ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Scuola di Ingegneria
Corso di Laurea in Ingegneria e Scienze Informatiche

LOCALIZZAZIONE DI OGGETTI A PARTIRE DA IMMAGINI DI PROFONDITÀ

Elaborata nel corso di: Architetture Degli Elaboratori

Tesi di Laurea di:
DAVIDE CONTI

Relatore:
Prof. DAVIDE MALTONI

Co-relatore:
Dott. LORENZO PELLEGRINI

ANNO ACCADEMICO 2018–2019
SESSIONE III

PAROLE CHIAVE

Localizzazione basata sulla profondità
Localizzazione di oggetti
Machine Learning
Visione artificiale
RealSense

Alla mia famiglia

Indice

Introduzione

L'uomo è in grado di riconoscere e localizzare una moltitudine di oggetti in pochi istanti e con uno sforzo contenuto. Lo scopo della visione artificiale è quello di replicare questa intelligenza in modo da poterla portare in un sistema artificiale.
Il rilevamento di oggetti nella visione artificiale è la capacità di trovare un determinato oggetto in una sequenza di immagini o video. Solitamente per ottenere risultati significativi si utilizzano algoritmi che sfruttano il machine learning o il deep learning. Tuttavia queste tecniche di machine learning hanno i seguenti problemi:

- Richiedono molto tempo per l'addestramento.
- Sono troppo pesanti per piattaforme robotiche/embedded/mobile per poter permettere una localizzazione fluida (con un numero di frame per secondo accettabile).

L'obiettivo di questo elaborato di tesi è la progettazione e lo sviluppo di un sistema in grado di localizzare uno specifico oggetto in una sequenza di immagini o video in tempi soddisfacenti, in modo tale da poterlo utilizzare anche in tempo reale. Tale sistema è proposto come evoluzione e sostituzione di quello attualmente utilizzato in CORe50 [7], dove viene utilizzata una finestra fissa in cui un utente umano deve necessariamente posizionarvi l'oggetto.
In questa tesi si approfondiranno i motivi che rendono la localizzazione di oggetti importante nella computer science, gli ambiti dove viene utilizzata, le metodologie che si possono adottare, i dispositivi che vengono impiegati per la raccolta delle immagini o dei video ed alcuni datasets utilizzati. Infine si esplorerà la soluzione adottata.

Capitolo 1

Localizzazione di oggetti

In questo capitolo vengono analizzati i problemi della visione artificiale e in quali ambiti essa viene utilizzata.

1.1 Che cos'è la localizzazione di oggetti?

La visione artificiale è un campo scientifico interdisciplinare che, dal punto di vista ingegneristico, cerca di automatizzare i compiti che il sistema visivo umano può svolgere includendo metodi per acquisire, processare, analizzare e capire immagini digitali.
I problemi che la visione artificiale cerca di risolvere sono di quattro tipi: riconoscimento, localizzazione, rilevazione e segmentazione di immagini.
Nel riconoscimento di immagini l'obiettivo è quello di assegnare un'etichetta a un'immagine. Ad esempio l'immagine di un cane riceverà l'etichetta "cane". Un immagine che contiene due cani riceverà ancora l'etichetta "cane".
La localizzazione di oggetti, invece, ha il compito di trovare l'area di uno o più oggetti che appartengono ad una specifica classe conosciuta e disegnarci, per esempio, un riquadro attorno. Ritornando all'esempio precedente la localizzazione avrebbe disegnato un riquadro attorno alla figura del cane oppure avrebbe disegnato un riquadro attorno a ciascun cane. La rilevazione di oggetti combina la localizzazione con il riconoscimento. In pratica assegna un'etichetta a ciascuna area trovata. Infine nella segmentazione si etichettano i singoli pixel dell'immagine (figura 1.1).
In questo elaborato userò il termine *localizzazione* per indicare l'individuazione dell'area di un singolo oggetto, con il rispettivo riquadro disegnato, e

rilevazione per indicare l'etichettatura dell'area trovata.

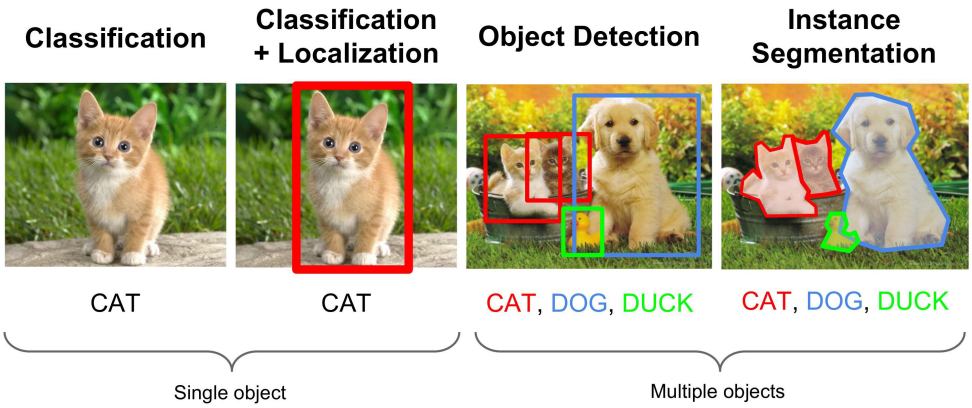


Figura 1.1: Esempio dei risultati attesi per i più comuni problemi di visione artificiale. Da sinistra verso destra: (a) riconoscimento di un singolo oggetto appartenente ad una specifica classe; (b) rilevazione di un singolo oggetto appartenente ad una specifica classe; (c) rilevazione di più oggetti appartenenti a classi differenti; (d) segmentazione di più oggetti appartenenti a classi differenti.

La localizzazione di oggetti è un problema fondamentale e stimolante nel campo della visione artificiale ed è stata un'area di ricerca attiva per molti decenni. I recenti sviluppi hanno cambiato radicalmente questo settore. Molti scenari che prima erano considerati di difficile risoluzione, ora non solo sono possibili ma addirittura semplici da risolvere. Nel 2014 la capacità di eseguire una classificazione in un'immagine era considerata limitata. Solo tre anni dopo, non solo siamo in grado di determinare se un'immagine contiene uno specifica classe, ma siamo addirittura in grado di determinarne la posizione. Questa evoluzione è dovuta all'introduzione delle reti neurali, in particolare all'uso di CNN [8], R-CNN [6], Fast R-CNN [5], Faster R-CNN [12], YOLO [11] e SSD [14], in contrasto con le precedenti tecniche "classiche" che in molti casi non portavano a risultati soddisfacenti.

1.2 Dove e perché viene utilizzata?

Si sta utilizzando la localizzazione di oggetti in una vasta gamma di settori, con casi d'uso che vanno dalla sicurezza personale alla produttività sul posto di lavoro. La localizzazione e il riconoscimento degli oggetti vengono applicate in molte aree della visione artificiale, incluso il recupero delle immagini, la sicurezza, la sorveglianza, i sistemi di veicoli automatizzati e l'ispezione delle macchine.

1.2.1 Riconoscimento ottico dei caratteri

Il riconoscimento ottico dei caratteri consente l'estrazione di caratteri da documenti, immagini o video. Alcune applicazioni di utilizzo sono le seguenti:

- convertire documenti cartacei contenti testo in documenti equivalenti in formato digitale.
- estrarre i caratteri presenti in un'immagine, come ad esempio la targa di un'automobile, il testo sui cartelli e cartelloni pubblicitari (come si vede in figura 1.2).

Figura 1.2: Riconoscimento dei caratteri contenuti nella targa dell'automobile.

1.2.2 Guida autonoma

Un'auto per essere in grado di decidere quale azione intraprendere (accelerare, frenare o girare) deve sapere dove si trovano tutti gli oggetti intorno a sé e la loro classe di appartenenza (automobili, pedoni, semafori, segnali stradali, biciclette, motociclette, ecc..) (figura 1.3).

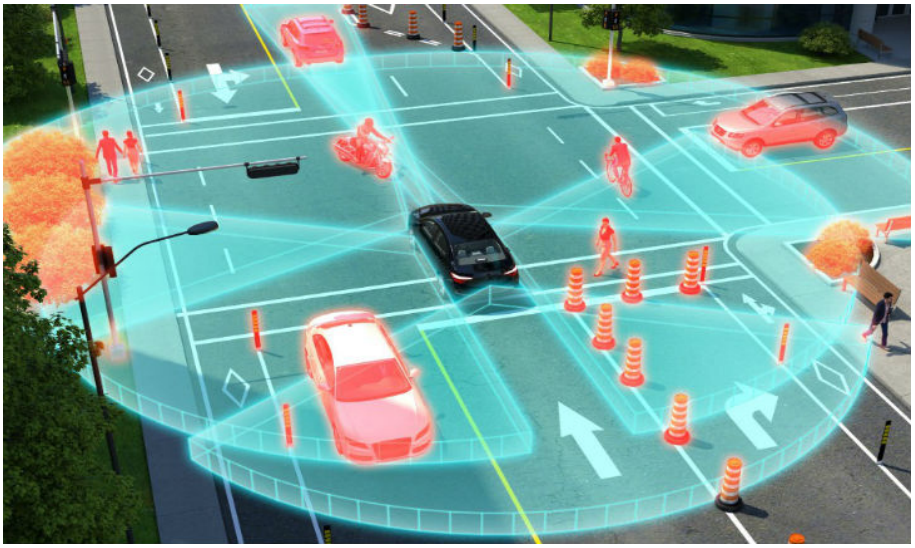


Figura 1.3: Un auto che riesce ad individuare gli oggetti intorno a sé.

1.2.3 Tracciamento di oggetti

Si usa la localizzazione degli oggetti anche per tracciare gli oggetti, ad esempio per tracciare una palla durante una partita di calcio o tracciare una persona in un video.

1.2.4 Rilevamento e riconoscimento del volto

Il rilevamento e il riconoscimento del volto sono ampiamente utilizzati nelle attività di visione artificiale tra cui il riconoscimento facciale per sbloccare i telefoni. Vengono usati anche in banche, aeroporti e stadi per verificare l'identità degli utenti e per individuare criminali.

1.2.5 Rilevamento dei pedoni

Il rilevamento dei pedoni è un compito fondamentale e significativo in qualsiasi sistema di videosorveglianza intelligente, in quanto fornisce le informazioni fondamentali per la comprensione semantica dei filmati video.

1.2.6 Immagini mediche

Gli strumenti di elaborazione delle immagini mediche svolgono un ruolo sempre più importante nell'assistere i medici nella diagnosi, nella pianificazione della terapia e negli interventi guidati dall'immagine. Il tracciamento accurato, robusto e veloce di oggetti anatomici deformabili come il cuore è un compito cruciale nell'analisi delle immagini mediche.

1.2.7 Industria manifatturiera

Dalla rivoluzione industriale, l'umanità ha fatto enormi progressi nella produzione manifatturiera. Con il passare del tempo abbiamo assistito a un numero sempre maggiore di lavori manuali sostituiti dall'automazione attraverso ingegneria avanzata, computer, robotica e ora IoT. L'intelligenza artificiale, più precisamente l'apprendimento profondo, aiuterà ad accelerare questa tendenza verso l'automazione. La rilevazione degli oggetti viene utilizzata nei processi industriali per effettuare:

- gestione della qualità: l'intelligenza artificiale è in grado di distinguere automaticamente e rapidamente i prodotti difettosi e di concedere il tempo necessario per intraprendere azioni correttive.

- gestione dell'inventario: l'intelligenza artificiale consente di contare l'inventario in modo accurato ed efficiente, riducendo il rischio di errore umano.

- ordinamento: utilizzando il rilevamento di oggetti basato sull'intelligenza artificiale, è possibile selezionare parametri specifici e visualizzare le statistiche corrispondenti al numero di oggetti. In questo modo il numero di anomalie durante la categorizzazione è ridotto e la catena di montaggio è più flessibile.

- catena di montaggio: l'uso dell'intelligenza artificiale permette di localizzare e differenziare correttamente i prodotti in correlazione con il loro movimento effettuando un lavoro più efficiente.

