campioni. È facile dimostrare che la versione ricorsiva dell'equazione (2.11) con  $c/t = c_T$  fissato è data da:

$$\hat{\pi}_m^{(t)} = \hat{\pi}_m^{(t-1)} + 1/t \left( \frac{o_m^{(t)}}{1 - Mc_T} - \hat{\pi}_m^{(t-1)} \right) - 1/t \frac{c_T}{1 - Mc_T}. \quad (2.13)$$

Poiché di solito ci aspettiamo solo pochi componenti  $M$  e  $c_T$  è piccolo, assumiamo  $1 - Mc_T \approx 1$ . Come detto, impostiamo  $1/t$  a  $\alpha$  e otteniamo l'equazione finale di aggiornamento adattativo modificato:

$$\hat{\pi}_m \leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m) - \alpha c_T \quad (2.14)$$

Questa equazione viene utilizzata al posto della (2.4). Dopo ogni aggiornamento è necessario normalizzare le componenti  $\pi_m$  in modo che la loro somma sia pari a uno. Si inizia con un GMM con una componente centrata sul primo campione e si aggiungono nuove componenti come indicato nella sezione precedente. Il priore di Dirichlet con pesi negativi sopprime le componenti che non sono confermate dai

dati e scartiamo la componente  $m$  quando il suo peso  $\pi_m$  diventa negativo. Questo assicura anche che i pesi di mixing rimangano non negativi. Per una scelta  $\alpha = 1/T$ , potremmo richiedere che almeno  $c = 0,01 * T$  campioni confermino una componente e otterremo  $c_T = 0,01$ .

Si noti che la versione ricorsiva diretta dell'espressione (2.11) data da  $\hat{\pi}_m^{(t)} = \hat{\pi}_m^{(t-1)} + (t - Mc)^{-1}(o_m^{(t)}(\vec{x}^{(t)}) - \hat{\pi}_m^{(t-1)})$  non è molto utile. Potremmo iniziare con un valore più grande per  $t$  per evitare un aggiornamento negativo per valori piccoli di  $t$ , ma in questo modo annulliamo l'influenza del priore. Questo motiva l'importante scelta di fissare l'influenza del priore.

### Metodi non parametrici

**Kernel density estimation** La stima della densità utilizzando un kernel uniforme consiste nel contare il numero di campioni  $k$  dell'insieme di dati  $\mathcal{X}_T$  che si trovano all'interno del volume  $V$  del kernel. Il volume  $V$  è un'ipersfera con diametro  $D$ . La stima della densità è data da:

$$\hat{p}_{non-parametric}(\vec{x}|\mathcal{X}_T, BG + FG) = \frac{1}{TV} \sum_{m=t-T}^t \mathcal{K}\left(\frac{\|\vec{x}^{(m)} - \vec{x}\|}{D}\right) = \frac{k}{TV}, \quad (2.15)$$

dove la funzione kernel è  $\mathcal{K}(u) = 1$  se  $u < 1/2$  e 0 altrimenti. Il volume  $V$  del kernel è proporzionale a  $D^d$ , dove  $d$  è la dimensionalità dei dati. Spesso vengono utilizzate altre funzioni kernel  $\mathcal{K}$  più morbide. Ad esempio, in (Elgammal et al., 2000 [6]) viene utilizzato un profilo gaussiano. In pratica, la forma del kernel  $\mathcal{K}$  ha poca influenza, ma la scelta di  $D$  è fondamentale. In (Elgammal et al., 2000 [6]) viene calcolata la mediana med per le differenze assolute  $\|\vec{x}^{(t)} - \vec{x}^{(t-1)}\|$  dei campioni di  $\mathcal{X}_T$  e viene utilizzata una semplice stima robusta della deviazione standard del tipo  $D = \text{med}/(0.68)\sqrt{2}$ .

**Semplice stima della densità del kernel variabile a palloncino** La stima del kernel utilizza una dimensione fissa del kernel  $D$  per l'intera funzione di densità, che potrebbe non essere la scelta migliore. Il cosiddetto "stimatore a palloncino" adatta la dimensione del kernel a ogni punto di stima  $\vec{x}$ . Invece di cercare di trovare la  $D$  ottimale a livello globale, si può aumentare l'ampiezza  $D$  del kernel per ogni nuovo punto  $\vec{x}$  fino a coprire una quantità fissa di dati  $k$ . In questo modo si ottengono kernel grandi nelle aree con un numero ridotto di campioni e kernel più piccoli nelle aree densamente popolate. Questa stima non è una vera e propria stima di densità poiché l'integrale della stima non è uguale a 1. Esistono molti altri approcci più elaborati (Hall et al., 1995 [11]). Tuttavia, la stima del palloncino è spesso utilizzata per i problemi di classificazione, poiché è legata alla classificazione k-NN. One nearest neighbor è il metodo più utilizzato, ma per essere più robusti agli outlier si usa  $k = [0,1T]$  dove  $[.]$  è l'operatore "round-to-integer".

L’approccio a palloncino porta a un’implementazione efficiente, equivalente all’utilizzo di un kernel uniforme fisso. Solo la scelta della soglia  $c_{thr}$  dall’equazione (2.2) cambia. Sia per il kernel fisso che per la stima a palloncino, la decisione se un nuovo campione  $\vec{x}$  si adatta al modello viene presa se ci sono più di  $k$  punti all’interno del volume  $V$  (2.15). L’approccio basato sul kernel ha  $V$  fisso e  $k$  è il parametro variabile che può essere utilizzato come soglia  $c_{thr} \sim k$  dall’equazione (2.2). Per il kernel uniforme  $k$  è discreto e si ottengono stime discontinue. L’approccio con kernel variabile a palloncino ha  $k$  fisso e il volume  $V$  è il parametro variabile  $c_{thr} \sim 1/V \sim 1/D^d$ . I problemi di discontinuità non si verificano. Un ulteriore vantaggio è che non si stima il parametro sensibile della dimensione del kernel come in (Elgammal et al., 2000 [6]).

## 2.4 Laplaciano e il suo utilizzo nel Blur Detection

Il Laplaciano di una funzione  $f$  in un punto  $p$  è la velocità con cui il valore medio di  $f$  su sfere centrate in  $p$  si discosta da  $f(p)$  man mano che il raggio della sfera si restringe verso 0. Di solito è indicato con i simboli  $\nabla \cdot \nabla$ ,  $\nabla^2$  (dove  $\nabla$  è l’operatore nabla).

Più semplicemente, l’operatore lapliciano è definito come la divergenza del gradiente di una funzione  $f$ .

$$\Delta f(x, y) = \text{div}(\text{grad}(f)) \quad (2.16)$$

Il gradiente è la pendenza del punto più ripido. Fondamentalmente, fornisce informazioni sulla direzione verso cui il punto (nella curva) dovrebbe dirigersi per raggiungere il punto più alto, cioè il massimo locale. Allo stesso modo, il gradiente negativo fornisce la direzione dei minimi locali.

Ad esempio, se consideriamo il grafico generato dalla funzione  $z = x \cdot \exp(-x^2 + y^2)$  e il gradiente in ogni punto della curva (rappresentato dalla linea con la freccia), possiamo ottenere una figura come la seguente:

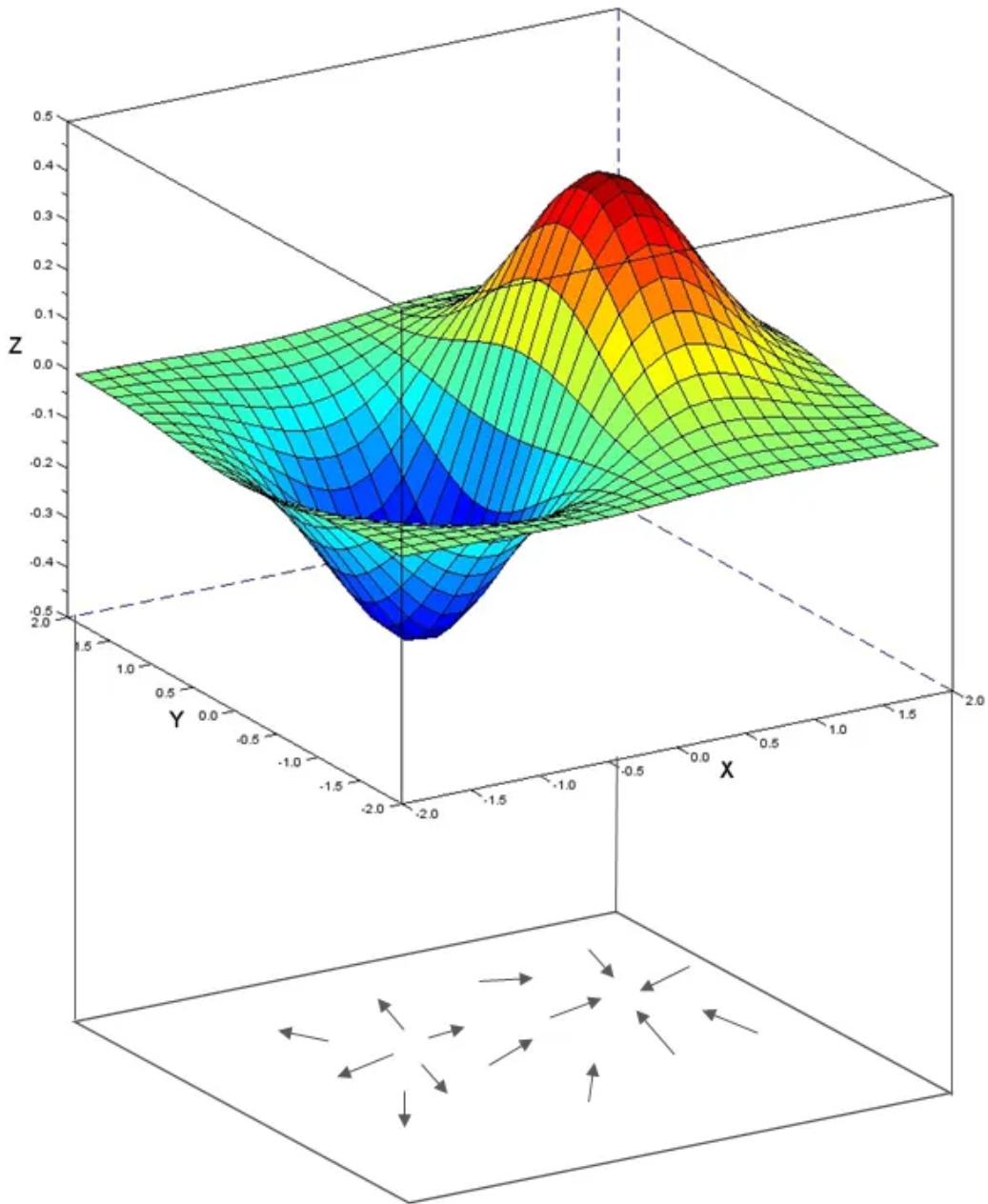


Figura 2.8: Grafico della funzione  $z = x \cdot \exp(-x^2 + y^2)$  che illustra la sua pendenza in vari punti della curva.

Poiché il gradiente dà la direzione della massima pendenza, possiamo osservare che tutte le frecce (vettori) puntano verso la collina e si allontanano dalla valle. Anche se può non essere chiaro dalla figura, la grandezza del vettore, cioè la lunghezza della freccia, ci dà un'idea di quanto sia ripida la curva in quel particolare

punto. Possiamo intendere la divergenza come il campo vettoriale corrispondente a un qualche tipo di movimento generato dal flusso di un fluido. Nella collina del grafico, quando ogni campo vettoriale punta verso la regione, la divergenza è negativa, mentre nella valle, dove ogni vettore punta verso l'esterno, la divergenza è positiva.