1.2.8 Robotica

L'evoluzione della scienza robotica ha portato i robot moderni a raggiungere autonomia e mobilità. L'autonomia è la capacità del robot di eseguire i propri compiti senza l'intervento umano. La mobilità è la capacità di un robot di potersi muovere liberamente nel mondo.
I robot devono essere dotati della capacità di elaborare i dati visivi in tempo reale in modo che possano reagire adeguatamente per adattarsi rapidamente ai cambiamenti dell'ambiente. La localizzazione e il riconoscimento affidabile degli oggetti è il primo passo necessario per raggiungere questo obiettivo.
Alcuni dei campi della robotica sono i seguenti:

- biomedica: robot capaci di assistere il chirurgo durante le operazioni, come ad esempio i robot telecontrollati con tecnologie dette di telepresenza che permettono al chirurgo di operare a distanza.

- industriale: robot che sostituiscono l'uomo in operazioni ripetitive: il loro impiego nelle catene di montaggio ha permesso alle aziende di abbattere notevolmente i costi accelerando e migliorando la produzione. Fra i robot più utilizzati dall'industria vi è il braccio robotico, o robot manipolatore, costruito a imitazione del braccio umano, ma spesso dotato di più gradi di libertà.

- militare: robot utilizzati allo scopo di non mettere a repentaglio le vite umane. Un esempio sono i robot utilizzati per la ricognizione e la vigilanza, come ad esempio i droni. Un altro esempio sono i robot artificieri che sono in grado di compiere analisi su un ordigno esplosivo ed eventualmente neutralizzarlo a distanza riducendo drasticamente i rischi degli artificieri.

- sociale: robot capaci di interagire e comunicare con gli esseri umani, o con altri agenti fisici, in modo autonomo.

- umanoide: robot con sembianze umane e con un'intelligenza che cerca di imitare l'uomo.

- spaziale: robot utilizzati fuori dall'atmosfera terrestre, come ad esempio le sonde esplorative impiegate in diverse missioni sui pianeti del sistema solare.
- arte: robot utilizzati per creare nuove forme di espressione artistica e per imitare e riprodurre le forme artistiche già esistenti, come ad esempio i robot progettati per dipingere o per suonare uno strumento musicale leggendo in tempo reale uno spartito.

1.3 Motivazione del progetto

La localizzazione a finestra fissa è la tecnica più semplice che si può utilizzare per la localizzazione di oggetti. L'obiettivo di questo progetto è quello di rendere fattibile, grazie all'uso delle informazioni di profondità, l'identificazione in maniera automatica della posizione del/degli oggetti in una sequenza di immagini o video in tempi soddisfacenti, in modo da poterlo utilizzare anche in tempo reale, e che allo stesso tempo risulti meno oneroso di un approccio basato sulle reti neurali, in modo da poter essere applicato in ambito robotico/mobile/embedded.

1.3.1 Localizzazione a finestra fissa

La localizzazione a finestra fissa non è una vera e propria tecnica di localizzazione poiché non utilizza nessun algoritmo. Questa tecnica si basa sull'utilizzo di una finestra statica fissa, in cui l'operatore deve necessariamente posizionare l'oggetto (vedi figura 1.4).
Il pregio di questa tecnica è la sua semplicità: non è richiesto alcun algoritmo per identificare l'oggetto. I suoi difetti, invece, sono la mancanza di generalità e la sua limitata applicabilità in tanti contesti reali.

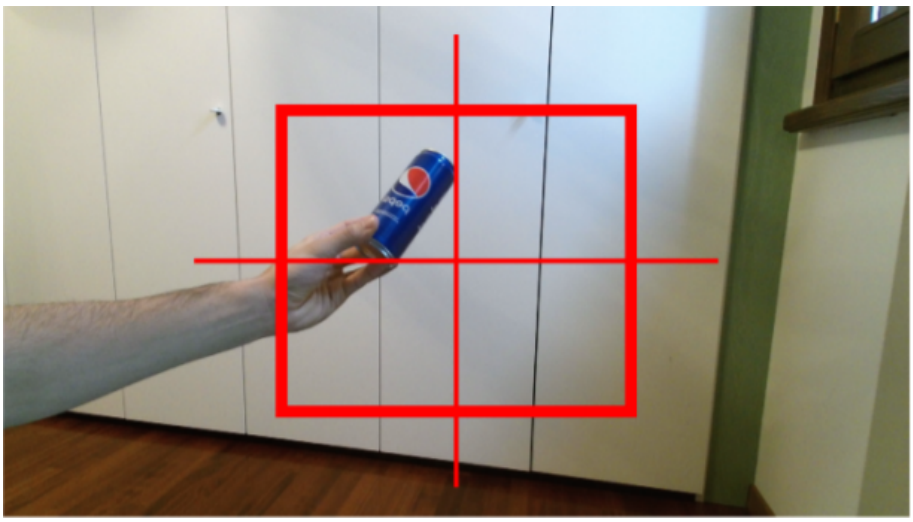


Figura 1.4: Localizzazione a finestra fissa in cui all'operatore è richiesto di posizionare l'oggetto all'interno della regione centrale.

Capitolo 2

Tecniche di localizzazione

In questo capitolo vengono approfondite le tecniche che si utilizzano per cercare di trovare la posizione degli oggetti all'interno di un'immagine.

2.1 Metodologie

Le metodologie che si possono adottare sono di quattro tipi: basata su feature, basata sul movimento, basata sull'uso della profondità e basata su reti neurali.

2.1.1 Localizzazione basata su feature

Le immagini vengono memorizzate con una feature per canale per pixel. Trattare le immagini con questo grande numero di feature non è impossibile, infatti molte tecniche classiche e tutte quelle basate su reti neurali usano tutte le feature, però non è efficiente per i seguenti motivi:

1. gli algoritmi richiedono molto tempo per essere completati.
2. le features conterranno un sacco di informazioni inutili, come ad esempio lo sfondo, la luce, la sfocatura e i cambi di rotazione.

La localizzazione basata su feature utilizza metodi che creano un vettore di lunghezza n, chiamato *feature vector*, contenente solo le informazioni rilevanti, estratte da un'immagine di dimensione $Larghezza * Altezza * Canali$. Per le comuni immagini RGB il numero di canali è pari a tre.

2.1.1.1 Histogram of Oriented Gradients (HOG)

Questa tecnica è stata introdotta da Navneet Dalal e Bill Triggs nel 2005 [2] ed ha il vantaggio di essere poco onerosa dal punto di vista computazionale e utile per molti problemi del mondo reale.
Questo algoritmo lavora con immagini di dimensione arbitraria e la dimensione dell'output dipende dall'immagine di input, dal numero di canali dell'istogramma e dalle orientazioni. Sostanzialmente, data un'immagine e fissati i parametri di HOG, il vettore restituito è di dimensione fissata.
In breve, il suo funzionamento consiste (vedi la figura 2.2 e la figura 2.3) nel calcolare i valori dei gradienti per le direzioni orizzontali e verticali attraverso i kernel mostrati in figura 2.1 (Dalal e Triggs hanno testato anche altri tipi di kernel più complessi, come ad esempio il filtro 3x3 di Sobel, ma hanno constatato che generamente restituiscono risultati peggiori nel rilevamento di esseri umani). Effettuando questa operazione si attiveranno

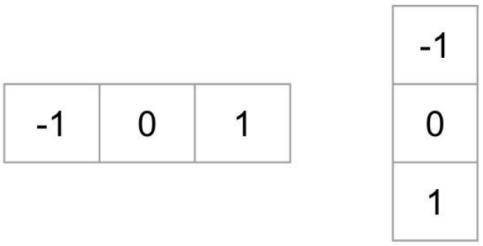


Figura 2.1: Kernel utilizzati per il calcolo dei valori dei gradienti per le direzioni orizzonali e verticali.

tutti i pixels che presentano un brusco cambiamento di intensità. In questo modo si evidenzieranno i contorni degli oggetti principali e si rimuoveranno una buona parte di informazioni inutili, come lo sfondo.
Successivamente l'immagine ottenuta viene divisa in piccole celle e per ogni cella viene calcolato l'istogramma dei gradienti. Le celle possono essere rettangoli o avere una forma radiale e i canali dell'istogramma sono distribuiti uniformemente da 0 a 180 gradi o da 0 a 360 gradi, a seconda che il gradiente sia con o senza segno. Dalal e Triggs nei loro esperimenti hanno scoperto

che i gradienti senza segno usati insieme ad istogrammi con 9 canali (corrispondenti agli angoli 0, 20, 40, 60, 80, 100, 120, 140 e 160) restituiscono risultati migliori per la rilevazione di umani.
I gradienti di un'immagine sono sensibili alle variazioni degli effetti di luce, per cui è necessario normalizzarli localmente. Questa normalizzazione richiede di raggruppare celle in blocchi più grandi, spazialmente collegati. Infine ciascun blocco ottenuto viene dato in input ad un algoritmo di machine learning per cercare di identificare se al suo interno è presente o meno uno specifico oggetto.
Una delle aree in cui questa tecnica ha avuto maggior successo è il rilevamento dei pedoni perché una persona può variare notevolemente nel colore, nell'abbigliamento e in altri fattori, ma in generale i bordi di un pedone rimangono relativamente costanti, specialmente attorno all'area delle gambe. In genere può essere utilizzato per rilevare anche altri tipi di oggetti, ma il suo successo può variare a seconda dell'applicazione specifica.

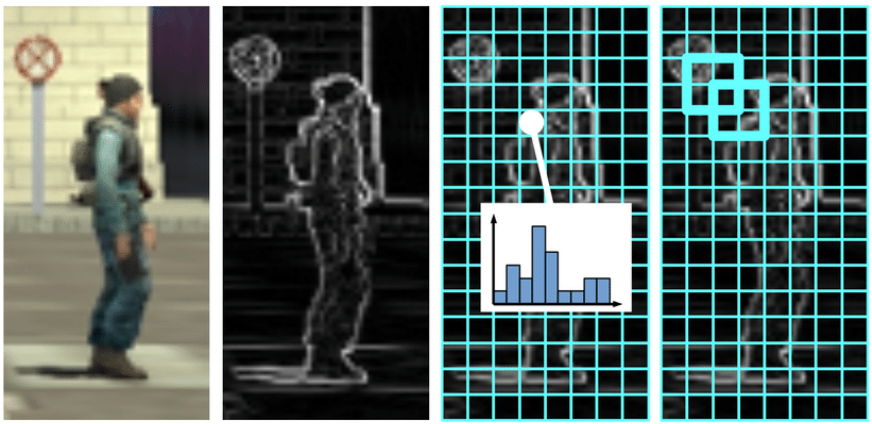


Figura 2.2: Algoritmo HOG. Da sinistra verso destra: (a) immagine di input; (b) immagine ottenuta dal calcolo dei gradienti; (c) l'immagine mostra il partizionamento in celle, in cui ogni cella contiene l'istogramma dei gradienti; (d) l'immagine mostra il raggruppamento di celle in blocchi normalizzati 2x2.

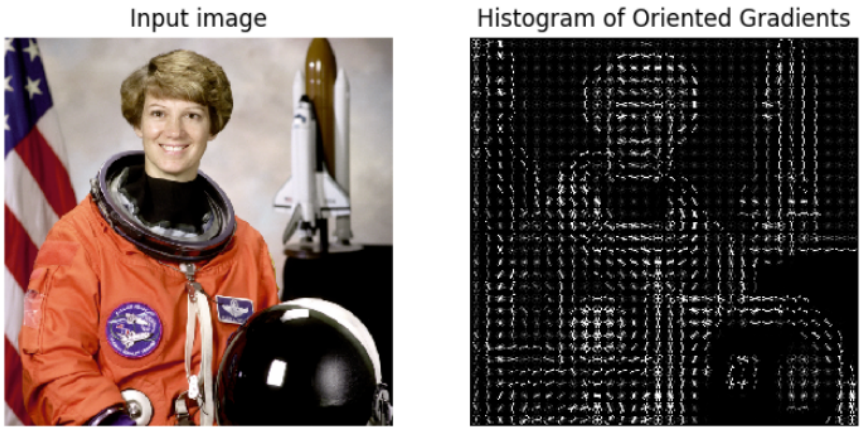


Figura 2.3: Esempio di output restituito dall'algoritmo HOG. Da sinistra verso destra: (a) immagine di input; (b) immagine di output.

2.1.2 Localizzazione basata sul movimento

La localizzazione basata sul movimento consiste nel confrontare più fotogrammi consecutivi provenienti da un video per determinare, con vari metodi, se viene rilevato un oggetto in movimento.
Questa tecnica è stata utilizzata per una vasta gamma di applicazioni come il monitoraggio delle condizioni stradali, la videosorveglianza, l'analisi delle attività umane, la sicurezza aeroportuale, ecc...

2.1.2.1 Frame difference

Frame difference è una tecnica in cui si vanno ad analizzare le differenze fra due fotogrammi provenienti da un video. Se i pixels sono stati cambiati significa che c'è stato qualche cambiamento e che qualcosa è cambiato nell'immagine (per esempio il movimento) (figura 2.4). Per distinguere i movimenti reali dal rumore la maggior parte delle tecniche lavorano con la sfocatura e un valore di soglia. Questo perché i fotogrammi potrebbero differire anche quando le condizioni di luce cambiano (ad esempio una camera che ha il focus automatico, correzioni di luce, ecc...).

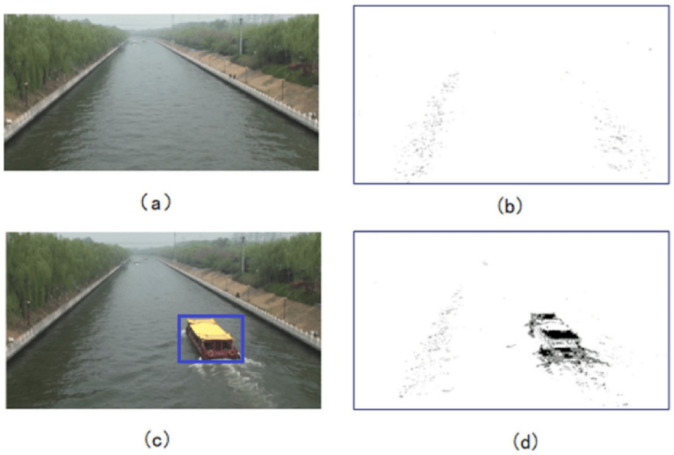


Figura 2.4: Esempio dei risultati ottenuti utilizzando la tecnica frame difference: (a) immagine originale senza nessun target; (b) differenza dei fotogrammi senza nessun target; (c) immagine originale contenente il target; (d) differenza dei fotogrammi evidenziando il target in movimento.

2.1.2.2 Temporal difference

La logica di questa tecnica è identica a quella descritta precedentemente (frame difference), ad essere diversa è la modalità di esecuzione della differenza. Nella temporal differance la differenza è eseguita con duo o tre fotogrammi successivi, mentre nella frame difference viene attuata con il fotogramma immediatamente successivo.

2.1.2.3 Background subtraction

Background subtraction è una tecnica che consente di estrarre il primo piano di un'immagine per un'ulteriore elaborazione (riconoscimento degli oggetti, ecc...). È un approccio ampiamente utilizzato per rilevare oggetti in movimento nei video, generati da telecamere fisse. La logica di questo metodo consiste nel rilevare gli oggetti in movimento dalla differenza tra il fotogramma corrente e un fotogramma di riferimento, spesso chiamato "immagine di sfondo".
Background subtraction viene utilizzato per numerose applicazioni della vi-

sione artificiale, come ad esempio il monitoraggio della sorveglianza o la stima delle pose umane.

2.1.3 Localizzazione basata sull'uso della profondità

La localizzazione basata sull'uso della profondità fa uso di telecamere, come ad esempio la Kinect e RealSense, in grado di restituire, oltre all'immagine RGB, un'immagine che contiene la distanza di ogni pixel (vedi figura 2.5).

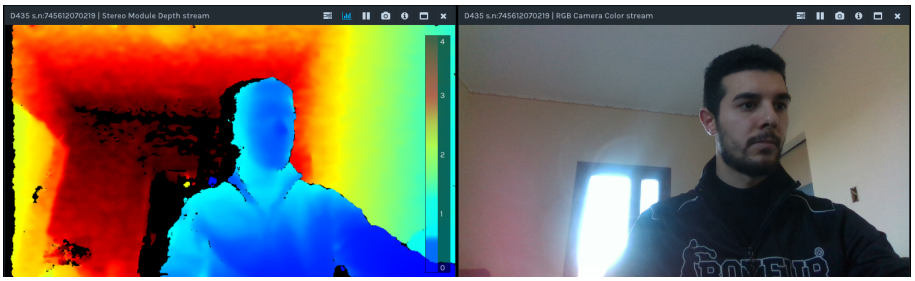


Figura 2.5: Esempio di immagine di profondità e di immagine RGB generata dalla camera RealSense D435. Da sinistra verso destra: (a) immagine di profondità; (b) immagine RGB.

La profondità può essere sfruttata per superare le principali difficoltà nell'individuazione della posizione degli oggetti, come ad esempio la variazione di luce, le forme, l'occlusione e per colmare il divario tra il 2D e il 3D. Inoltre può essere un valido indizio per la preelaborazione delle immagini, riducendo lo spazio di ricerca per la localizzazione. A partire da queste immagini, di profondità e RGB, si utilizzano diversi algoritmi, come ad esempio l'etichettatura delle componenti connesse [3], per cercare di individuare la posizione di specifici oggetti.

2.1.4 Localizzazione basata su reti neurali

Nel settore della visione artificiale una delle innovazioni più promettenti negli ultimi anni sono state le reti neurali convoluzionali (CNN). Questi tipi di reti vengono utilizzate esclusivamente per compiti di classificazione e regressione e non possono risolvere il problema della localizzazione degli oggetti in quanto restituiscono un output di dimensione fissa (il numero di

oggetti all'interno di un'immagine non è mai fisso). Un possibile approccio per poter utilizzare queste reti per risolvere un problema di rilevamento potrebbe essere quello di prendere differenti regioni di interesse dall'immagine, e usare la rete convoluzionale per classificare la presenza di un oggetto all'interno di ogni regione. Il problema con questo approccio è che gli oggetti di interesse potrebbero avere posizioni spaziali diverse all'interno dell'immagine e diverse proporzioni. Questo potrebbe portare a dover classificare una grande quantità di regioni portando a un carico computazionale ingestibile.

2.1.4.1 R-CNN (Region-based Convolutional Neural Networks)

Per risolvere questo problema Ross Girshick et al hanno limitato il numero di regioni a 2000 attraverso un algoritmo chiamato *Selective Search*. In seguito queste regioni ottenute vengono deformate in un quadrato e date in input ad una rete neurale convoluzionale che produce in output un feature vector di 4096 dimensioni. Infine questo feature vector viene passato in input ad un classificatore SVM (Support Vector Machine) per classificare la presenza di un determinato oggetto all'interno della regione proposta (figura 2.6). Oltre a prevedere la presenza di un oggetto all'interno della regione, l'algoritmo prevede anche quattro valori di offset che si riferiscono all'area predetta. Tali valori permettono di aumentare la precisione del rettangolo (regolando il rettangolo in base alla posizione e dimensione dell'oggetto).
Questo algoritmo presenta i seguenti problemi:

- l'addestramento richiede molto tempo poiché bisogna classificare 2000 regioni per immagini.
- non può essere usato in tempo reale poiché ogni immagine richiede un esecuzione della durata di molti secondi.
- l'algoritmo selective search è un algoritmo fisso, pertando in quella fase non c'è nessun apprendimento. Questo potrebbe condurre alla generazione di regioni inadeguate.

2.1.4.2 Fast R-CNN

Lo stesso autore del precedente paper (R-CNN) ha creato un nuovo algoritmo, *Fast R-CNN*, di rilevamento degli oggetti più veloce che ha risolto

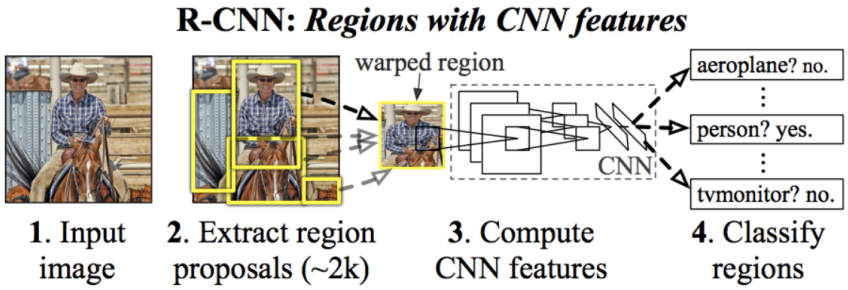


Figura 2.6: Algoritmo R-CNN. Da sinistra verso destra: (1) immagine di input; (2) estrazione delle 2000 regioni dall'immagine di input; (3) ogni regione viene data in input alla CNN; (4) l'output ottenuto dalla CNN viene dato in input al classificatore.

alcuni problemi di R-CNN.
L'approccio è simile a quello dell'algoritmo R-CNN, ma invece di passare in input, alla rete neurale convoluzionale, la regione proposta, si passa l'intera immagine e come output si ottiene una mappa di caratteristiche convoluzionali. Da questa mappa, si identificano le regioni proposte, si deformano in un quadrato e per mezzo di uno strato di *Rol pooling* si rimodellano in una dimensione fissa in modo che possano essere inserite in uno strato completamente connesso. Dal *Rol feature vector* si usa il livello *softmax* per prevedere la classe della regione proposta e anche i valori di offset per il rettangolo di delimitazione.
Fast R-CNN è più veloce di R-CNN perché non richiede, ogni volta, di passare in input le 2000 regioni proposte alla rete neurale convoluzionale, ma solamente una volta per immagine.

2.1.4.3 Faster R-CNN

Entrambi gli algoritmi precedenti, R-CNN e Fast R-CNN, usano l'algoritmo selective search per trovare le regioni proposte. Questo algoritmo è dispendioso in termini di tempo e quindi influisce sulle prestazioni della rete. Shaoqing Ren et al. hanno inventato un algoritmo, *Faster R-CNN*, di rilevamento degli oggetti che elimina l'algoritmo di selective search e consente alla rete di apprendere le regioni proposte.

Ugualmente a Fast R-CNN, l'intera immagine è fornita in input alla rete neurale convoluzionale, la quale fornisce una mappa di caratteristiche convoluzionali. Ora per predire le regioni, invece di utilizzare l'algoritmo selective search, si passa in input la mappa ottenuta ad un'altra rete separata. Le regioni ottenute saranno rimodellate attraverso il livello Rol polling e utilizzate per classificare l'immagine e fornire i valori di offset dei riquadri di delimitazione.

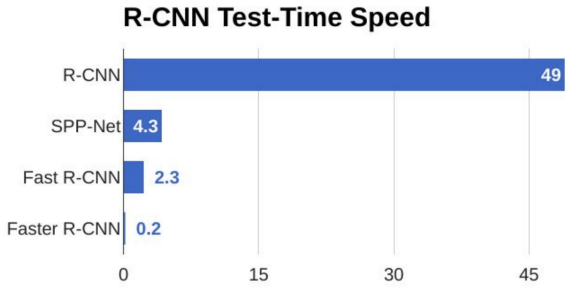


Figura 2.7: Comparazione dei tempi di esecuzione richiesti da R-CNN, SPP-Net, Fast R-CNN e Faster R-CNN per il processamento di una singola immagine.

La figura 2.7 mostra i tempi richiesti per processare una singola immagine dai vari algoritmi descritti precedentemente. Da questa figura possiamo notare che Faster R-CNN è talmente più veloce dei suoi predecessori da poter essere utilizzato anche in tempo reale.

2.1.4.4 YOLO (You Only Look Once)

Tutti gli algoritmi precedenti fanno uso di regioni per cercare di localizzare gli oggetti all'interno dell'immagine. YOLO, invece, utilizza una singola rete neurale convoluzionale che predice i riquadri di delimitazione e la classe di appartenenza di ogni riquadro con la rispettiva probabilità. Il funzionamento di YOLO consiste nel dividere l'immagine in una griglia SxS e, per ogni griglia, predire N riquadri di delimitazione, ognuno con la rispettiva classe di appartenenza e con la rispettiva probabilità. Infine selezionerà solo i riquadri che hanno una probabilità superiore ad un certo valore soglia (vedi figura 2.8).