Quindi, in un certo senso, l'operatore Lapliciano è una sorta di misura di quanto è minimo un punto  $x, y$ . Si comporta quindi come la derivata seconda per una funzione multivariabile con valore scalare. La derivata seconda può essere utilizzata per determinare gli estremi locali di una funzione. Se una funzione ha un punto critico  $f'(x) = 0$  su cui la derivata seconda del punto è  $f''(x) < 0$ , cioè negativa, allora  $f$  ha un massimo locale nel punto. Questo può essere ulteriormente illustrato con la figura seguente:

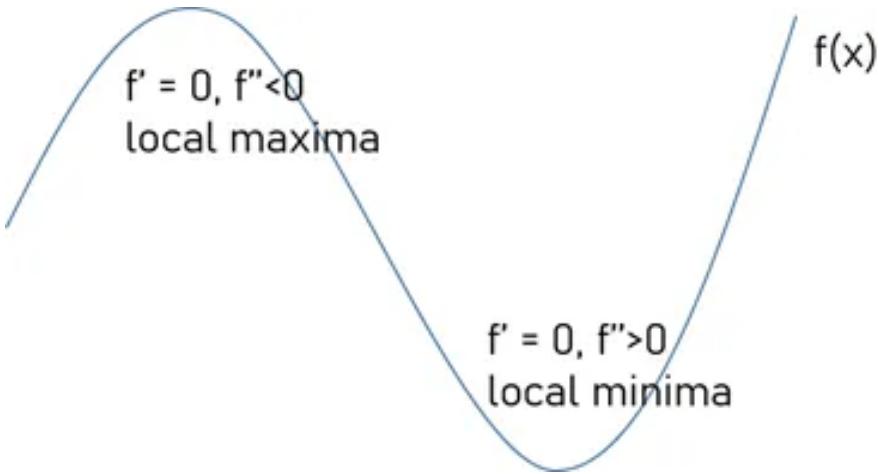


Figura 2.9: Minimi e massimi locali per la funzione  $f(x)$ .

L'operatore Laplaciano si ottiene mediante la formula:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y} \quad (2.17)$$

E la derivata parziale nella direzione  $x$  può essere data come:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y) \quad (2.18)$$

Analogamente, la derivata parziale nella direzione  $y$  può essere definita come:

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y) \quad (2.19)$$

Combinando queste equazioni per le derivate parziali, si ottiene:

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y) \quad (2.20)$$

Consideriamo ora l'aspetto di un filtro. Un filtro  $3 \times 3$  per un'immagine può essere rappresentato per coordinate  $(x, y)$  e può essere rappresentato nel seguente modo:

$x-1, y-1$	$x, y-1$	$x+1, y-1$
$x-1, y$	$x, y$	$x, y+1$
$x-1, y+1$	$x, y+1$	$x+1, y+1$

Se si inseriscono i coefficienti dell'equazione nell'immagine, si ottiene la seguente matrice.

0	1	0
1	-4	1
0	1	0

Questa matrice, se convoluta con l'immagine, fornisce un'immagine trasformata basata su Laplician. Questa matrice viene utilizzata per trovare l'area di rapidi cambiamenti nelle immagini in assenza di rumore. Fondamentalmente l'operatore prende la derivata seconda dell'immagine su un piano di dimensioni superiori. Se l'immagine è sostanzialmente uniforme, il risultato sarà pari a zero. Ovunque si verifichi un cambiamento, la matrice risultante avrà elementi positivi sul lato più scuro e un elemento negativo sul lato più luminoso.

L'operazione di convoluzione dell'immagine dà il seguente risultato:

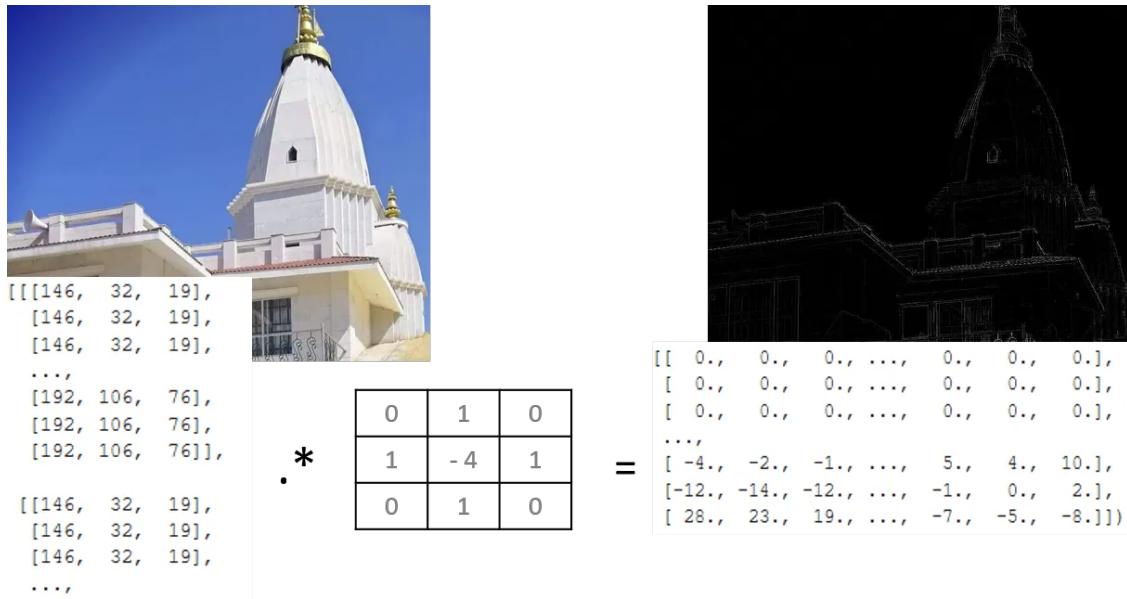


Figura 2.10: Operazione di convoluzione con l'operatore di Laplace su un'immagine.

## 2.5 Elaborazione morfologica delle immagini

Le operazioni morfologiche sono definite in termini di insiemi. Nell'elaborazione delle immagini [15], utilizziamo la morfologia con due tipi di insiemi di pixel: gli oggetti e gli elementi strutturanti (SE). In genere, gli oggetti sono definiti come insiemi di pixel in primo piano. Gli elementi strutturanti possono essere specificati in termini di pixel in primo piano e di pixel di sfondo. Inoltre, gli elementi strutturanti a volte contengono i cosiddetti elementi "don't care", indicati con  $\times$ , che indicano che il valore di quel particolare elemento nel SE non è importante. In questo caso, il valore può essere ignorato, oppure può essere adattato a un valore desiderato nella valutazione di un'espressione; ad esempio, può assumere il valore di un pixel di un'immagine in applicazioni in cui l'obiettivo è la corrispondenza dei valori.

Poiché le immagini con cui lavoriamo sono matrici rettangolari e gli insiemi in generale sono di forma arbitraria, le applicazioni della morfologia nell'elaborazione delle immagini richiedono che gli insiemi siano inglobati in matrici rettangolari. Nel formare tali matrici, assegniamo un valore di sfondo a tutti i pixel che non sono membri di insiemi di oggetti. La riga superiore della figura 2.11 mostra un esempio. A sinistra ci sono gli insiemi nel formato grafico che siamo abituati a vedere nelle figure dei libri. Al centro, gli insiemi sono stati inseriti in uno sfondo rettangolare (bianco) per formare un'immagine grafica. A destra, invece, viene mostrata un'immagine digitale (si noti la griglia), che è il formato utilizzato per l'elaborazione delle immagini digitali.

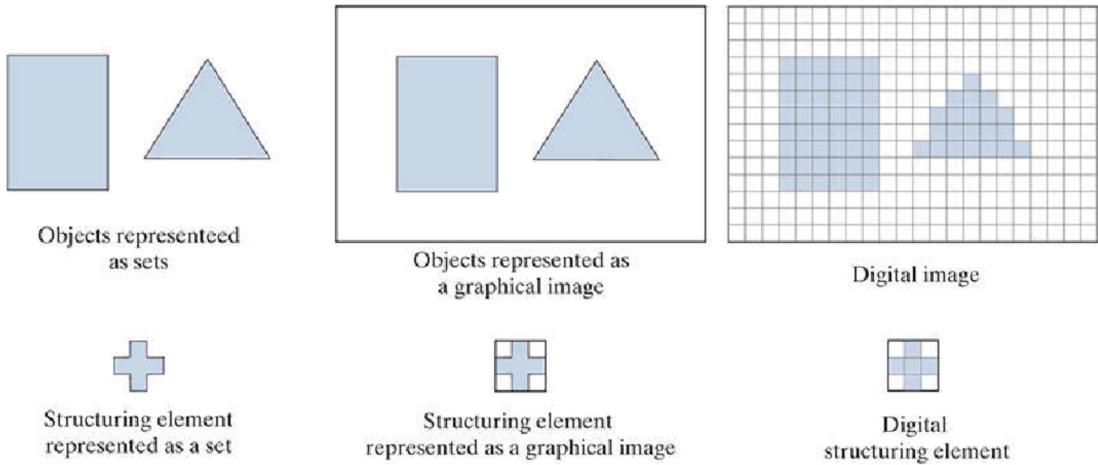


Figura 2.11: In alto. A sinistra: Gli oggetti rappresentati come insiemi grafici. Al centro: Gli oggetti incorporati in uno sfondo per formare un'immagine grafica. A destra: L'oggetto e lo sfondo vengono digitalizzati per formare un'immagine digitale (notare la griglia). Seconda riga: Esempio di elemento strutturante (SE) rappresentato come insieme, immagine grafica e infine come SE digitale.

Gli elementi strutturanti sono definiti allo stesso modo e la seconda riga della figura 2.11. Un aspetto significativo è che gli elementi strutturanti vengono utilizzati in una forma simile ai kernel di convoluzione spaziale. Le operazioni sono diverse nella morfologia, ma le operazioni di padding e di scorrimento sono le stesse della convoluzione. Abbiamo 4 principali operazioni morfologiche:

- Erosione
- Dilatazione
- Apertura
- Chiusura

### 2.5.1 Erosione

Le espressioni morfologiche si scrivono in termini di elementi strutturanti e di un insieme,  $A$ , di pixel in primo piano, oppure in termini di elementi strutturanti e di un'immagine,  $I$ , che contiene  $A$ . Consideriamo prima il primo approccio. Con  $A$  e  $B$  come insiemi in  $Z^2$ , l'erosione di  $A$  da parte di  $B$ , indicata con  $A \ominus B$ , è definita come:

$$A \ominus B = \{z | (B)_z \subseteq A\} \quad (2.21)$$

dove  $A$  è l'insieme dei pixel che si trovano in primo piano,  $B$  è un elemento strutturante e gli  $z$  sono valori in primo piano (gli 1). In parole povere, questa equazione indica che l'erosione di  $A$  da parte di  $B$  è l'insieme di tutti i punti  $z$  tali che  $B$ , traslato di  $z$ , è contenuto in  $A$ .

Dal momento che l'affermazione che  $B$  deve essere contenuto in  $A$  è equivalente alla condizione che  $B$  non condivida alcun elemento comune con lo sfondo (cioè il complemento dell'insieme  $A$ ), possiamo definire l'erosione in modo equivalente come:

$$A \ominus B = \{z | (B)_z \cap A^c = \emptyset\} \quad (2.22)$$

dove  $\emptyset$  è l'insieme vuoto. La Figura 2.12 mostra un esempio di erosione. Gli elementi dell'insieme  $A$  (ombreggiati) sono i pixel in primo piano dell'immagine  $I$  e, come in precedenza, lo sfondo è rappresentato in bianco. Il bordo continuo all'interno del bordo tratteggiato in Figura 2.12(C) è il limite oltre il quale ulteriori spostamenti dell'origine di  $B$  causerebbero che alcuni elementi dell'elemento strutturante cessino di essere completamente contenuti in  $A$ .