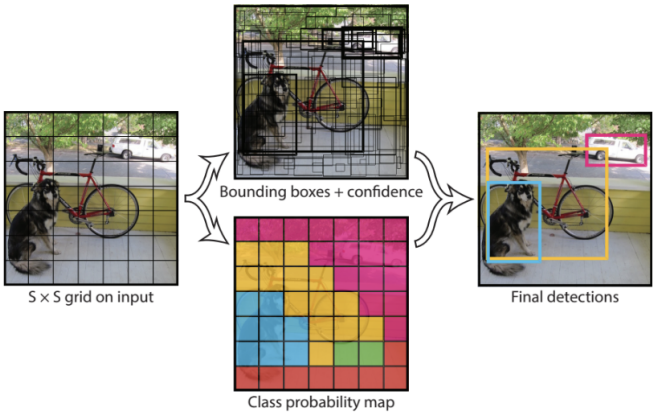


Figura 2.8: Algoritmo YOLO. Da sinistra verso destra: (a) l'immagine viene divisa in una griglia SxS; (b) per ogni griglia vengono ottenuti N riquadri di delimitazione, ognuno con la rispettiva classe di appartenenza e con la rispettiva probabilità; (c) immagine di output in cui vengono visualizzati solo i riquadri che hanno una probabilità superiore ad un certo valore soglia.

Questo algoritmo risulta molto più veloce delle tecniche esposte in precedenza (raggiunge 45 fotogrammi per secondo) ed è utilizzato per una vasta gamma di problemi come ad esempio il rilevamento dei veicoli in strada, il monitoraggio del traffico, il rilevamento degli esseri umani, etc... Tuttavia riscontra difficoltà nel rilevare piccoli oggetti come ad esempio il rilevamento di uno stormo di uccelli. Ciò è dovuto ai vincoli spaziali dell'algoritmo.

2.1.4.5 SSD (Single Shot Detector)

SSD è l'ideale per avere un buon compromesso fra velocità e accuratezza. Il suo funzionamento è il seguente: passa in input l'intera immagine ad una rete neurale convoluzionale, la quale restituisce la mappa di caratteristiche. Successivamente predice i riquadri di delimitazione, con la rispettiva classe di appartenenza e con la rispettiva probabilità, passando la mappa ottenuta ad un livello convoluzionale 3x3.

2.1.4.6 Comparazione tra Fast R-CNN, Faster R-CNN, YOLO e SSD in relazione all'accuratezza e alla velocità di esecuzione

Quale fra questi algoritmi basati sulle reti neurali è il migliore? La risposta dipende da quello che si vuole ottenere: Faster R-CNN è la scelta migliore per quanto riguarda l'accuratezza, mentre YOLO è la scelta migliore per quanto riguarda la velocità di esecuzione. Un compromesso fra velocità e accuratezza è sicuramente SSD.
Riguardo all'accuratezza, bisogna tenere conto anche della dimensione degli oggetti che si vogliono localizzare. Con grandi oggetti sembra che SSD ottiene un'accuratezza simile a quella di Faster R-CNN, ma con piccoli oggetti il divario che separa questi due algoritmi è più grande.

Capitolo 3

Panoramica dei dispositivi

In questo capitolo viene mostrata una panoramica di alcuni dispositivi utilizzati per la generazione di immagini di profondità.

3.1 Depth

Le camere con cui, di solito, siamo abituati a fare foto e video generano immagini formate da migliaia di pixels. Ogni pixel possiede tre attributi: il valore del colore rosso, il valore del colore verde ed il valore del colore blu (RGB). Una camera di profondità, invece, genera immagini in cui ogni pixel ha un solo attributo: il valore della distanza dalla camera (depth). Alcune camere di profondità restituiscono immagini in cui ogni pixel ha tutti e quattro gli attributi (RGB-D).
Questi tipi di camere offrono il vantaggio di aggiungere informazioni utili all'immagine: rendono di più facile comprensione distinguere gli oggetti in primo piano dallo sfondo. Questo diventa molto utile nella segmentazione dello sfondo, in cui è possibile rimuovere oggetti di sfondo da un'immagine. Le camere di profondità sono molto utili nel campo della robotica e di dispositivi autonomi come i droni: se un drone, o un robot, si muove attorno ad uno spazio, la profondità è utile per sapere quali oggetti sono davanti a lui in modo da poterli evitare.
Per la visualizzazione dell'immagine di profondità è possibile scegliere quale mappa utilizzare. Non importa quale mappa si utilizza poiché essa serve solo a visualizzare in modo semplice quali pixel sono lontani dalla camera e

quali sono vicini. Per esempio la figura 3.1 mostra che i pixel di colore ciano sono più vicini alla camera, mentre quelli di colore rosso sono più lontani.

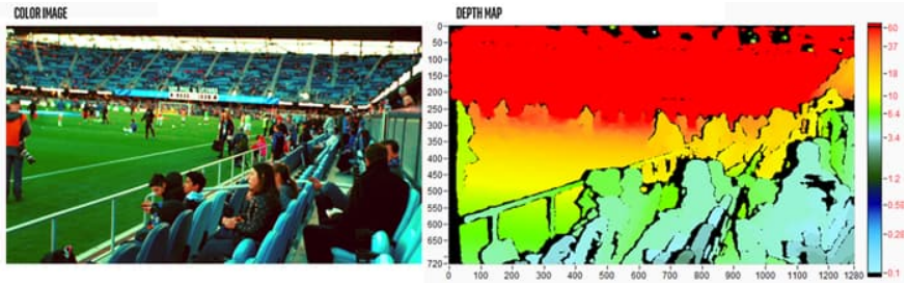


Figura 3.1: Esempio di un'immagine catturata da una camera di profondità. Da sinistra verso destra: (a) immagine RGB; (b) immagine di profondità con applicata una specifica mappa (i pixel di colore rosso sono più lontani dalla camera, mentre quelli di colore ciano sono più vicini).

3.2 Depth e infrarossi

Esistono differenti metodi per calcolare la distanza di ogni singolo pixel dell'immagine dalla camera. Il metodo più appropriato da utilizzare dipende dalle caratteristiche del problema: la distanza massima che si vuole raggiungere, l'accuratezza che si vuole ottenere e se si vuole operare in ambienti all'aperto.
Alcuni metodi che utilizzano la luce ad infrarossi, e che verranno approfonditi, per calcolare la distanza sono i seguenti: luce strutturata e luce codificata, profondità stereo e tempo di volo.
La luce ad infrarossi aiuta a percepire profondità dettagliate anche in condizioni di scarsa luminosità. Ad esempio nel riconoscimento facciale e nei sistemi di visione artificiale la luce ad infrarossi è molto utile quando la luce illumina in maniera troppo forte, troppo debole o "a macchie" gli oggetti.

3.2.1 Luce strutturata e luce codificata

La luce strutturata e la luce codificata sono tecnologie simili ma non identiche. Si basano sulla proiezione di luce (di solito luce ad infrarossi) di un

certo tipo di emettitore. Questa luce proiettata è modellata o visivamente o durante il tempo o con una combinazione delle due. Poiché il modello proiettato è conosciuto, il sensore della camera è in grado di fornire le informazioni della distanza attraverso il pattern osservato nella scena. Ad esempio, se il modello è una serie di strisce proiettate su una palla, tali strisce si deformerebbero e piegherebbero intorno alla superficie della palla in modo specifico. Se la palla si avvicinasse all'emettitore, anche il pattern cambierebbe (vedi figura 3.2). Per ogni pixel è possibile calcolare la

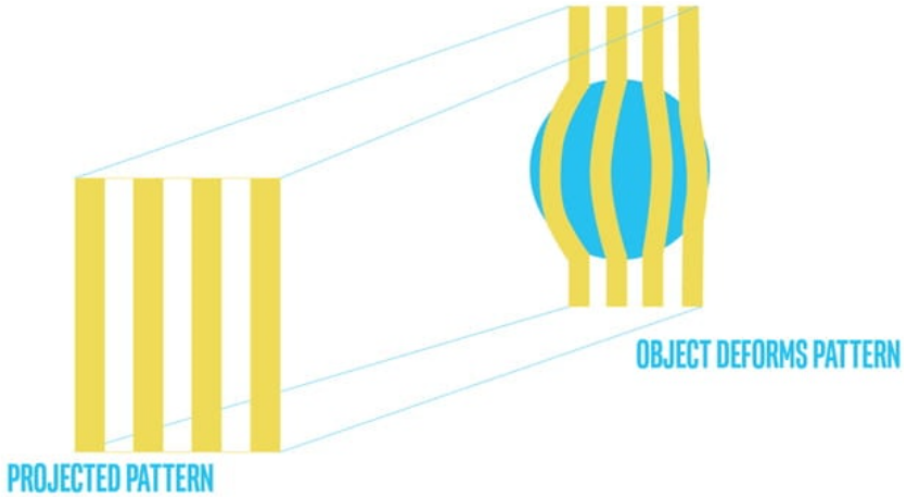


Figura 3.2: Tecnica luce strutturata e luce codificata. Da sinistra verso destra: (a) modello proiettato dalla camera; (b) oggetto che deforma il modello.

distanza dalla camera utilizzando la disparità tra l'immagine visualizzata e un'immagine prevista.
Le camere che utilizzano questo metodo funzionano meglio in ambienti chiusi e, a seconda della potenza della luce emessa, a distanze relativamente brevi poiché questa tecnologia si basa sulla visione accurata di un modello proiettato di luce. Inoltre questa tecnica è vulnerabile ai rumori dell'ambiente prodotti da altre camere o dispositivi che emettono luce ad infrarossi. Gli ambiti ideali di utilizzo di queste camere sono il riconoscimento dei gesti e la segmentazione dello sfondo.

3.2.2 Profondità stereo

La profondità stereo, a differenza delle due tecniche precedentemente descritte, può utilizzare qualsiasi luce per misurare la profondità. Il suo funzionamento si basa su come come noi usiamo i nostri due occhi per percepire la distanza. Siccome il nostro cervello misura la distanza fra i nostri due occhi, gli oggetti più vicini a noi sembrano muoversi più velocemente da un occhio all'altro, mentre invece gli oggetti in lontananza sembrano muoversi più lentamente. In modo analogo questa tecnica cattura due immagini, da due diversi sensori, e le confronta. Dato che la distanza fra questi due sensori è conosciuta, il confronto fra queste due immagini produce l'informazione della distanza.
La profondità stereo funziona bene anche in condizioni di luce variabili inclusi gli ambienti esterni, poiché utilizza qualsiasi caratteristica visiva per misurare la profondità. Il vantaggio di questa tecnica è che non interferisce con altre camere, come invece avviene con camere che utilizzano metodi di luce strutturata o luce codificata.
La distanza che queste camere possono misurare è direttamente correlata con la distanza dei due sensori. Più è grande questa distanza e più la camera sarà precisa sulla lunga distanza.

3.2.3 Tempo di volo

Il tempo di volo utilizza la velocità della luce, che è conosciuta, per calcolare la profondità. Il suo funzionamento consiste nell'emettere luce ad infrarossi e calcolare il tempo impiegato da tale luce per ritornare indietro al sensore. Questi sensori sono in grado di misurare, in relazione alla potenza e alla lunghezza d'onda della luce, profondità a distanze significative.
Gli svantaggi di questo metodo sono i seguenti:

- la luce che colpisce il sensore potrebbe non essere stata emessa dalla specifica camera ma potrebbe provenire da qualche altra sorgente, come ad esempio il sole o altre camere.

- in ambienti all'aperto non produce risultati soddisfacenti.

3.2.4 RealSense

Intel RealSense (figura 3.3) è una camera leggera, potente e a basso costo che utilizza la profondità stereo per calcolare la distanza di ogni singolo pixel dell'immagine, e che genera una profondità di qualità adatta per una vasta gamma di applicazioni. Il suo ampio campo visivo e la sua portata, fino a dieci metri, sono perfetti per applicazioni come la robotica, il riconoscimento di oggetti o la realtà aumentata e virtuale, dove vedere quanta più scena possibile è di vitale importanza.

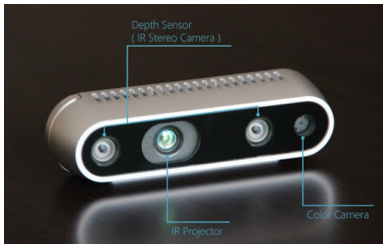


Figura 3.3: Camera Intel RealSense D435.

3.2.5 Kinect

Kinect è un sensore di movimento ideato per rendere le console Xbox360 e XboxOne ancora più coinvolgenti ed appassionanti, offrendo agli utenti l'opportunità di divenire essi stessi i controller del gioco, sostituendo i joypad. Questo sensore è in grado di registrare i movimenti del corpo nelle 3 differenti dimensioni ed elaborarli in tempo reale, quindi a seconda di come noi ci muoveremo nello spazio il sensore trasformerà i dati ricevuti in movimenti del nostro avatar nel gioco. Dato che il sensore effettua questo compito attraverso immagini di profondità viene utilizzato anche nel campo della localizzazione degli oggetti.
Attualmente esistono due versioni di Kinect: Kinect v1 e Kinect v2.

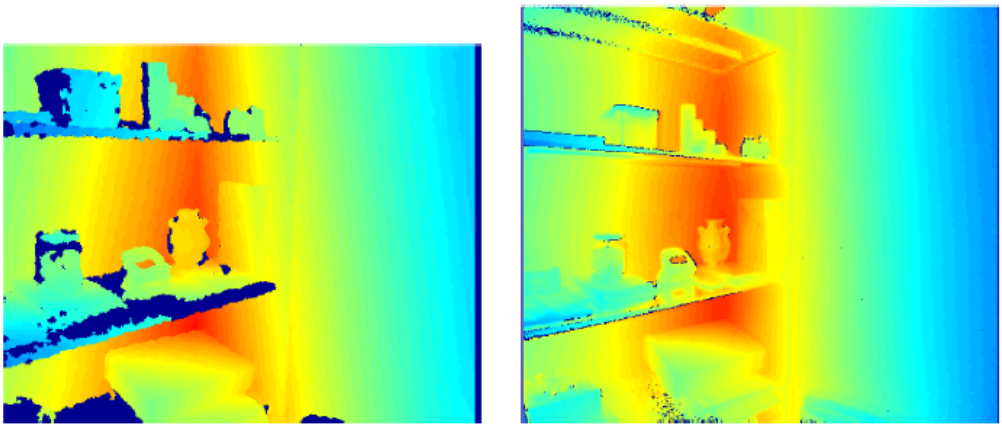
3.2.6 Comparazione Kinect/RealSense

La tabella 3.1, la figura 3.4 e la figura 3.5 mostrano le principali differenze fra Kinect v1, Kinect v2 e RealSense D435 e si può notare che:

- Kinect v1 utilizza il metodo della luce strutturata per il calcolo della profondità: ci potrebbero essere problemi se si utilizzano più Kinect v1, poiché interferiscono tra loro.
- Kinect v2 utilizza il metodo del tempo di volo per il calcolo della profondità: è più stabile, preciso e meno soggetto ad interferenze rispetto al metodo utilizzato da Kinect v1.
- RealSense D435 utilizza il metodo della profondità stereo per il calcolo della profondità: è sicuramente migliore rispetto ai metodi utilizzati da Kinect V1 e Kinect V2, in quando non ci saranno problemi di interferenze con altre camere.
- Kinect v2 ha prestazioni incredibilmente migliori rispetto a Kinect v1: la risoluzione RGB raggiunge il full-HD e il campo visivo è stato notevolmente migliorato.
- RealSense D435 raggiunge prestazioni migliori rispetto Kinect v2: la risoluzione depth ed il campo visivo sono migliori e la distanza massima è stata più che raddoppiata.
- RealSense D435 è molto più compatta rispetto a Kinect v1 e Kinect v2.

Differenze Kinect v1, Kinect v2 e Realsense D435			
	Kinect v1	**Kinect v2**	**RealSense D435**
Metodo per calcolare la profondità	Luce strutturata	Tempo di volo	**Profondità stereo**
Sviluppata da	Microsoft e PrimeSense	Microsoft	Intel
Risoluzione RGB	640x480 a 30fps	**1920x1080 a 30fps**	**1920x1080 a 30 fps**
Risoluzione Depth	640x480 a 30fps	512x424 a 30fps	**fino a 1280x720 a 90fps**
Minima distanza di profondità	40 cm	50 cm	**10cm**
Massima distanza di profondità	circa 4.5 m	circa 4.5 m	**10m**
Campo visivo orizzontale di profondità	57 gradi	70 gradi	**87 gradi**
Campo visivo verticale di profondità	43 gradi	**60 gradi**	58 gradi
Dimensione fisica (lunghezza x profondità x altezza)	38,1 cm x 11,934 cm x 12,954 cm	24.9 cm x 6.6 cm x 6.7 cm	**9 cm x 2,5 cm x 2,5 cm**
USB Standard	2.0	3.0	**3.0 Type-C**

Tabella 3.1: Principali differenze fra Kinect v1, Kinect v2 e RealSense D435.



(a) Kinect v1 (b) Kinect v2

Figura 3.4: Esempio di comparazione delle immagini di profondità generate da Kinect v1 e Kinect v2. Da sinistra verso destra: (a) immagine di profondità generata da Kinect v1; (b) immagine di profondità generata da Kinect v2.

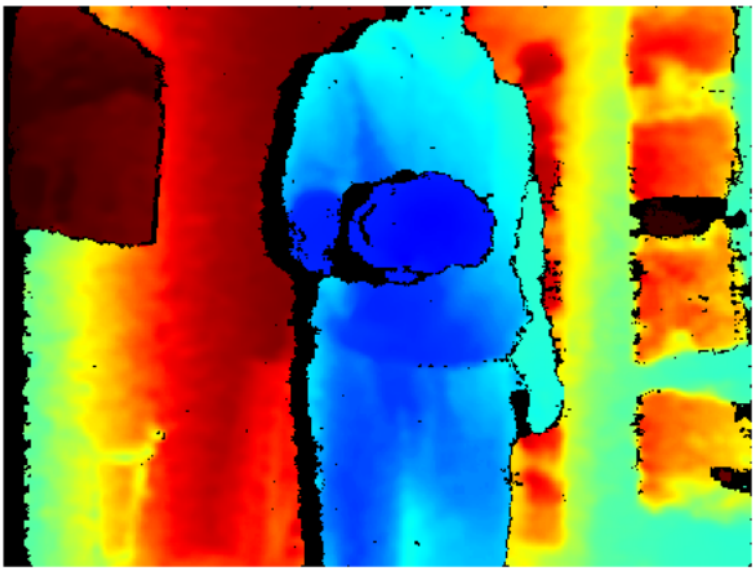


Figura 3.5: Esempio di immagine di profondità generata dalla camera RealSense D435.

Capitolo 4

Progetto

In questo capitolo viene descritta la soluzione adottata per implementare un sistema che individui la posizione di un singolo oggetto in una sequenza di immagini o video in tempi soddisfacenti, in modo da poterlo utilizzare anche in tempo reale. Essendo l'obiettivo quello di applicare la soluzione proposta agli ambiti robotico/mobile/embedded, l'algoritmo è stato sviluppato prendendo in considerazione tecniche di visione classiche al fine di essere molto meno oneroso di un approccio basato su reti neurali.
Nella prima parte del capitolo verranno anche analizzati alcuni datasets.

4.1 Datasets utilizzati in ambito Robotic Vision

In questa sezione verranno discussi alcuni datasets che vengono utilizzati in ambito robotic vision per la localizzazione degli oggetti. I datasets che verranno discussi sono: CoRE50, iCubWorld e OpenLORIS [13].

4.1.1 CORe50

CORe50 è un dataset e benchmark specificamente progettato per il riconoscimento continuo degli oggetti.
É una collezione di 50 oggetti appartenenti a 10 categorie (vedi figura 4.1): adattatori, telefoni cellulari, forbici, lampadine, lattine, bicchieri, palline, pennarelli, tazze e telecomandi.

Figura 4.1: Immagine di esempio dei 50 oggetti in CORe50. Ogni colonna indica una delle 10 categorie.

Il dataset consiste di 164.866 128 x 128 RGB-D immagini e di 11 sessioni distinte (8 interne e 3 esterne) con sfondi e illuminazioni diverse. Per ogni sessione e per ogni oggetto è stato registrato un video della durata di 15 secondi (a 20 fps) con un sensore Kinect v2 che fornisce 300 fotogrammi RGB-D. Gli oggetti sono tenuti in mano dall'operatore e il punto di vista della telecamera è quello degli occhi dell'operatore. All'operatore è stato richiesto di allungare il suo braccio e spostare/ruotare l'oggetto davanti alla telecamera. Un punto di vista soggettivo con oggetti afferrati e alla stessa distanza è adatto per un gran numero di applicazioni robotiche. La mano (sinistra o destra) che afferra l'oggetto cambia durante le sessioni e le occlusioni sono spesso prodotte dalla stessa mano.
Tre delle undici sessioni (la terza, la settima e la decima sessione) sono state selezionate per costituire il test set, mentre le rimanenti otto sessioni vengono utilizzate per l'addestramento. Gli autori hanno provato a bilanciare il più possibile la difficoltà delle sessioni tra addestramento e test rispetto a: interno/esterno, stretta della mano (sinistra o destra) e la complessità dello sfondo, come visibile nell'esempio proposto in figura 4.2.

Figura 4.2: Immagine di esempio di uno stesso oggetto durante le 11 sessioni di acquisizione. Da notare la variabilità in termini di sfondo, illuminazione, sfocatura, occlusione, posa e scala.

4.1.2 iCubWorld

iCubWorld datasets sono una collezione di immagini che registrano l'esperienza visiva di iCub mentre osserva gli oggetti nel suo ambiente tipico, un laboratorio o un ufficio. L'impostazione dell'acquisizione è concepita per consentire un interazione uomo-robot naturale, dove un insegnante mostra l'oggetto di interesse e verbalmente gli assegna l'etichetta. In questi datasets gli oggetti sono tenuti in mano ad una distanza quasi costante dalla fotocamera e vengono mossi casualmente.
Esistono quattro tipi di datasets: Hello iCubWorld [4], iCubWorld 1.0 [1], iCubWorld28 [9] e iCubWorld Transformations [10].

4.1.2.1 Hello iCubWorld

Il dataset Hello iCubWorld comprende 7 oggetti e 4 acquisizioni (circa 2000 immagini) per oggetto. È la prima versione di iCubWorld, dove viene validata la configurazione dell'acquisizione ed eseguito il benchamrking dei metodi di riconoscimento. Ogni oggetto è stato acquisito da un umano e tramite un robot (vedi figura 4.3) e la procedura di acquisizione è stata ripetuta due volte per ottenere train e test set separati. Per ogni acquisizione sono state registrate 500 immagini di dimensione 320 x 240 da una telecamera iCub. Ogni immagine è stata successivamente ritagliata in un riquadro, attorno all'oggetto, di dimensione 80 x 80 in modalità umana e 160 x 160 in modalità robot.

Figura 4.3: Immagine di esempio dal dataset Hello iCubWorld che mostra oggetti acquisiti da un umano e tramite un robot.

4.1.2.2 iCubWorld 1.0

Il dataset iCubWorld 1.0 comprende 40 oggetti, divisi in 10 categorie, e 1 sola acquisizione (circa 200 immagini) per oggetto.
Per l'addestramento sono stati raccolti tre oggetti per categoria. Ogni oggetto è stato acquisito da un umano (vedi figura 4.4), registrando 200 immagini di dimensione 640 x 480 da una telecamera iCub. Ogni immagine è stata successivamente ritagliata in un riquadro, attorno all'oggetto, di dimensione 160 x 160.
I quattro insiemi di test disponibili sono:

- dimostratore: una istanza nota per categoria è stata acquisita in modalità umana con un dimostratore diverso rispetto all'addestramento.
- categorizzazione: una nuova istanza per categoria è stata acquisita con la stessa impostazione dell'addestramento.
- robot: una istanza nuova o conosciuta per categoria è stata acquisita in modalità robotica (il riquadro in questo caso ha dimensione di 320 x 320).
- sfondo: sono state selezionate 10 immagini per categoria e fornite le maschere di segmentazione degli oggetti per verificare se il classificatore riconosce l'oggetto o lo sfondo.