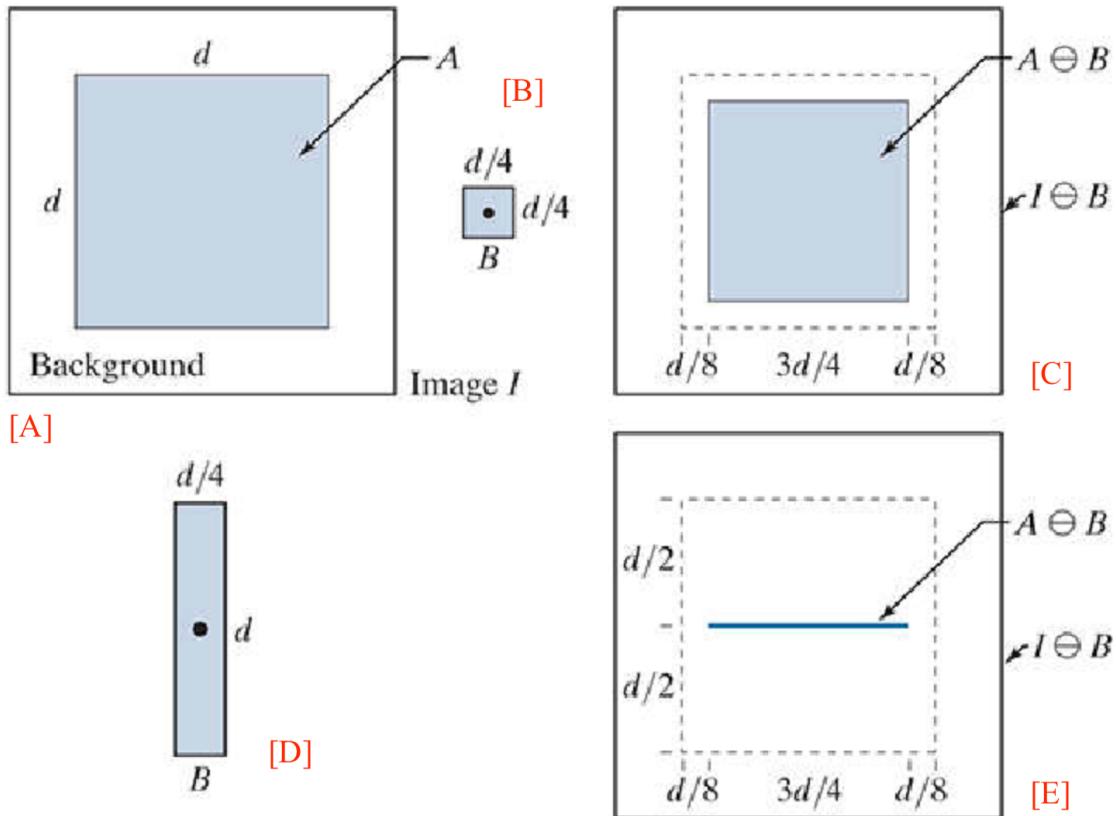


Figura 2.12: (A) Immagine  $I$ , composta da un insieme (oggetto)  $A$  e dallo sfondo. (B) Quadrato SE,  $B$  (il punto è l'origine). (C) Erosione di  $A$  da parte di  $B$  (mostrato ombreggiato nell'immagine risultante). (D) SE allungato. (E) Erosione di  $A$  da parte di  $B$ .

La Figura 2.13 in alto a sinistra mostra un'immagine binaria di una semplice maschera wire-bond. Come già detto, in genere nelle immagini binarie i pixel in primo piano sono bianchi e lo sfondo è nero. Supponiamo di voler rimuovere le linee che collegano la regione centrale ai rilievi di confine riportati.

L'erosione dell'immagine (cioè l'erosione dei pixel in primo piano dell'immagine) con un elemento strutturante quadrato di dimensioni  $11 \times 11$  i cui componenti sono tutti 1 ha rimosso la maggior parte delle linee, come mostra la Figura 2.13 in alto a destra. Il motivo per cui le due linee verticali al centro sono state assottigliate ma non rimosse completamente è che la loro larghezza è superiore a 11 pixel. Modificando le dimensioni di SE a  $15 \times 15$  elementi ed erodendo nuovamente l'immagine originale, si ottiene la rimozione di tutte le linee di collegamento, come mostra la Figura 2.13 in basso a sinistra.

Un approccio alternativo consiste nell'erodere nuovamente l'immagine di Figura

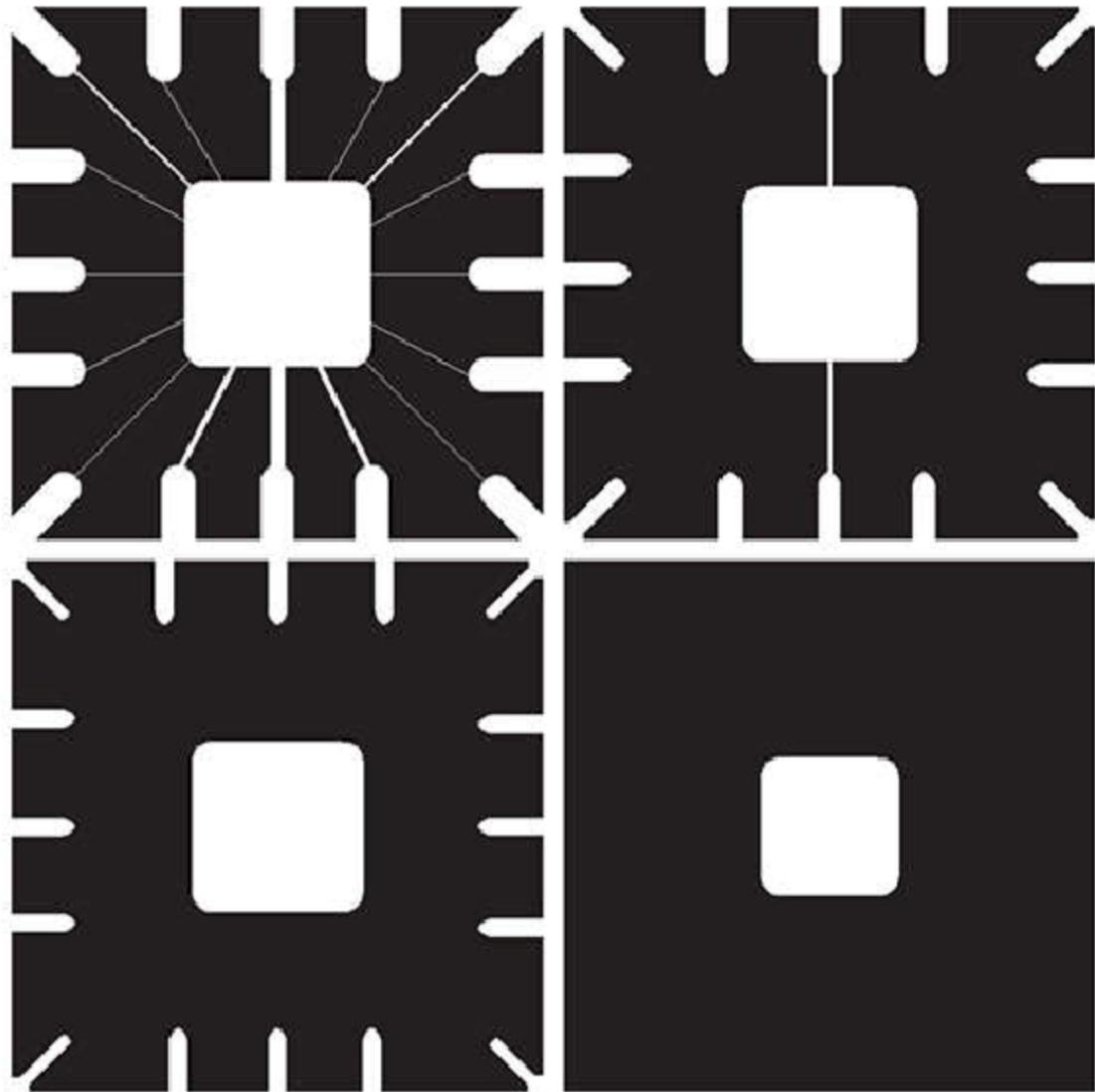


Figura 2.13: Utilizzo dell’erosione per rimuovere i componenti dell’immagine. **(A)** Immagine binaria  $486 \times 486$  di una maschera wire-bond in cui i pixel in primo piano sono mostrati in bianco. **(B)-(D)** Immagine erosa utilizzando elementi strutturanti quadrati di dimensioni  $11 \times 11$ ,  $15 \times 15$  e  $45 \times 45$  elementi, rispettivamente, tutti con valore 1.

[2.13](#) in alto a destra, utilizzando lo stesso SE o un SE più piccolo. Aumentando ulteriormente le dimensioni dell’elemento strutturante si eliminerebbero le componenti più grandi. Ad esempio, le linee di collegamento e i bordi possono essere rimossi con un elemento strutturante di dimensioni  $45 \times 45$  elementi applicato all’immagine originale, come mostra la Figura [2.13](#) in basso a destra.

### 2.5.2 Dilatazione

Dati  $A$  e  $B$  come insiemi in  $Z^2$ , la dilatazione di  $A$  per  $B$ , indicata come  $A \oplus B$ , è definita come:

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\} \quad (2.23)$$

A differenza dell'erosione, che è un'operazione di riduzione o assottigliamento, la dilatazione fa "crescere" o "addensare" gli oggetti in un'immagine binaria. Il modo e l'entità di questo ispessimento sono controllati dalla forma e dalle dimensioni dell'elemento strutturante utilizzato. La Figura 2.14(A) mostra lo stesso oggetto utilizzato nella Figura 2.12 (l'area dello sfondo è più grande per adattarsi alla dilatazione) e la Figura 2.14(B) mostra un elemento strutturante (in questo caso  $\hat{B} = B$  perché il SE è simmetrico rispetto all'origine).

La linea tratteggiata nella Figura 2.14(C) mostra il bordo dell'oggetto originale come riferimento, mentre la linea continua mostra il limite oltre il quale qualsiasi ulteriore spostamento dell'origine di  $\hat{B}$  di  $z$  causerebbe un'intersezione vuota tra  $\hat{B}$  e  $A$ . Pertanto, tutti i punti all'interno di questo confine costituiscono la dilatazione di  $A$  per  $B$ . La Fig. 9.6(d) mostra un elemento strutturante progettato per ottenere una maggiore dilatazione verticale rispetto a quella orizzontale. Pertanto, tutti i punti su e all'interno di questo confine costituiscono la dilatazione di  $A$  per  $B$ . La Figura 2.14(D) mostra un elemento strutturante progettato per ottenere una dilatazione maggiore in verticale che in orizzontale e la Figura 2.14(E) mostra la dilatazione ottenuta con questo elemento.