Figura 4.4: Immagine di esempio dal dataset iCubWorld 1.0 che mostra le istanze degli oggetti appartenenti alle dieci categorie.

4.1.2.3 iCubWorld28

Il dataset iCubWorld28 comprende 28 oggetti, divisi in 7 categorie, e 8 acquisizioni (circa 1600 immagini) per oggetto. Ogni oggetto è stato acquisito in quattro giorni diversi, al fine di studiare approcci di apprendimento incrementale, in modalità umana (vedi figura 4.5) e la procedura di acquisizione è stata ripetuta due volte per ottenere train e test separati. Per ogni acquisizione sono state registrate circa 150 immagini di dimensioni 320 x 240 da una telecamera iCub. Ogni immagine è stata successivamente ritagliata in un riquadro, attorno all'oggetto, di dimensione 128 x 128.

4.1.2.4 iCubWorld Transformations

Il dataset iCubWorld Transformations comprende 200 oggetti, divisi in 20 categorie, e 10 acquisizioni (circa 3600 immagini) per oggetto. Ogni oggetto è stato acquisito mentre subisce cinque isolate trasformazioni visive (vedi figura 4.6), al fine di studiare l'impatto delle variazioni di posa tipiche del mondo reale. Le cinque isolate trasformazioni visive sono:

- rotazione 2D: l'operatore ruota l'oggetto di fronte al robot, parallelamente al piano della telecamera, mantenendolo alla stessa distanza e posizione.

Figura 4.5: Immagine di esempio dal dataset iCubWorld28 che mostra le quattro istanze degli oggetti appartenenti alle sette categorie.

- rotazione 3D: come la rotazione 2D. La differenza è che l'operatore ruota l'oggetto fuori al piano della telecamera e cambia la faccia visibile al robot.
- scala: l'operatore muove la mano tenendo l'oggetto avanti e indietro, quindi cambiando la scala dell'oggetto rispetto alle telecamere.
- cambio dello sfondo: l'operatore si muove in un semicerchio attorno all'iCub, mantenendo approssimativamente la stessa distanza e posa dell'oggetto nella mano rispetto alle telecamere. Lo sfondo cambia radicalmente, mentre l'aspetto dell'oggetto rimane lo stesso.
- mix: l'operatore muove l'oggetto in modo naturale di fronte al robot, come farebbe una persona quando mostra un nuovo oggetto a un bambino. In questa sequenza possono apparire tutte le combinazioni.

Per ogni acquisizione sono state registrate immagini di dimensione 640 x 480. Ogni immagine è stata successivamente ritagliata in un riquadro, attorno all'oggetto, di dimensione 256 x 256.

4.1.3 OpenLORIS

OpenLORIS è un dataset e benchmark specificamente progettato per il riconoscimento continuo degli oggetti. Questo dataset è stato raccolto at-

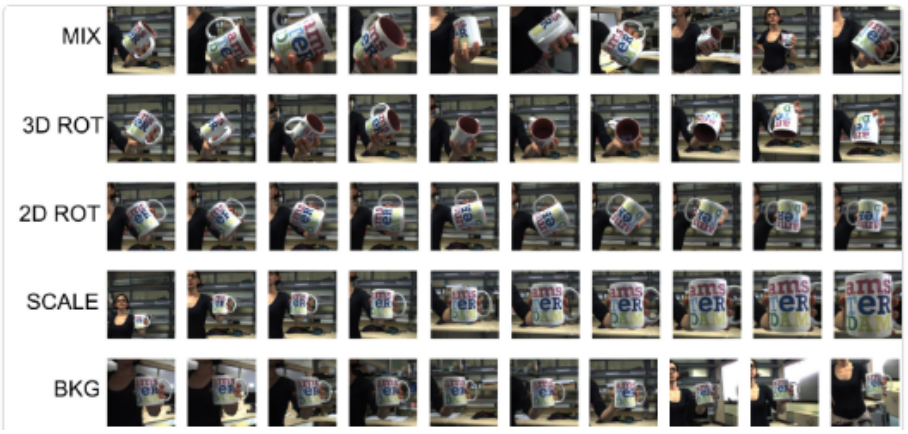


Figura 4.6: Immagine di esempio dal dataset iCubWorld Transformations che mostra le cinque isolate trasformazioni visive: mix, rotazione 3D, rotazione 2D, scala e cambio di sfondo.

traverso robot reali che montano telecamere di profondità. I dati sono stati raccolti in ambienti dinamici con diverse illuminazioni, occlusioni, dimensioni dell'oggetto, distanza/angolo dalla telecamera e disordine. Più nello specifico questo dataset include sfide comuni che i robot di solito affrontano, come l'illuminazione, l'occlusione, la distanza dell'oggetto dalla telecamera, ecc... Inoltre, tutti questi fattori sono stati decomposti in livelli di difficoltà. Ogni livello considera quattro fattori ambientali (vedi tabella 4.1): la variazione di illuminazione durante la registrazione, la percentuale di occlusione dell'oggetto, la grandezza in pixel dell'oggetto in ogni fotogramma e il disordine della scena. In sintesi, per capire meglio quali sono le caratteristiche dei dati robotici che influenzano negativamente i risultati del riconoscimento continuo degli oggetti, vengono considerati in modo indipendente l'illuminazione, l'occlusione, la grandezza dell'oggetto, la distanza dalla camera, l'angolo della camera dall'oggetto e il disordine. La grandezza dell'oggetto e la sua distanza dalla telecamera sono combinati insieme perché in scenari del mondo reale è difficile distinguere gli effetti di questi due fattori.
Il dataset OpenLORIS è una collezione di 69 oggetti, appartenenti a 19 categorie, su 7 scene. Per ogni oggetto è stato registrato un video, con una camera di profondità che genera 500 fotogrammi RGB-D, dalla durata di 17 secondi (a 30 fps). Per ogni oggetto a ciascun livello vengono forniti

260 esempi, con immagini RGB e di profondità. Quindi il numero totale di immagini fornite è circa 2 (RGB e depth) x 260 (esempi per oggetto) x 69 (oggetti) x 4 (fattori per livello) x 3 (livelli di difficoltà) = 430.560 immagini. Inoltre, per ogni immagine sono forniti anche i riquadri e le maschere di segmentazione.

	Livello 1	**Livello 2**	**Livello 3**
Illuminazione	Forte	Normale	Debole
Occlusione (percentuale)	0%	25%	50%
Grandezza oggetto	maggiore di 200 x 200 pixel	30 x 30 - 200 x 200 pixel	minore di 30 x 30 pixel
Disordine	Semplice	Normale	Complesso
Contesto	casa/ufficio/centro commerciale		
Numero categorie	19		
Numero oggetti	69		

Tabella 4.1: Differenza dei tre livelli di difficoltà definiti per ciascun fattore.

4.2 Dispositivo utilizzato

La soluzione adottata utilizza il metodo della localizzazione basata sull'uso della profondità. A tal fine, per catturare le immagini di profondità e RGB, si è scelta la camera Intel RealSense D435 in quanto il suo ampio campo visivo, la sua risoluzione depth e la sua portata, fino a dieci metri, sono perfetti per la localizzazione degli oggetti.

4.3 Etichettatura delle componenti connesse

Una componente connessa è un sottoinsieme dell'insieme dei pixel di foreground (in genere i pixel con valore 255 o 0) o di background (in genere i pixel con valore 0 o 255) tale che presi due qualsiasi dei suoi pixel, esiste

tra questi un percorso appartenente all'insieme dei pixel di foreground o di background. A seconda della metrica adottata si parla di 4-connessione (d4) o di 8-connessione (d8). La differenza tra queste due metriche è la seguente (vedi figura 4.7):

- con la metrica d4 i vicini di un pixel p possono trovarsi solamente orizzontalmente e verticalmente.
- con la metrica d8 i vicini di un pixel p possono trovarsi anche diagonalmente.

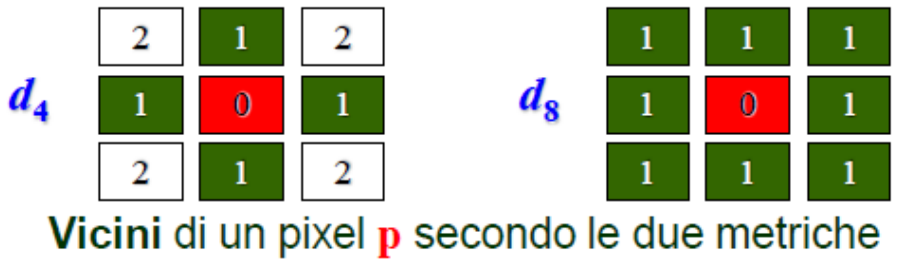


Figura 4.7: Immagine di esempio dei vicini di un pixel p secondo le due metriche: d4 e d8. Da sinistra verso destra: (a) vicini di un pixel p secondo la metrica d4; (b) vicini di un pixel p secondo la metrica d8.

L'algoritmo individua automaticamente le diverse componenti connesse in un'immagine, assegnando loro etichette (tipicamente numeriche).
Inizialmente esegue una binarizzazione dell'immagine, in modo da ottenere i pixel di foreground e i pixel di background.
In seguito scandisce l'immagine e, per ogni pixel di foreground p, a seconda della metrica utilizzata (d4 o d8) considera i pixel di foreground vicini già visitati:

- se nessuno è etichettato, assegna una nuova etichetta a p.
- se uno è etichettato, assegna al pixel p la stessa etichetta.
- se più di uno è etichettato, assegna a p una delle etichette e si annotano le equivalenze.

Infine definisce un'unica etichetta per ogni insieme di etichette marcate come equivalenti e effettua una seconda scansione assegnando le etichette finali (vedi figura 4.8).

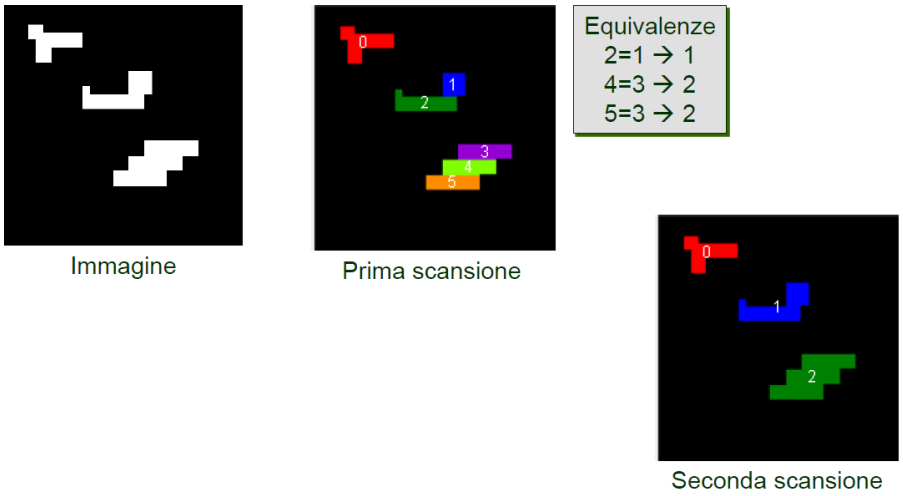


Figura 4.8: Esempio di etichettatura delle componenti connesse. Da sinistra verso destra: (a) immagine di input binarizzata; (b) prima scansione dell'immagine: si assegnano le etichette ai pixel e si annotano le equivalenze; (c) seconda scansione dell'immagine: si definisce un'unica etichetta per ogni insieme di etichette marcate come equivalenti e si assegnano le etichette finali.

4.4 Descrizione dell'algoritmo

La figura 4.9 mostra le due immagini di input che vengono utilizzate per descrivere il procedimento dell'algoritmo. La prima fase consiste nel trovare una porzione dell'immagine in modo tale da ridurre lo spazio di ricerca dell'oggetto. A questo scopo viene analizzata l'intera immagine e vengono inseriti in una lista i primi N pixel con la distanza più piccola. RealSense restituisce anche valori di distanza pari a 0 (in nero nella figura 4.9): questi vengono scartati in quanto sono pixel in cui la loro distanza non è stata definita. Dato che l'oggetto che si vuole localizzare quasi certamente sarà

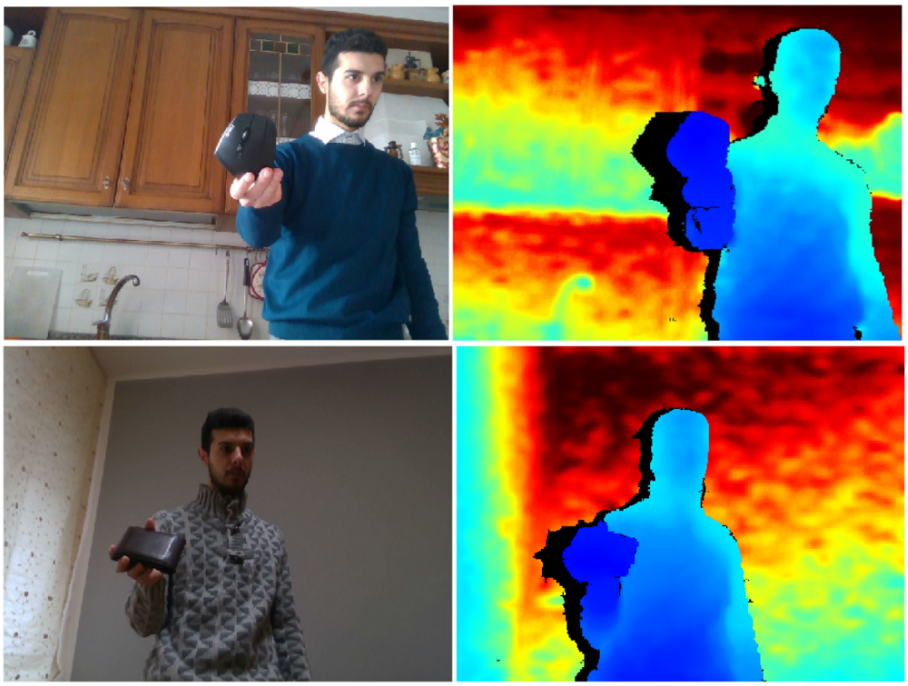


Figura 4.9: Immagini di input utilizzate per descrivere il procedimento dell'algoritmo. Da sinistra verso destra: (a) prima immagine RGB; (b) prima immagine di profondità; (c) seconda immagine RGB; (d) seconda immagine di profondità.

posizionato in avanti rispetto a tutti gli altri oggetti molto probabilmente alcuni dei suoi pixel saranno contenuti all'interno della lista. Il riquadro cercato viene centrato, il più possibile, nel baricentro di questi N pixel contenuti nella lista (vedi figura 4.10).
Come si può vedere nella figura 4.10 si ottiene un riquadro che non è perfettamente centrato nell'oggetto, infatti questo è solo un riquadro approssimativo che serve per limitare lo spazio di ricerca dell'oggetto nella fase successiva. La seconda fase consiste nel cercare il riquadro definitivo attorno all'oggetto. Inizialmente il riquadro ottenuto dalla prima fase viene ritagliato, dall'intera immagine, e trasformato in un'immagine binaria (vedi la figura 4.11), ovvero viene assegnato il valore 255 a tutti i pixel che hanno un valore sopra ad una certa soglia e 0 a tutti gli altri. Successivamente, su

Figura 4.10: Le due immagini mostrano i riquadri ottenuti aventi come centro, il più possibile, il baricentro degli N pixel con la distanza più piccola.

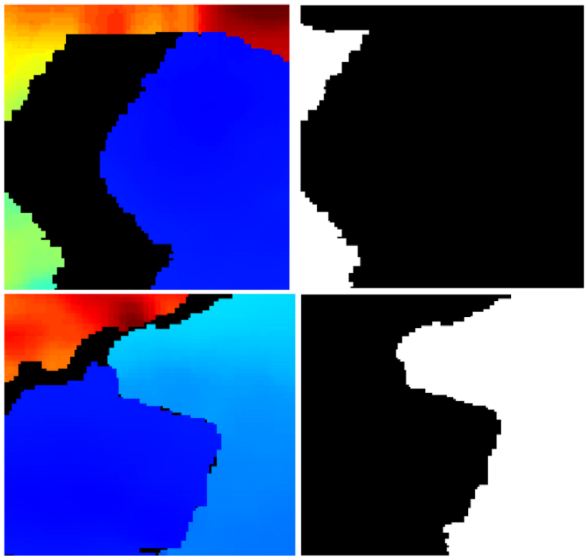


Figura 4.11: La figura mostra i ritagli, dall'intera immagine, dei riquadri ottenuti dalla prima fase e la loro binarizzazione.

questa immagine binaria, viene eseguito l'algoritmo delle componenti connesse e quindi verranno individuate le diverse componenti, ognuna con una

propria etichetta. Infine viene creato un riquadro centrato nel baricentro della componente connessa più rilevante, ovvero quella con area maggiore (vedi figura 4.12).
Tra un fotogramma e l'altro è stato limitato il massimo spostamento che il riquadro può eseguire, in modo tale che il suo movimento risulti fluido. Questo permette al sistema di essere più robusto rispetto a eventuali errori temporanei di misurazione.

Figura 4.12: Riquadri finali trovati dall'algoritmo nelle due immagini di input di esempio.

4.4.1 Pseudocodice dell'algoritmo

Lo pseudocodice dell'algoritmo per localizzare l'oggetto in un specifico fotogramma è il seguente:

```
//Ricerca degli N pixel con distanza più piccola
pixels ← creazione array di N elementi
indice_valore_massimo ← 0
valore_massimo ← 100
for ogni pixel p dell'immagine do
    distanza ← distanza del pixel p
    if distanza != 0 e distanza < valore_massimo then
        pixels[indice_valore_massimo] ← [distanza, posizione x
            di p, posizione y di p]
        indice_valore_massimo ← indice dell'array pixels
            il cui elemento contiene la distanza più grande
```

```
      valore_massimo ← pixels[indice_valore_massimo]
    end if
  end for
  //Calcolo del baricentro degli N pixel trovati
  x ← 0
  y ← 0
  for ogni elemento i dell'array pixels do
    x ← x + pixels[i][1]
    y ← y + pixels[i][2]
  end for
  x ← x / N
  y ← y / N
  //Creazione ed elaborazione della finestra
  finestra ← creazione della finestra centrata, il più
     possibile, rispetto al pixel x, y
  finestra ← ritaglio della finestra dall'intera immagine
  finestra ← conversione della finestra da RGB a GRAY SCALE
  finestra ← trasformazione della finestra in immagine binaria
  etichette, aree, baricentri ← esecuzione dell'algoritmo delle
     componenti connesse sulla finestra
  //Ricerca dell'etichetta con area maggiore
  etichetta ← 0
  area_etichetta ← 0
  for ogni etichetta e in etichette do
    area_e ← aree[e]
    if area_e > area_etichetta then
      etichetta ← e
      area_etichetta ← area_e
    end if
  end for
  x, y ← baricentro[etichetta]
  finestra ← creazione della finestra centrata, il più
     possibile, rispetto al pixel x, y
  //Limitazione dello spostamento della finestra
  finestra ← limitazione spostamento della finestra di
     massimo X pixel
```

4.5 Soluzioni alternative

Di seguito si andranno a descrivere delle soluzioni alternative, sviluppate inizialmente. Tali soluzioni sono state scartate in quanto richiedono troppo tempo per il calcolo di un singolo fotogramma e sono poco precise nell'individuazione dell'oggetto.

4.5.1 Prima soluzione

L'algoritmo della prima soluzione è il seguente:

1. L'intera immagine viene analizzata e per ogni pixel viene inserito in una lista un elemento del tipo: distanza del pixel x e y, posizione pixel x, posizione pixel y.
2. La lista ottenuta viene ordinata per valori di distanze crescenti.
3. Vengono estratti dalla lista i primi N elementi e per ciascuno di questi viene creata una finestra centrata, il più possibile, rispetto al pixel x, y dell'elemento.
4. Per ogni finestra vengono sommati i valori delle distanze ottenute dai singoli pixel.
5. Viene scelta la finestra con la somma più piccola.

In sintesi questa prima soluzione crea N finestre. Per ciascuna finestra calcola la somma delle distanze dei pixel che si trovano al suo interno e infine sceglie la finestra con la somma minore.
Per il calcolo della somma si possono utilizzare diversi metodi. In tutti i metodi presi in considerazione si penalizzano i pixel che hanno un valore di distanza uguale a 0 o superiore ad una certa soglia.

4.5.1.1 Primo metodo per il calcolo della somma

Questo metodo fa uso di una soglia costante, indicata successivamente come *S*, di due valori costanti, indicati successivamente come *C1* e *C2*, e mappa i valori delle distanze dei pixel nel seguente modo:

- se il valore della distanza del pixel x, y è 0: si assegna il valore costante C1.

- se il valore della distanza del pixel x, y è superiore alla soglia S: si assegna il risultato ottenuto dal calcolo del valore della distanza del pixel moltiplicato per il valore costante C2.

- se il valore della distanza del pixel x, y è minore o uguale alla soglia S: si assegna il valore della distanza del pixel.

In questo metodo i pixel che hanno un valore di distanza sopra alla soglia S sono penalizzati maggiormente rispetto ai pixel con distanza uguale a 0 (la figura 4.13 mostra il risultato ottenuto).

Figura 4.13: Le due immagini mostrano il risultato ottenuto utilizzando la prima soluzione con il primo metodo per il calcolo della somma.

4.5.1.2 Secondo metodo per il calcolo della somma

Il secondo metodo riprende dal primo metodo, modificando il seguente aspetto: ai pixel con valore di distanza sopra alla soglia S viene assegnato lo stesso valore costante C1 che viene assegnato ai pixel con distanza uguale a 0. In questo modo tali pixel sono penalizzati in egual modo (la figura 4.14 mostra il risultato ottenuto).

4.5.1.3 Terzo metodo per il calcolo della somma

Il terzo metodo riprende dal primo metodo, introducendo un nuovo valore costante, indicato successivamente come *C3*, e modificando il seguente

Figura 4.14: Le due immagini mostrano il risultato ottenuto utilizzando la prima soluzione con il secondo metodo per il calcolo della somma.

aspetto: i pixel con valore di distanza sopra alla soglia S sono penalizzati maggiormente, per mezzo del valore costante C3, rispetto ai pixel con distanza uguale a 0 (la figura 4.15 mostra il risultato ottenuto).

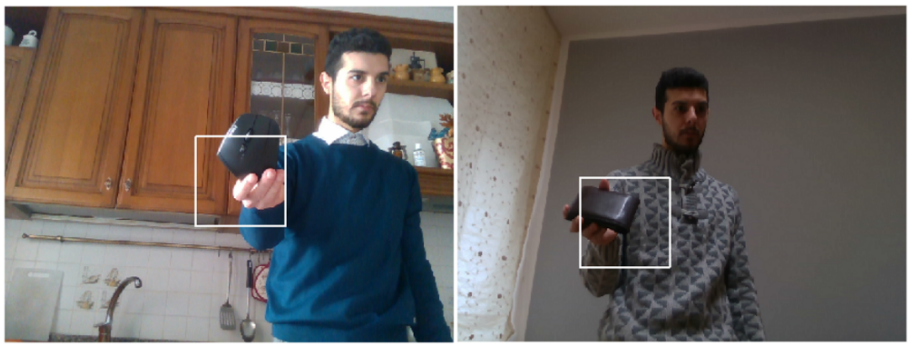


Figura 4.15: Le due immagini mostrano il risultato ottenuto utilizzando la prima soluzione con il terzo metodo per il calcolo della somma.

4.5.1.4 Quarto metodo per il calcolo della somma

Il quarto metodo riprende dal terzo metodo, introducendo due nuovi valori costanti, indicati successivamente come *C4* e *C5*, e modificando i seguenti aspetti:

1. La soglia S non è più impostata con un valore costante ma è ottenuta dalla somma del valore della distanza più piccola (il primo elemento della lista) e il valore costante C4.

2. Gli N pixel presi dalla lista ordinata, per la creazione delle finestre, non vengono estratti in modo consecutivo ma a distanza del valore costante C5 (la figura 4.16 mostra il risultato ottenuto).