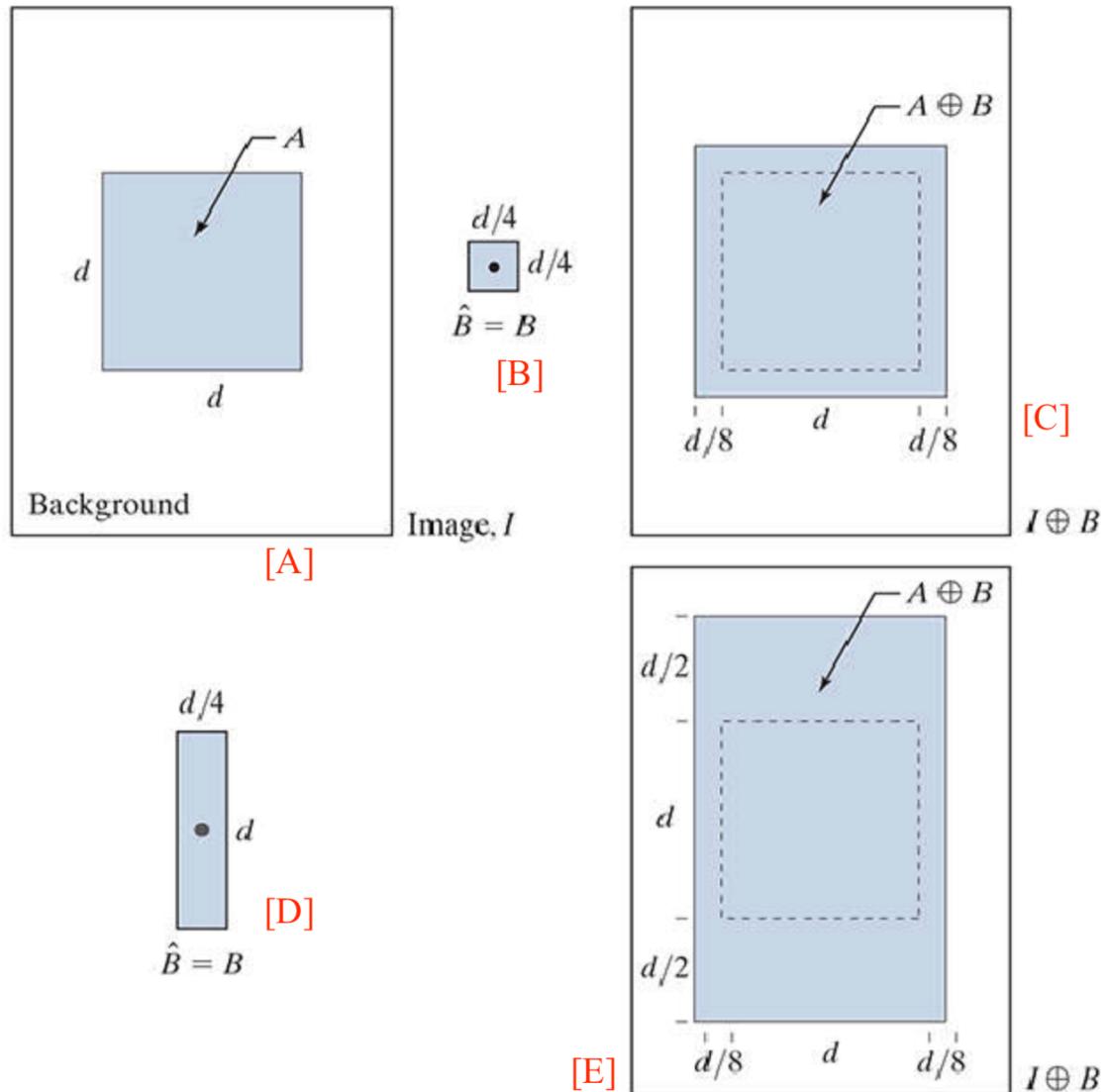


Figura 2.14: **(A)** Immagine  $I$ , composta dall’insieme (oggetto)  $A$  e dallo sfondo. **(B)** Quadrato SE (il punto è l’origine). **(C)** Dilatazione di  $A$  mediante  $B$  (mostrato in ombra). **(D)** SE allungato. **(E)** Dilatazione di  $A$  per tale elemento. La linea tratteggiata in **(C)** ed **(E)** è il confine di  $A$ , indicato come riferimento.

### 2.5.3 Apertura e Chiusura

Come si è visto nella sezione precedente, la dilatazione espande i componenti di un insieme e l’erosione li restringe. In questa sezione vengono discusse altre due importanti operazioni morfologiche: l’apertura e la chiusura. L’apertura generalmente leviga il contorno di un oggetto, interrompe gli spazi stretti ed elimina le sporgenze sottili. Anche la chiusura tende a levigare le sezioni dei contorni, ma,

a differenza dell'apertura, in genere fonde le interruzioni strette e i lunghi e sottili divari, elimina i piccoli buchi e riempie le lacune del contorno.

L'apertura dell'insieme  $A$  tramite l'elemento strutturante  $B$ , definita con  $A \circ B$ , è definita come:

$$A \circ B = (A \ominus B) \oplus B \quad (2.24)$$

Quindi, l'apertura di  $A$  da parte di  $B$  è l'erosione di  $A$  da parte di  $B$ , seguita da una dilatazione del risultato da parte di  $B$ .

Analogamente, la chiusura dell'insieme  $A$  tramite l'elemento strutturante  $B$ , indicata con  $A \bullet B$ , è definita come:

$$A \bullet B = (A \oplus B) \ominus B \quad (2.25)$$

la quale afferma che la chiusura di  $A$  da parte di  $B$  è semplicemente la dilatazione di  $A$  da parte di  $B$ , seguita da un'erosione del risultato da parte di  $B$ .

L'equazione 2.24 ha una semplice interpretazione geometrica: L'apertura di  $A$  da parte di  $B$  è l'unione di tutte le traslazioni di  $B$  tali che  $B$  si inserisca interamente in  $A$ . La Figura 2.15(A) mostra un'immagine contenente un insieme (oggetto)  $A$  e la Figura 2.15(B) è un elemento strutturante solido e circolare,  $B$ . La Figura 2.15(C) mostra alcune delle traslazioni di  $B$  tali da essere contenute in  $A$ , e l'insieme mostrato ombreggiato nella Figura 2.15(D) è l'unione di tutte queste possibili traslazioni. Si noti che, in questo caso, l'apertura è un insieme composto da due sottoinsiemi disgiunti, derivanti dal fatto che  $B$  non poteva entrare nel segmento stretto al centro di  $A$ . Come si vedrà tra breve, la capacità di eliminare le regioni più strette dell'elemento strutturante è una delle caratteristiche principali dell'apertura morfologica.

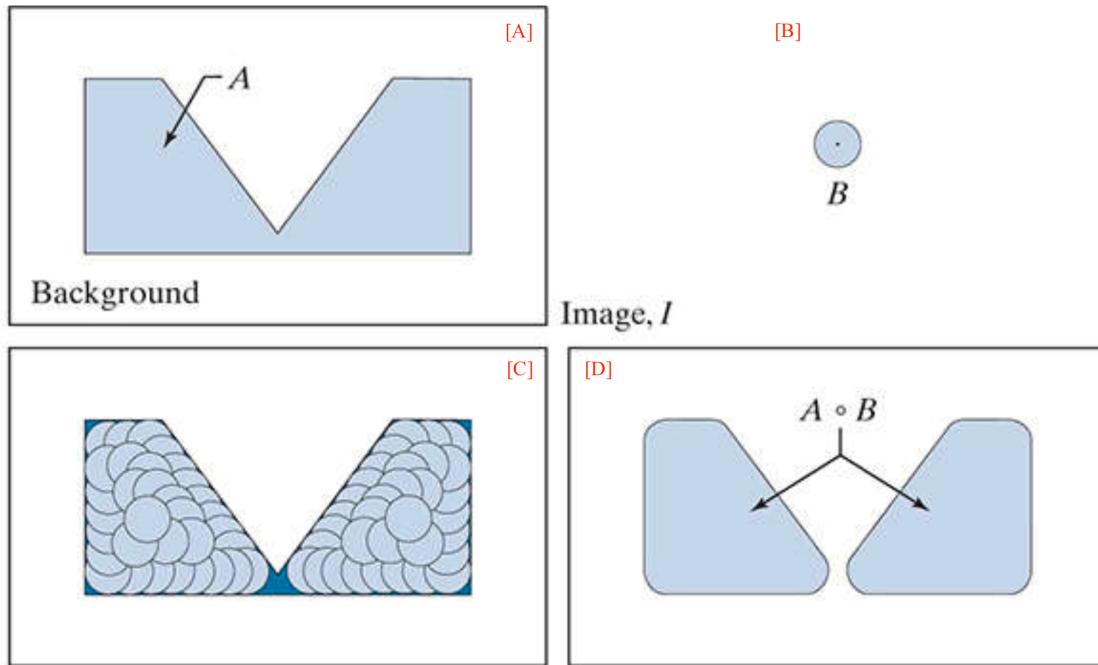


Figura 2.15: **(A)** Immagine  $I$ , composta dall’insieme (oggetto)  $A$  e dallo sfondo. **(B)** Elemento strutturante,  $B$ . **(C)** Traslazione di  $B$  contenuto in  $A$  ( $A$  è mostrato scuro per chiarezza). **(D)** Apertura di  $A$  da parte di  $B$ .

Quando si utilizza un elemento strutturante circolare per l’apertura, si fa spesso l’analogia con la forma dell’apertura determinata da una "palla che rotola" che raggiunge il più possibile il confine interno di un insieme. Per la chiusura morfologica, la palla rotola all’esterno e la forma della chiusura è determinata da quanto la palla può raggiungere il confine.

L’interpretazione secondo cui l’apertura di  $A$  da parte di  $B$  è l’unione di tutte le traslazioni di  $B$  tali da far rientrare  $B$  interamente in  $A$  può essere scritta in forma di equazione come:

$$A \circ B = \cup\{(B)_z | (B)_z \subseteq A\} \quad (2.26)$$

dove  $\cup$  indica l’unione degli insiemi all’interno delle parentesi graffe.

La chiusura ha un’interpretazione geometrica simile, tranne per il fatto che ora trasliamo  $B$  all’esterno di  $A$ . La chiusura è quindi il complemento dell’unione di tutte le traslazioni di  $B$  che non si sovrappongono ad  $A$ . La Figura 2.16 illustra questo concetto. Si noti che il bordo della chiusura è determinato dai punti più lontani che  $B$  può raggiungere senza entrare in nessuna parte di  $A$ . Sulla base di questa interpretazione, possiamo scrivere la chiusura di  $A$  da parte di  $B$  come:

$$A \bullet B = [\cup\{(B)_z | (B)_z \cap A = \emptyset\}]^c \quad (2.27)$$

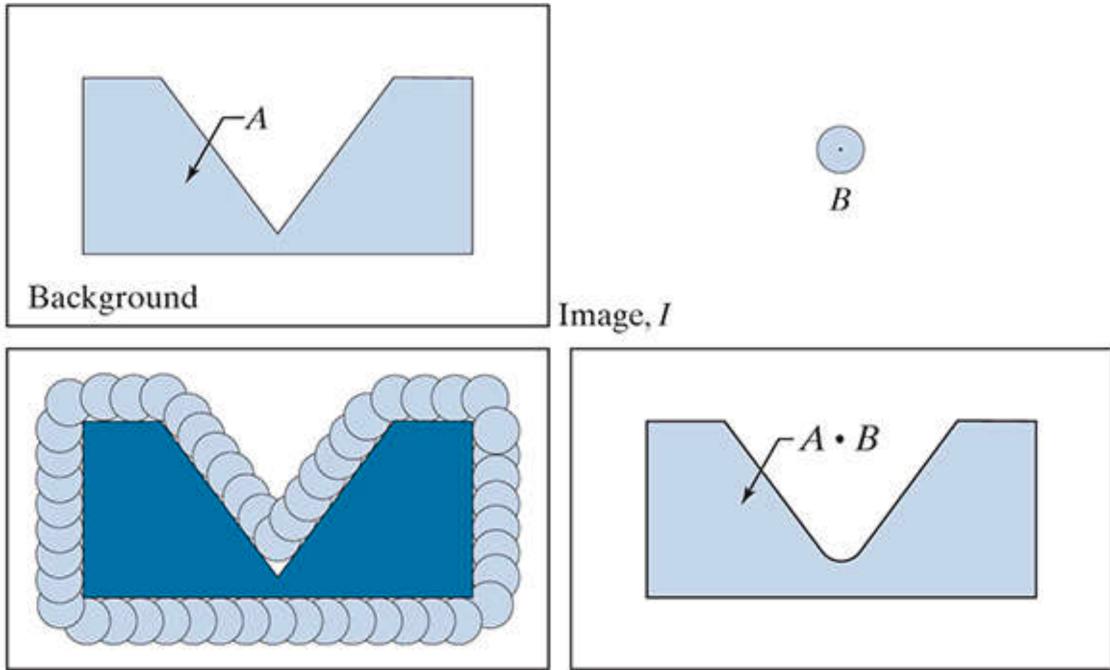
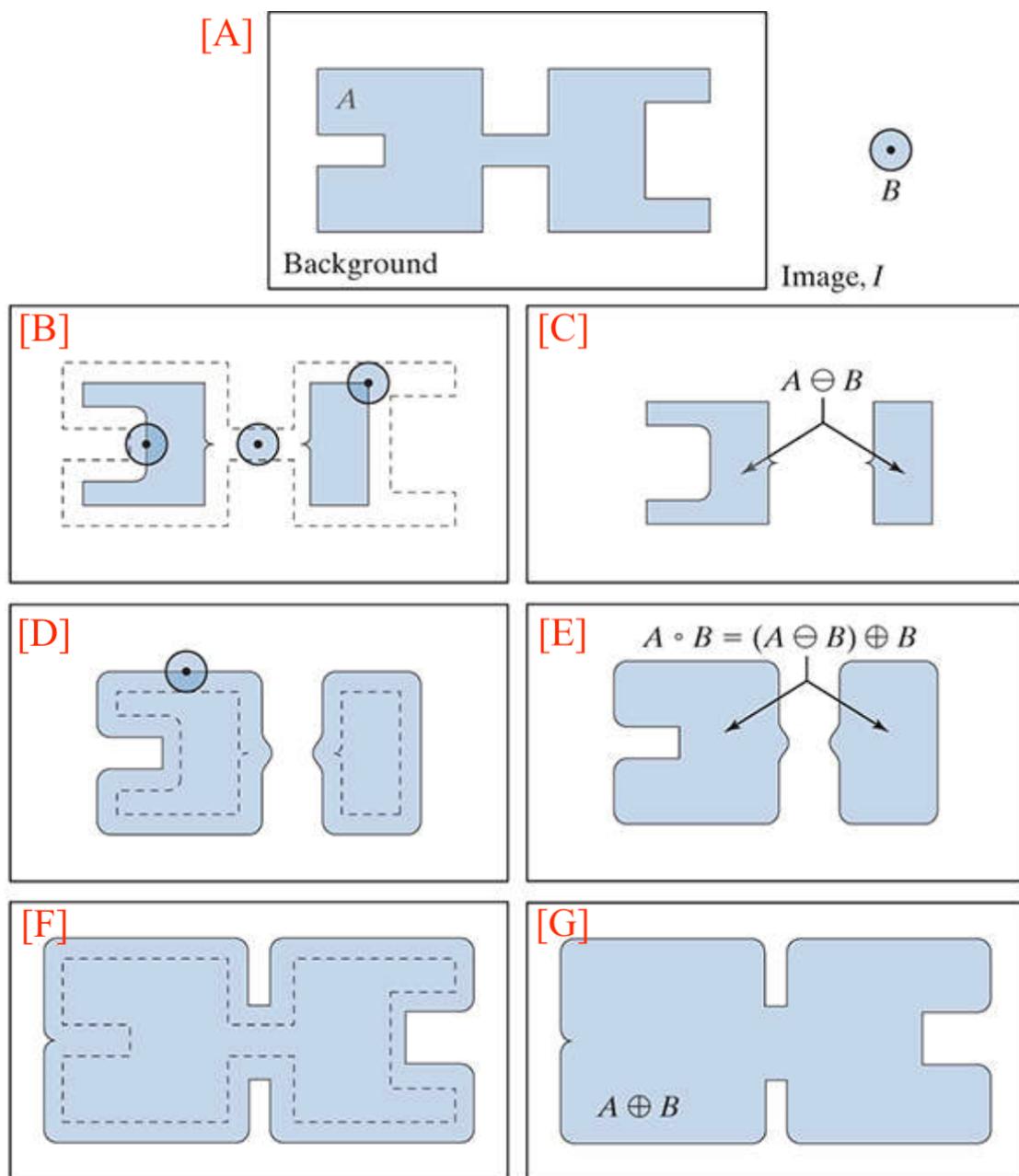


Figura 2.16: **(A)** Immagine  $I$ , composta dall'insieme (oggetto)  $A$  e dallo sfondo. **(B)** Elemento strutturante  $B$ . **(C)** Traslazione di  $B$  tale che  $B$  non si sovrapponga a nessuna parte di  $A$ . ( $A$  è mostrato scuro per chiarezza). **(D)** Chiusura di  $A$  da parte di  $B$ .

### Esempio

La Figura 2.17 mostra in modo più dettagliato il processo e le proprietà di apertura e chiusura. A differenza delle figure 2.15 e 2.16, i cui obiettivi principali sono le interpretazioni geometriche complessive, questa figura mostra i singoli processi e presta maggiore attenzione alla relazione tra la scala dei risultati finali e le dimensioni degli elementi strutturanti.



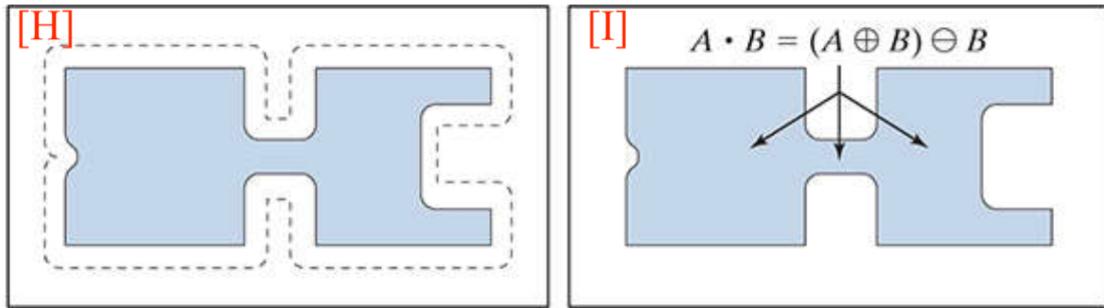


Figura 2.17: Apertura e chiusura morfologica. **(A)** Immagine  $I$ , composta da un insieme (oggetto)  $A$  e da uno sfondo; è mostrato anche un elemento strutturante solido e circolare. (Il punto è l'origine). **(B)** Elemento strutturante in varie posizioni. **(C)-(I)** Le operazioni morfologiche utilizzate per ottenere l'apertura e la chiusura.

La Figura 2.17(A) mostra un’immagine contenente un singolo oggetto (set)  $A$  e un elemento strutturante a disco. La Figura 2.17(B) mostra varie posizioni dell’elemento strutturante durante l’erosione. Questo processo ha dato origine all’insieme disgiunto della figura 2.17(C) . Si noti come il ponte tra le due sezioni principali sia stato eliminato. La sua larghezza era ridotta rispetto al diametro dell’elemento strutturante, che non poteva essere completamente contenuto in questa parte dell’insieme, violando così la definizione di erosione. Lo stesso vale per i due membri più a destra dell’oggetto. Gli elementi sporgenti in cui il disco non si adattava sono stati eliminati. La figura 2.17(D) mostra il processo di dilatazione dell’insieme eroso e la figura 2.17(E) mostra il risultato finale dell’apertura. L’apertura morfologica rimuove le regioni che non possono contenere l’elemento strutturante, leviga i contorni dell’oggetto, interrompe le connessioni sottili e rimuove le sporgenze sottili.

Le figure da 2.17(F) a (I) mostrano i risultati della chiusura di  $A$  con lo stesso elemento strutturante. Come nel caso dell’apertura, anche la chiusura leviga i contorni degli oggetti. Tuttavia, a differenza dell’apertura, la chiusura tende a unire le fratture strette, a riempire gli spazi lunghi e sottili e a riempire gli oggetti più piccoli dell’elemento strutturante. In questo esempio, il risultato principale della chiusura è stato quello di riempire il piccolo abisso a sinistra dell’insieme  $A$ .

Di seguito, la Figura 2.18, mostra il risultato dell’applicazione degli operatori di apertura e chiusura a due figure.



Figura 2.18: [a sinistra] Esempio di apertura. [a destra] Esempio di chiusura.



# Capitolo 3

## Sistema Proposto

In questo capitolo verrà descritto in maniera dettagliata il progetto spiegando le motivazioni che hanno portato alla realizzazione dello stesso.

Come detto in precedenza, il progetto in esame, svolto in collaborazione con la TIM presso i laboratori di Catania, consiste nella creazione di un dispositivo in grado di individuare e classificare i rifiuti con lo scopo di aiutare e supportare l'utente nella fase di riciclo. Possiamo distinguere due fasi all'interno del progetto: programmazione degli aspetti software (OpenCV, pytorch, ecc..), inserimento ed esecuzione di quest'ultimi all'interno di un Single Board Computer.

### 3.1 Aspetti Software

Prima di addentrarci nella descrizione degli aspetti software diamo uno sguardo a come opera l'algoritmo, Figura 3.1:

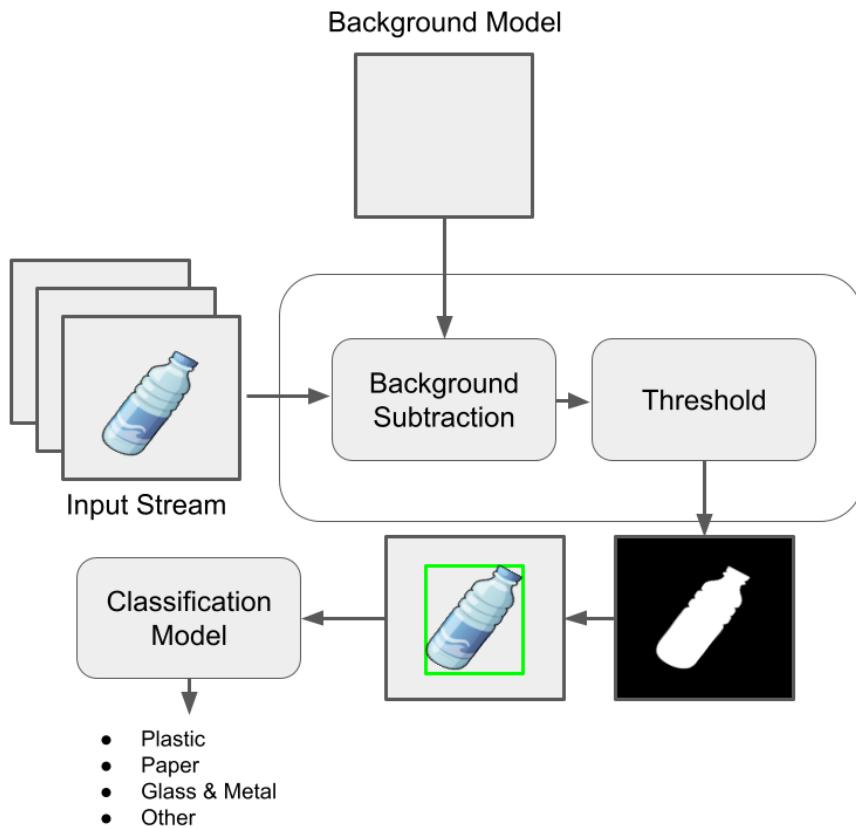


Figura 3.1: Pipeline del progetto

Dato un flusso video utilizziamo l'algoritmo di Background Subtraction per identificare il nuovo oggetto (rifiuto) apparso nella scena, l'algoritmo genera una maschera di primo piano (ovvero un'immagine binaria contenente i pixel appartenenti agli oggetti in movimento della scena). Si calcola, quindi, la maschera di foreground (primo piano) eseguendo una sottrazione tra il fotogramma corrente e un modello di sfondo, contenente la parte statica della scena o, più in generale, tutto ciò che può essere considerato come sfondo date le caratteristiche della scena osservata. Infine, l'immagine ritagliata, viene mandata come input ad una rete neurale la quale restituirà la classe di appartenenza del rifiuto.