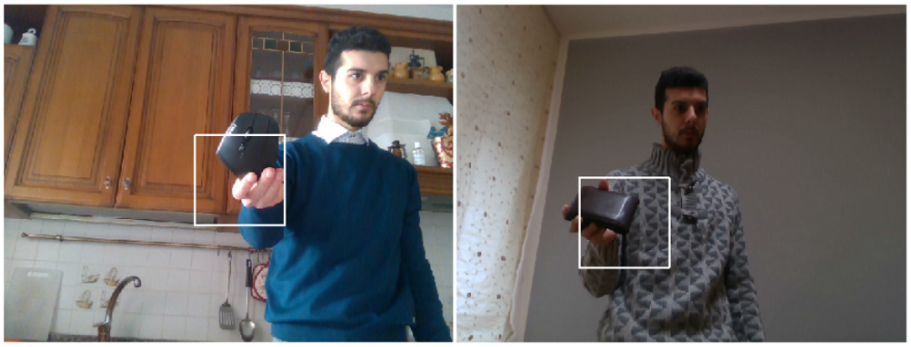


Figura 4.16: Le due immagini mostrano il risultato ottenuto utilizzando la prima soluzione con il quarto metodo per il calcolo della somma.

4.5.2 Seconda soluzione

Con la prima soluzione si ottiene una finestra che non è esattamente centrata rispetto all'oggetto di interesse. Questa soluzione cerca di centrare tale finestra.
Si sono analizzati due metodi per svolgere tale compito. Gli esempi mostrati in questi due metodi prendono come input la finestra ottenuta dalla prima soluzione con il quarto metodo per il calcolo della somma.

4.5.2.1 Primo metodo per cercare di centrare la finestra rispetto all'oggetto

In questo metodo si tiene in considerazione solo la riga e la colonna del pixel centrato nella finestra (vedi figura 4.17) e il suo funzionamento è il seguente:

1. Viene fissata una soglia.
2. Si scorre la riga finché non viene individuato un pixel che è sotto la soglia (p1).
3. Si continua a scorrere la riga finché non viene individuato un pixel che oltrepassa la soglia (p2).
4. Si calcola la posizione centrale come: $(p1 + p2)/2$.
5. Viene effettuato lo stesso procedimento per calcolare la posizione centrale della colonna.
6. La finestra viene centrata, il più possibile, rispetto al nuovo punto trovato (la figura 4.18 mostra il risultato ottenuto).

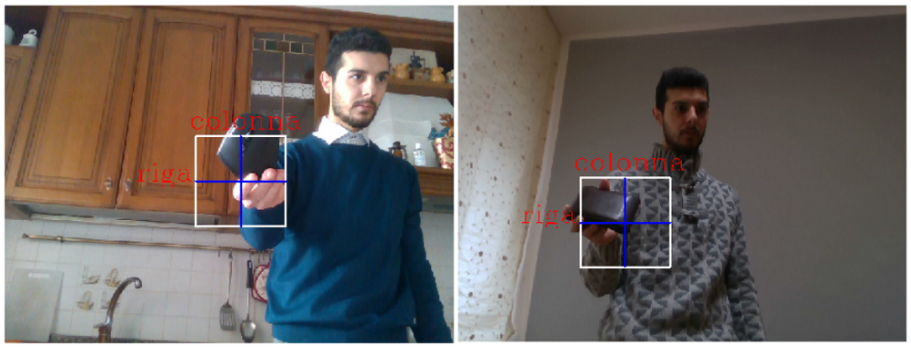


Figura 4.17: Le due immagini mostrano la riga e la colonna del pixel centrato nella finestra.

4.5.2.2 Secondo metodo per cercare di centrare la finestra rispetto all'oggetto

Il secondo metodo riprende dal primo metodo, modificando il seguente aspetto: si seleziona la riga e la colonna che hanno il maggior numero di pixel che non oltrepassano la soglia S all'interno della finestra (la figura 4.19 mostra il risultato ottenuto).

Figura 4.18: Le due immagini mostrano il risultato ottenuto utilizzando la seconda soluzione con il primo metodo per cercare di centrare la finestra rispetto all'oggetto.

4.5.3 Terza soluzione

Il tempo impiegato dalle soluzioni precedenti per calcolare un singolo fotogramma dipende dal numero di finestre prese in considerazione:

- 8000 finestre: circa 48 secondi.
- 4000 finestre: circa 24 secondi.
- 2000 finestre: circa 12 secondi.
- 1000 finestre: circa 6 secondi.
- 500 finestre: circa 3 secondi.

Le soluzioni precedenti richiedono troppo tempo. Questo è dovuto dall'enorme calcolo effettuato per ogni singola finestra presa in considerazione. La terza soluzione risolve questo problema dando in input al secondo metodo della seconda soluzione l'intero fotogramma (perciò non si calcoleranno più le N finestre). In questo modo si ottengono risultati simili, ma con un tempo decisivamente inferiore (circa 0.4 secondi). La soglia viene impostata nel seguente modo: si scorre la lista ordinata e per ogni elemento viene effettuata la differenza fra il valore della distanza dell'elemento corrente e il valore della distanza dell'elemento precedente. Se la differenza oltrepassa un certo valore viene scelto il valore della distanza dell'elemento corrente come soglia (la figura 4.20 mostra il risultato ottenuto).

Figura 4.19: Le due immagini mostrano il risultato ottenuto utilizzando la seconda soluzione con il secondo metodo per cercare di centrare la finestra rispetto all'oggetto.

4.5.4 Quarta soluzione

La terza soluzione restituisce una finestra con differenti spazi di margine fra l'oggetto e i quattro bordi della finestra. Questa soluzione prende come input il risultato prodotto dalla terza soluzione e cerca di spostare la finestra in modo da ottenere lo stesso margine di spazio.
Il procedimento per ottenere tale risultato è il seguente:

1. Viene impostata una soglia.

2. Si calcola il margine sinistro. Dal bordo sinistro della finestra vengono contate le colonne consecutive che oltrepassano la soglia: ogni colonna viene contata solo se tutti i suoi pixel oltrepassano la soglia. Il conteggio termina quando un pixel di una colonna oltrepassa la soglia.

3. Il margine destro viene calcolato con lo stesso procedimento del punto 2 (vengono contate il numero di colonne a partire dal bordo destro).

4. Anche il margine superiore ed inferiore vengono calcolati con lo stesso procedimento del punto 2 (in questo caso si contano le righe).

5. La finestra viene spostata in modo tale che il numero di colonne del margine destro e del margine sinistro siano le stesse.

Figura 4.20: Le due immagini mostrano il risultato ottenuto dalla terza soluzione.

6. La finestra viene spostata in modo tale che il numero di righe del margine superiore e del margine inferiore siano le stesse (la figura 4.21 mostra il risultato ottenuto).

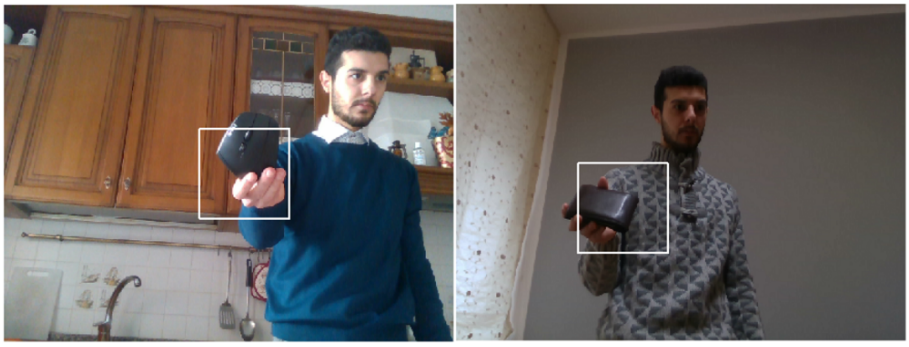


Figura 4.21: Le due immagini mostrano il risultato ottenuto dalla quarta soluzione.

4.6 Risultati sperimentali

L'algoritmo della soluzione adottata riesce a localizzare correttamente l'oggetto solo se esso è in primo piano rispetto a tutti gli altri oggetti. Questo

è dovuto dal fatto che nella prima fase dell'algoritmo vengono considerati solo gli N pixel più vicini per cercare la porzione dell'immagine da utilizzare nella seconda fase. In questi casi la precisione ottenuta è buona.
Il numero di fotogrammi al secondo dipende dal numero di pixel considerati nella prima fase. Dai test risulta che:

- considerando 22 pixel si ottengono 7/8 fotogrammi al secondo.
- considerando 45 pixel si ottengono 6/7/8 fotogrammi al secondo.
- considerando 90 pixel si ottengono 6/7 fotogrammi al secondo.
- considerando 180 pixel si ottengono 5/6 fotogrammi al secondo.
- considerando 360 pixel si ottengono 3/4 fotogrammi al secondo.
- considerando 500 pixel si ottengono 3/4 fotogrammi al secondo.

Solitamente i pixel con la distanza più piccola sono concentrati in una certa zona, perciò è più che sufficiente utilizzare dai 22 ai 90 pixel.

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi si sono affrontati i principali aspetti, tecniche e tecnologie della localizzazione degli oggetti.
Dall'analisi delle tecniche utilizzate è emerso che la localizzazione basata su reti neurali, in particolare Fast R-CNN, YOLO e SSD, riesce a raggiungere ottimi risultati ma non è adatta per piattaforme robotiche/embedded/mobile in quanto questi algoritmi sono troppo pesanti e non permettono una localizzazione fluida (con un numero di frame per secondo accettabile).
A tale scopo si è presentata la realizzazione di un progetto basato sull'uso della profondità, utilizzando la camera RealSense D435 che, rispetto alle camere Kinect v1 e Kinect v2, ha un'ottima risoluzione, un ampio campo visivo, una buona distanza massima, fino a dieci metri, e un'ottima compattezza. Al fine di ottenere la posizione di un oggetto posto in primo piano, sono stati proposti e testati diversi metodi. Sono state valutate le performance di essi, sia velocità che precisione. La soluzione proposta fa uso di componenti connesse.
Sicuramente ci sono alcune pecche dovute dalla complessità del dominio, ma nel complesso il sistema riesce a raggiungere un risultato soddisfacente (dai 6 agli 8 fotogrammi per secondo) tale da poterlo utilizzare anche in tempo reale.
Le migliorie che si potrebbero apportare sono le seguenti:

- localizzare più oggetti: al momento è possibile localizzare un singolo oggetto.

- rendere il sistema più fluido: attualmente il collo di bottiglia consiste nel numero di pixel presi in considerazione nella prima fase.
- aumentare la precisione di localizzazione: attualmente l'oggetto viene inquadrato correttamente solo se esso è in primo piano rispetto a tutti gli altri oggetti.

Ringraziamenti

Ringrazio il relatore, Davide Maltoni, e il correlatore, Lorenzo Pellegrini, per avermi offerto questo progetto e per tutti gli spunti utili alla realizzazione di questa tesi.
Un ringraziamento particolare va anche a tutti i miei famigliari e a tutte le persone a me care che mi sono state vicine durante tutto il mio percorso universitario.

Bibliografia

[1] C. Ciliberto, S. R. Fanello, M. Santoro, L. Natale, G. Metta, and L. Rosasco. On the impact of learning hierarchical representations for visual recognition in robotics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3759–3764, Nov 2013.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[3] M. B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 39(2):253–280, Apr. 1992.

[4] S. R. Fanello, C. Ciliberto, L. Natale, and G. Metta. Weakly supervised strategies for natural object recognition in robotics. In *2013 IEEE International Conference on Robotics and Automation*, pages 4223–4229, May 2013.

[5] R. Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[7] V. Lomonaco and D. Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In S. Levine, V. Vanhoucke, and

K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017.

[8] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

[9] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale. Teaching icub to recognize objects using deep convolutional neural networks. In *Machine Learning for Interactive Systems*, pages 21–25, 2015.

[10] G. Pasquale, C. Ciliberto, L. Rosasco, and L. Natale. Object identification from few examples by improving the invariance of a deep convolutional neural network. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4904–4911. IEEE, 2016.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[13] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang, et al. Openloris-object: A dataset and benchmark towards lifelong object recognition. *arXiv preprint arXiv:1911.06487*, 2019.

[14] A. Womg, M. J. Shafiee, F. Li, and B. Chwyl. Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 95–101. IEEE, 2018.