### 3.1.1 Algoritmo di Background Subtractor

Per implementare l'algoritmo di BS utilizziamo la seguente riga di codice:

---

```
cv.createBackgroundSubtractorMOG2(varThreshold=5,
                                  detectShadows = False)
```

---

dove varThreshold=5 imposta una soglia di tolleranza per la corrispondenza tra pixel e modello. Mentre con detectShadows=False si disabilita il rilevamento delle ombre.

### 3.1.2 ROI - Region Of Interest

Per selezionare la regione di interesse (ROI) di un’immagine si utilizza il seguente metodo:

---

```
cv.selectROI(frame)
```

---

è possibile selezionare manualmente e con precisione l’area di interesse (ROI, region of interest) desiderata dall’immagine e quindi eseguire tutte le successive operazioni su quell’area specifica. La libreria NumPy svolge un ruolo fondamentale in questo procedimento, in quanto OpenCV utilizza NumPy come strumento di base per eseguire tutte le manipolazioni delle immagini. Alla funzione selectRoi() passeremo l’immagine (frame) come parametro e ciò che la funzione restituirà è una matrice di valori differenti che contengono le coordinate del punto in alto a sinistra dell’area selezionata e la larghezza e l’altezza della ROI (regione di interesse), Figura 3.2 . Il risultato è memorizzato sottoforma di array, i cui componenti sono:

---

```
[Top_Left_X, Top_Left_Y, Width, Height]
```

dove:

```
top_left_y = top_left_row = y1  
top_left_x = top_left_col = x1
```

---

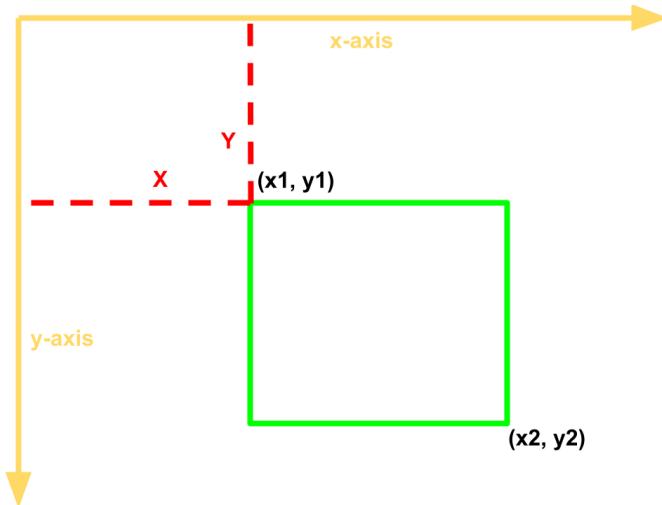


Figura 3.2: Roi

### 3.1.3 Opening (apertura) e Closing (chiusura)

Dopo aver computato la maschera di primo piano applichiamo due operatori morfologici: opening (apertura) e closing (chiusura). Come descritto nel paragrafo 2.5.3 del capitolo precedente, si applica l'operatore morfologico di apertura per rimuovere piccoli Blob da un'immagine. Mentre utilizziamo l'operatore morfologico di chiusura per chiudere i fori all'interno degli oggetti o per collegare tra loro i diversi elementi.

Applichiamo quindi in serie gli operatori sopracitati. In risultato è mostrato nella Figura 3.3:



Figura 3.3: **[A]** Mostra il risultato dell'applicazione della maschera di primo piano. **[B]** Mostra l'applicazione dell'operatore di apertura. **[C]** Mostra l'applicazione dell'operatore di chiusura.

Nell’immagine mostrata in Figura 3.3[A] sono presenti piccole macchioline bianche che rappresentano piccoli cambiamenti dello sfondo. Questi lievi cambiamenti andrebbero eliminati in quanto non fanno parte dell’oggetto che ci interessa evidenziare. Applicando il filtro di apertura, in effetti, eliminiamo questi piccoli elementi ottenendo l’immagine rappresentata in [B]. Nell’immagine [B] possiamo notare la presenza di piccoli crateri costituiti da diversi punti neri più o meno grandi. Tramite l’operatore di chiusura si rimuovono questi crateri ottenendo l’immagine in [C].

Gli operatori di apertura e chiusura si implementano mediante il seguente codice:

---

```
# Morphological opening and closing to improve mask
mask_morph = cv.morphologyEx(mask, cv.MORPH_OPEN,
                             cv.getStructuringElement(cv.MORPH_ELLIPSE, (21, 21)))
mask_morph = cv.morphologyEx(mask_morph, cv2.MORPH_CLOSE,
                             cv.getStructuringElement(cv.MORPH_ELLIPSE, (21, 21)))
```

---

In particolare con questo codice stiamo utilizzando un operatore di forma ellittica con un kernel di dimensione  $21 \times 21$  pixel.

### 3.1.4 Contorni

Una volta applicati gli operatori morfologici, è necessario un metodo che consideri solo gli oggetti più grandi. Per fare ciò utilizziamo il seguente metodo di OpenCV:

---

```
contours, _ = cv.findContours(mask_morph, cv.RETR_TREE,
                             cv.CHAIN_APPROX_SIMPLE)
for cnt in contours:
    #Calculate area and remove small elements
    area = cv.contourArea(cnt)
```

---

Per spiegare in modo esaustivo questo estratto di codice dobbiamo prima capire che cosa sono i contorni.

I contorni si possono definire semplicemente come una curva che unisce tutti i punti continui (lungo il confine), aventi lo stesso colore o la stessa intensità, ma per una maggiore accuratezza, è preferibile utilizzare immagini binarie. I contorni sono uno strumento utile per l’analisi delle forme e per il rilevamento e il riconoscimento degli oggetti.

La funzione `cv.findContours()` contiene tre argomenti: il primo è l’immagine di partenza, il secondo è la modalità di estrazione dei contorni, il terzo è il metodo di

approssimazione dei contorni. La funzione restituisce i contorni e la relativa struttura gerarchica. I contorni sono una lista Python di tutti i contorni dell’immagine. Ogni singolo contorno è un array Numpy di coordinate  $(x, y)$  dei punti di contorno dell’oggetto.

### Metodo di approssimazione dei contorni

A questo punto una domanda sorge spontanea: Che cosa indica il terzo argomento della funzione cv.findContours? In precedenza abbiamo detto che i contorni sono i confini di una figura con la stessa intensità, e memorizza le coordinate  $(x, y)$  del contorno di una figura. Tuttavia, memorizza tutte le coordinate? Questo aspetto è specificato dal metodo di approssimazione dei contorni. Se si passa cv.CHAIN\_APPROX\_NONE, vengono memorizzati tutti i punti del contorno. Mentre se si passa cv.CHAIN\_APPROX\_SIMPLE, rimuove tutti i punti ridondanti e comprime il contorno, risparmiando così memoria. Ad esempio, per poter rappresentare una linea retta non sono necessari tutti i punti che giacciono sulla linea, bensì solo i due estremi.

La seguente immagine, 3.4, di un rettangolo è una chiara dimostrazione di questa tecnica. È sufficiente disegnare un cerchio su tutte le coordinate dell’array dei contorni (disegnate in colore blu). La prima immagine mostra i punti ottenuti con cv.CHAIN\_APPROX\_NONE (734 punti) e la seconda immagine mostra il risultato ottenuto con cv.CHAIN\_APPROX\_SIMPLE (solo 4 punti).

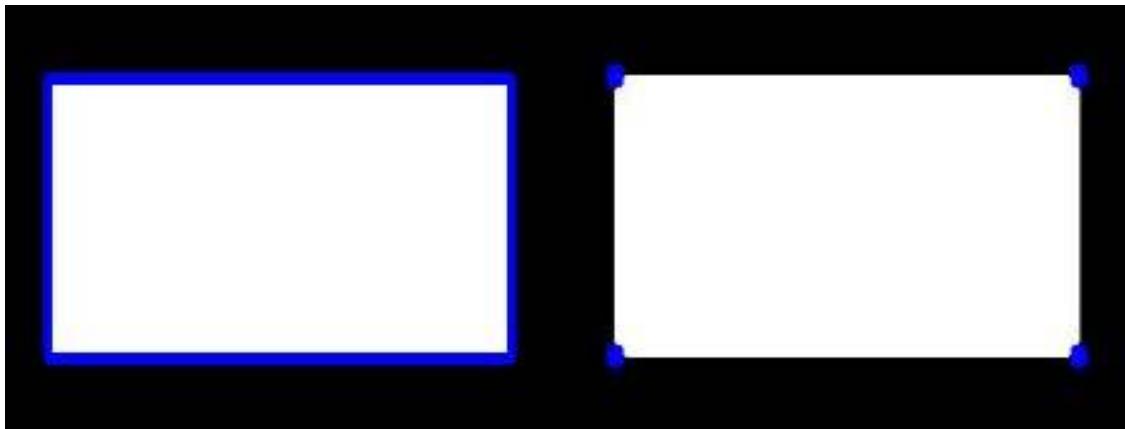


Figura 3.4: A sinistra l’immagine i cui bordi sono specificati tramite il parametro cv.CHAIN\_APPROX\_NONE. A destra l’immagine i cui bordi sono specificati tramite il parametro cv.CHAIN\_APPROX\_SIMPLE.

Da come è possibile notare c’è un evidente risparmio di memoria.

Per disegnare i contorni si utilizza la funzione cv.drawContours. Può essere utilizzata anche per disegnare qualsiasi forma, purché si disponga dei suoi punti

di delimitazione. Il suo primo parametro è l'immagine di partenza, il secondo parametro è il contorno che deve essere fornito come lista Python, il terzo parametro è l'indice del contorno (utile per disegnare un singolo contorno. Per disegnare tutti i contorni, passare -1) e i restanti parametri sono il colore, lo spessore ecc.

### 3.1.5 Blur detection

Come detto nel paragrafo 2.4 del capitolo precedente, possiamo utilizzare il Laplaciano per calcolare il livello di sfocatura nei frame e di scartarli se eventualmente presentano un livello elevato di sfocatura. Con la funzione:

---

```
def variance_of_laplacian(image):
    return cv.Laplacian(image, cv2.CV_64F).var()
```

---

Inizializziamo il metodo variance\_of\_laplacian che accetta un solo parametro, ovvero l'immagine (presumibilmente a un solo canale, ad esempio un'immagine in scala di grigi) per la quale si vuole calcolare la misura di messa a fuoco. Da qui, il metodo cv.Laplacian(image, cv2.CV\_64F).var() convolge semplicemente l'immagine con l'operatore laplaciano  $3 \times 3$  e restituisce la varianza.

Definiamo anche una soglia da utilizzare per il test di sfocatura. Se la misura della messa a fuoco di una data immagine scende al di sotto di questa soglia, l'immagine verrà contrassegnata come sfocata. Il valore di soglia va tarato in maniera empirica e può variare a seconda del dispositivo di acquisizione in uso. Per questo progetto è stato scelto un valore di soglia pari a 50.



Figura 3.5: [a sinistra] Immagine sfocata con valore di varianza basso, 8.32. [a destra] Immagine non sfocata con valore di varianza alto, 62.16. L'immagine di sinistra verrà scartata dall'algoritmo.

### 3.1.6 Metodo di classificazione

Una volta identificato il rifiuto entriamo nella fase di classificazione che, come è facilmente intuibile, ha lo scopo di restituirci un'etichetta con la classe di appartenenza del rifiuto. Come rete neurale ho utilizzato la ResNet-18 nella sua configurazione base (Figura 3.6).

ResNet-18 è una rete neurale convoluzionale a 18 strati di profondità. È possibile caricare una versione pre-addestrata della rete, ed è addestrata su oltre un milione di immagini del database ImageNet<sup>1</sup>. La rete preaddestrata è in grado di classificare le immagini in 1000 categorie di oggetti, come tastiera, mouse, matita e molti animali. Di conseguenza, la rete ha appreso rappresentazioni dettagliate di features per un'ampia gamma di immagini.

---

<sup>1</sup>Il progetto ImageNet è un ampio database di immagini pensato per essere utilizzato nella ricerca sui software di riconoscimento degli oggetti visuali. Più di 14 milioni di immagini sono state annotate a mano per fornire indicazioni sugli oggetti raffigurati e, in almeno un milione di immagini, sono stati forniti anche i bounding box.

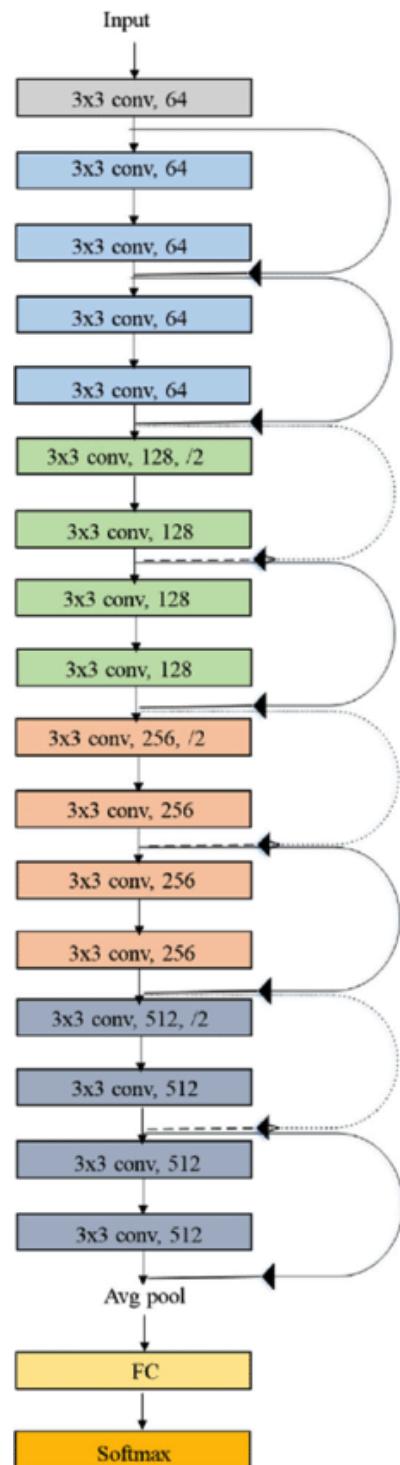


Figura 3.6: Immagine che mostra l'architettura della rete ResNet 18.

Per classificare correttamente i rifiuti, la rete ResNet è stata allenata con un dataset contenente immagini di vari rifiuti denominato Trashnet (Mindy Yang, Gary Thung, 2016 [16]). Il dataset comprende sei classi: vetro, carta, cartone, plastica, metallo e indifferenziato. Attualmente, il dataset è composto da 2527 immagini distribuiti nella seguente maniera:

- 501 vetro
- 594 carta
- 403 cartone
- 482 plastica
- 410 metallo
- 137 indifferenziato

Le immagini sono state scattate posizionando l'oggetto su uno sfondo bianco e utilizzando la luce del sole e/o dell'ambiente.

Di seguito vengono mostrate 6 immagini estratte da ciascuna categoria.



(a) Cartone



(b) Vetro



(c) Metallo



(d) Carta



(e) Plastica



(f) Indifferenziato

Figura 3.7: Immagini che mostrano un campione rappresentativo della classe di appartenenza.

Per il progetto è stata utilizzata una rete pre-addestrata<sup>2</sup> sul dataset appena citato. Sono stati fatti studi di accuracy sia sulle reti non pre-addestrate che pre-addestrate, ottenendo fino al 93.735 di accuracy sulla rete Pre-trained ResNet50:

---

<sup>2</sup><https://github.com/sangminwoo/RecycleNet>

Method	Accuracy	Parameters
ResNet18	70.302	11.18
ResNet34	64.965	21.29
ResNet50	58.701	23.52
Pre-trained ResNet18	90.023	11.18
Pre-trained ResNet34	93.271	21.29
Pre-trained ResNet50	93.735	23.52

Terminata l’analisi degli aspetti software, nel prossimo capitolo affronteremo gli aspetti hardware.

## 3.2 Aspetti Hardware

### 3.2.1 Single Board Computer - Raspberry Pi

Come spesso accade nei progetti scientifici è usuale eseguire il proprio codice in sistemi *embedded*, dove il termine si riferisce a un dispositivo le cui capacità sono limitate ai requisiti del compito per cui è stato progettato. A differenza di altre applicazioni e strumenti, tale dispositivo non può essere ulteriormente ampliato per includere altre funzionalità. Tali dispositivi vengono anche chiamati **Single Board Computer** (abbreviato in SBC), in quanto si tratta di computer completi e già funzionanti che comprendono un microprocessore, funzioni di I/O, memoria e altre caratteristiche che costituiscono un sistema informatico, costruiti su un’unica scheda elettronica. Sono inoltre dotati di una RAM integrata, spesso con una quantità predefinita, ma senza slot di espansione per le periferiche.

Un classico esempio di SBC può essere un distributore automatico. In questo caso, un SBA controlla le semplici funzioni di un distributore automatico e, a parte i compiti necessari, non è in grado di fare altro.

In particolare, il SBC utilizzato per questo progetto è una Raspberry Pi Model 4, Figura 3.8.

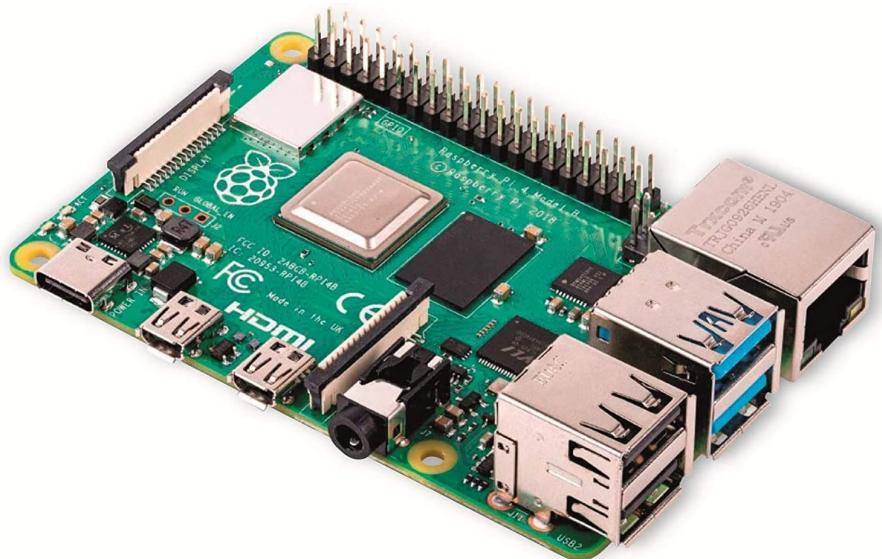


Figura 3.8: **SoC**: Broadcom BCM2711B0 quad-core A72 (ARMv8-A) 64-bit @ 1.5GHz; **GPU**: Broadcom VideoCore VI; **Networking**: 2.4 GHz and 5 GHz 802.11b/g/n/ac wireless LAN; **RAM**: 2GB LPDDR4 SDRAM; **Bluetooth**: Bluetooth 5.0, Bluetooth Low Energy (BLE); **GPIO**: 40-pin GPIO header, populated; **Memoria**: microSD; **Porte**: 2 x micro-HDMI 2.0, 3.5mm analogue audio-video jack, 2 x USB 2.0, 2 x USB 3.0, Gigabit Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI); **Dimensioni**: 88mm x 58mm x 19.5mm, 46g.

### 3.2.2 GrovePi+

Come detto nel paragrafo 3.1.6 del capitolo precedente il modello di classificazione restituisce un’etichetta che descrive la classe di appartenenza del rifiuto. Questa informazione è necessaria che sia mostrata dal dispositivo, magari mediante un display LCD.

Quindi per poter visualizzare questa informazione bisogna installare nella Raspberry un dispositivo di espansione denominato GrovePi+, Figura 3.9[A], che porta i molteplici sensori di Grove su Raspberry Pi, tra cui il Grove LCD RGB Backlight, Figura 3.9[B].

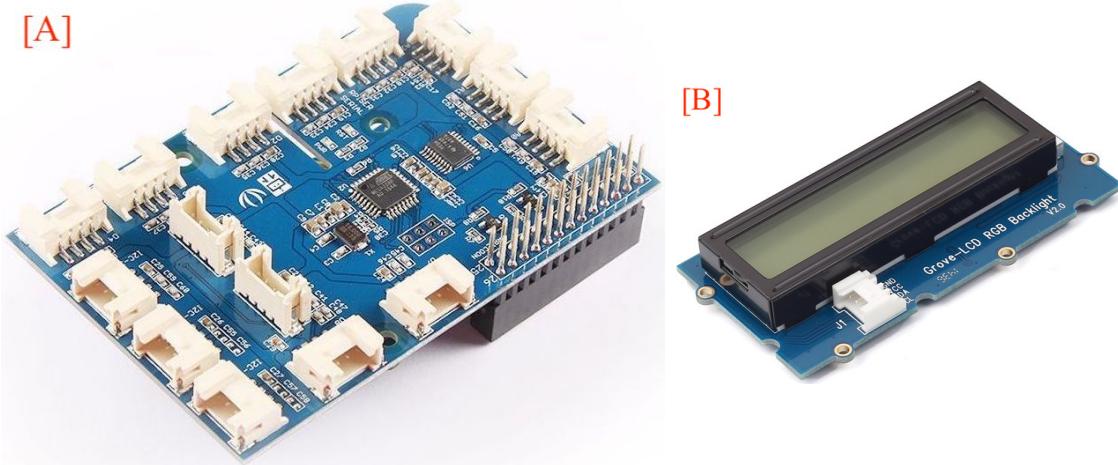


Figura 3.9: [A] Figura che mostra la scheda di espansione GrovePi+. [B] Grove LCD RGB Backlight

### Caratteristiche di GrovePi+

In questa sezione ci concentreremo sulle caratteristiche hardware del sistema GrovePi+. Sullo shield sono presenti 16 connettori Grove che possono essere suddivisi in tre diverse aree funzionali: porte digitali (8), porte analogiche (4) e porte I2C (4), Figura 3.10[A].

- **Porte digitali.** Le otto porte digitali Grove, contrassegnate dalle linee rosse in Figura 3.10[A], vengono normalmente utilizzate per la lettura di un sensore digitale che emette solo 0 o 1, quindi per l'accensione o lo spegnimento di un attuatore. Alcune di queste porte sono multifunzionali e possono funzionare come uscite PWM (modulazione di larghezza di impulso). Sono la porta 3, la porta 5 e la porta 6. Queste porte sono necessarie per il pilotaggio di un servocomando o per la regolazione della luminosità di un LED. Le porte digitali sono indispensabili anche per la comunicazione seriale.
- **Porte di ingresso analogiche.** Sul lato sinistro, in giallo, sono presenti quattro porte Grove per l'acquisizione delle letture analogiche. I sensori analogici possono fornire letture comprese tra 0 e 1024, fornendo letture più dettagliate e precise rispetto ai sensori digitali che restituiscono solo 0 o 1.
- **Porte I2C.** Sotto le porte digitali, in giallo, si trovano quattro porte I2C Grove. L'I2C è un protocollo bus a bassa velocità che trasferisce i dati attraverso due canali: SCL e SDA. SCL è la linea di clock che sincronizza il trasferimento dei dati sul bus I2C, mentre SDA è la linea dei dati. Il diagramma in Figura 3.10[B] illustra la struttura di un bus I2C.

## Sistema Proposto

---

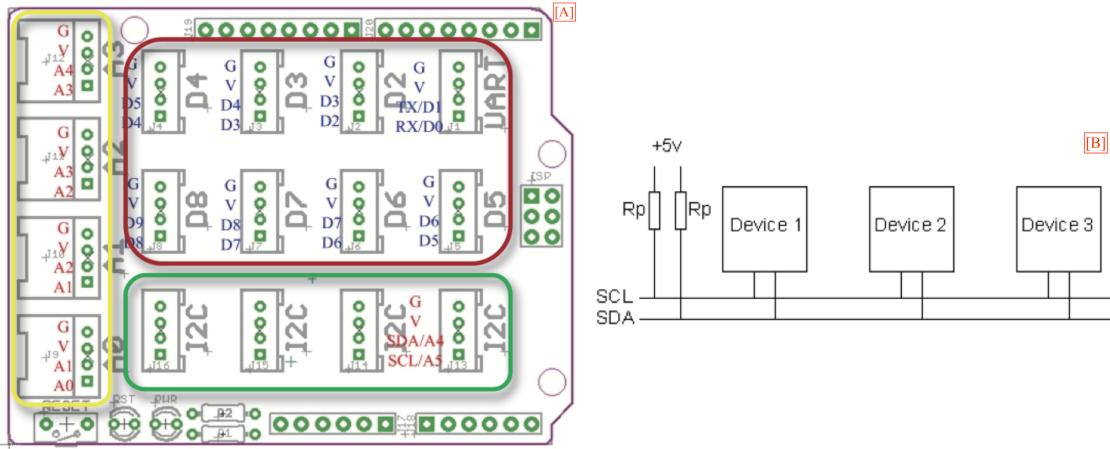


Figura 3.10: [A] Scheda GrovePi+ che evidenzia i 16 connettori disponibili: porte digitali (8), porte analogiche (4) e porte I2C (4). [B] Diagramma che illustra la struttura di un bus I2C.

Per poter utilizzare il Grove LCD RGB Backlight è sufficiente collegarlo ad una delle porte I2C. Successivamente, mediante la seguente funzione è possibile scrivere nel display:

---

```
# set display text \n for second line(or auto wrap)
def setText(text):
    textCommand(0x01) # clear display
    time.sleep(.05)
    textCommand(0x08 | 0x04) # display on, no cursor
    textCommand(0x28) # 2 lines
    time.sleep(.05)
    count = 0
    row = 0
    for c in text:
        if c == '\n' or count == 16:
            count = 0
            row += 1
            if row == 2:
                break
            textCommand(0xc0)
        if c == '\n':
            continue
        count += 1
        bus.write_byte_data(DISPLAY_TEXT_ADDR,0x40,ord(c))
```

---

Mentre, la funzione *setRGB* prende come parametri i valori rgb da (0,255) e li applica al display Grove:

---

```
# set backlight to (R,G,B) (values from 0..255 for each)
def setRGB(r,g,b):
    bus.write_byte_data(DISPLAY_RGB_ADDR,0,0)
    bus.write_byte_data(DISPLAY_RGB_ADDR,1,0)
    bus.write_byte_data(DISPLAY_RGB_ADDR,0x08,0xaa)
    bus.write_byte_data(DISPLAY_RGB_ADDR,4,r)
    bus.write_byte_data(DISPLAY_RGB_ADDR,3,g)
    bus.write_byte_data(DISPLAY_RGB_ADDR,2,b)
```

---



# Capitolo 4

## Risultati ottenuti

In questo capitolo tratteremo l'efficacia dell'algoritmo proposto. Con particolare focus sulla fase di individuazione e classificazione del rifiuto. La seguente tabella mostra i test pratici fatti su quattro categorie di rifiuti, quali: plastica, carta/cartone, vetro/metallo, indifferenziato.

	Individuazione	Classificazione
Carta/Cartone	7/3	4/6
Plastica	8/2	6/4
Vetro/Metallo	7/3	5/5
Indifferenziato	6/4	2/8

Dalla tabella si evince che:

- Su 10 diversi rifiuti proposti di tipo carta/cartone il detector individua 7 volte il rifiuto mentre 3 volte non lo riesce ad individuare correttamente; Mentre, il classificatore riesce a etichettare il rifiuto correttamente come carta/cartone solo 4 volte, mentre 6 volte lo classifica erroneamente in un'altra classe;
- Su 10 diversi rifiuti proposti di tipo plastica il detector individua 8 volte il rifiuto mentre 2 volte non lo riesce ad individuare correttamente; Mentre, il classificatore riesce a etichettare il rifiuto correttamente come plastica 6 volte, mentre 4 volte lo classifica erroneamente in un'altra classe;
- Su 10 diversi rifiuti proposti di tipo vetro/metallo il detector individua 7 volte il rifiuto mentre 3 volte non lo riesce ad individuare correttamente; Mentre, il classificatore riesce a etichettare il rifiuto correttamente come vetro/metallo 5 volte, mentre per altre 5 volte lo classifica erroneamente in un'altra classe;

- Su 10 diversi rifiuti proposti di tipo indifferenziato il detector individua 6 volte il rifiuto mentre 4 volte non lo riesce ad individuare correttamente; Mentre, il classificatore riesce a etichettare il rifiuto correttamente come indifferenziato solo 2 volte, mentre 8 volte lo classifica erroneamente in un'altra classe.

I test pratici suesposti hanno dimostrato come la fase di detection risulti essere maggiormente precisa rispetto al lavoro svolto dal classificatore. Questo dovuto probabilmente alle ridotte dimensioni del dataset di addestramento, che causa l'overfitting<sup>1</sup>.

---

<sup>1</sup>L'overfitting è un comportamento di machine learning indesiderato che si verifica quando il modello di machine learning fornisce previsioni accurate per i dati di addestramento ma non per i nuovi dati.

# Capitolo 5

## Conclusioni

La classificazione dei rifiuti in varie categorie è possibile grazie ad algoritmi di machine learning e computer vision. Uno dei maggiori punti deboli è la notevole varietà degli elementi da classificare (ad esempio, ogni oggetto può essere classificato in una delle categorie di rifiuti o di raccolta differenziata). Pertanto, per creare un sistema più accurato, è necessario disporre di una mole di dati molto ampia e in continua espansione.

Per quanto riguarda l'aspetto Hardware, ci sono alcuni aspetti da considerare sulla scelta di un Single Board Computer. Utilizzare un dispositivo di limitate dimensioni ha sicuramente il vantaggio di poterlo installare in luoghi dove computer di grandi dimensioni non potrebbero entrare, come per esempio all'interno di un bidone dei rifiuti. D'altra parte però, eseguire in un SBC algoritmi di computer vision e machine learning richiede un notevole sforzo computazionale che, nei casi di algoritmi molto sofisticati, le SBC non possono offrire. Una soluzione potrebbe essere quella di abbassare il frame-rate del flusso video. Tale scelta d'altro canto riduce la reattività generale dell'algoritmo, rischiando di abbassare le probabilità di individuazione dei rifiuti. Bisogna fare del fine-tuning sui parametri ed adattarli al particolare caso d'uso.

### 5.1 Sviluppi futuri

Innanzitutto, si potrebbe continuare a lavorare sulla CNN per perfezionarla a raggiungere una migliore precisione. Inoltre, sarebbe opportuno estendere questo progetto per identificare e classificare più oggetti da una singola immagine o video. Questo, per esempio, potrebbe aiutare maggiormente gli impianti di riciclaggio, elaborando un flusso di materiale riciclato piuttosto che singoli oggetti. Un'altra importante aggiunta potrebbe essere il rilevamento e la classificazione su più oggetti. Questo migliorerebbe la classificazione su larga scala degli oggetti da riciclare.

---

*Conclusioni*

Infine, continuare a espandere il dataset aggiungendo altre foto e possibilmente altre classi.

# Bibliografia

- [1] Mittal e Paragios, 2004 *Motion-based background subtraction using adaptive kernel density estimation.* [https://www.researchgate.net/publication/4082269\\_Motion-based\\_background\\_subtraction\\_using\\_adaptive\\_kernel\\_density\\_estimation](https://www.researchgate.net/publication/4082269_Motion-based_background_subtraction_using_adaptive_kernel_density_estimation)
- [2] Toyama, K., Krumm, J., Brumitt, B., Meyers, B., 1999 *Wallflower: principles and practice of background maintenance.* [https://www.researchgate.net/publication/3816601\\_Wallflower\\_Principles\\_and\\_practice\\_of\\_background\\_maintenance](https://www.researchgate.net/publication/3816601_Wallflower_Principles_and_practice_of_background_maintenance)
- [3] Monnet, A., Mittal, A., Paragios, N., Ramesh, V., 2003 *Background modeling and subtraction of dynamic scenes.* [https://lear.inrialpes.fr/people/triggs/events/iccv03/cdrom/iccv03/1305\\_monnet.pdf](https://lear.inrialpes.fr/people/triggs/events/iccv03/cdrom/iccv03/1305_monnet.pdf)
- [4] Stenger, B., Ramesh, V., Paragios, N., Coetzec, F., Buhmann, J.M., 2001 *Topology free hidden Markov models: application to background modeling.* [https://bjornstenger.github.io/papers/stenger\\_iccv2001.pdf](https://bjornstenger.github.io/papers/stenger_iccv2001.pdf)
- [5] Kato, J., Joga, S., Rittscher, J., Blake, A., 2002 *An HMM-based segmentation method for traffic monitoring movies.* <https://ieeexplore.ieee.org/document/1033221>
- [6] Elgammal, A., Harwood, D., Davis, LS., 2000 *Non-parametric background model for background subtraction.* <https://typeset.io/papers/non-parametric-model-for-background-subtraction-1p88su6g9w>
- [7] Cemgil, A.T., Zajdel, W., Kroese, B., 2005 *A hybrid graphical model for robust feature extraction from video.* <https://ieeexplore.ieee.org/document/1467397>
- [8] Prati, A., Mikic, I., Trivedi, M., Cucchiara, R., 2003 *Detecting moving shadows: Formulation, algorithms and evaluation.* [https://www.researchgate.net/publication/3193568\\_Detecting\\_Moving\\_Shadows\\_Algorithms\\_and\\_Evaluation](https://www.researchgate.net/publication/3193568_Detecting_Moving_Shadows_Algorithms_and_Evaluation)
- [9] Titterington, D., 1984 *Recursive parameter estimation using incomplete data.* <https://vdocuments.mx/recursive-parameter-estimation-using-incomplete-data.html?page=1>
- [10] Zivkovic, Z., van der Heijden, F., 2004 *Recursive unsupervised learning of finite mixture models.* [https://pure.uva.nl/ws/files/4070253/39501\\_170736y](https://pure.uva.nl/ws/files/4070253/39501_170736y).

- [pdf](#)
- [11] Hall, P., Hui, T.C., Marron, J.S., 1995 *Improved variable window kernel estimates of probability densities.* <https://arxiv.org/abs/1007.4350>
  - [12] Zoran Z., Ferdinand van der H., 2006 *Efficient adaptive density estimation per image pixel for the task of background subtraction.* [https://www.researchgate.net/publication/222571251\\_Efficient\\_adaptive\\_density\\_estimation\\_per\\_image\\_pixel\\_for\\_the\\_task\\_of\\_background\\_subtraction](https://www.researchgate.net/publication/222571251_Efficient_adaptive_density_estimation_per_image_pixel_for_the_task_of_background_subtraction)
  - [13] Zoran Zivkovic, 2004 *Improved adaptive gaussian mixture model for background subtraction.* [https://www.researchgate.net/publication/4090386\\_Improved\\_Adaptive\\_Gaussian\\_Mixture\\_Model\\_for\\_Background\\_Subtraction](https://www.researchgate.net/publication/4090386_Improved_Adaptive_Gaussian_Mixture_Model_for_Background_Subtraction)
  - [14] Sagar, 2020 *Laplacian and it's use in Blur Detection.* <https://medium.com/@sagardhungel/laplacian-and-its-use-in-blur-detection-fbac689f0f88>
  - [15] Gonzalez, Rafael C. Woods, Richard E., 2018 Pearson *Capitulo 9. Digital image processing.*
  - [16] Mindy Yang, Gary Thung, 2016 *Classification of Trash for Recyclability Status*