

UNIVERSITÀ
DI PAVIA

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN BIOINGEGNERIA

TESI DI LAUREA

INTEGRAZIONE VISUO-MOTORIA NELLA ROBOTICA RIABILITATIVA: PIPELINE PER RICOSTRUZIONE 3D E VALIDAZIONE SPERIMENTALE DI TASK DI PICK AND PLACE SU ROBOT TIAGo

Studentessa: Contu Federica

Relatore: Prof. Ramat Stefano

Correlatore: Dott. Soldi Roberto

Anno Accademico 2025/2026

Sommario

INTRODUZIONE 1

1. CAPITOLO UNO. STATO DELL'ARTE SULLE METODOLOGIE RIABILITATIVE PER LA NEGLIGENZA SPAZIALE UNILATERALE 4

1.1 Definizione Clinica E Quadro Patologico 4

1.1.1 Profilo Comportamentale E Diagnosi Preliminare Della Malattia 5

1.2 Approcci Riabilitativi Tradizionali Per La Negligenza Spaziale 7

1.2.1 Training Di Esplorazione Visiva 8

1.2.3 Stimolazione Sensoriale E Attivazione Motoria 9

1.3 Criticità e Limiti degli Approcci Riabilitativi Tradizionali 11

1.4 Innovazione Tecnologica: La Riabilitazione Robotica 11

1.4.1 I Robot Socialmente Assistivi (SAR) 12

2. CAPITOLO DUE. ARCHITETTURA DI SISTEMA E BACKGROUND TECNOLOGICO 14

2.1 Piattaforma Robotica TIAGo: Architettura E Capacità Operative 14

2.1.1 Dotazione Hardware: Sensoristica Di Bordo e Cinematica Del Manipolatore 15

2.1.2 Infrastruttura Software: Paradigmi Di Comunicazione Nel Framework ROS 2 18

2.2 Percezione Visiva: Il Sensore RGB-D Orbbec Astra 20

2.2.1 Principi Di Acquisizione: Ottica RGB E Infrarossi Strutturati 21

2.2.2 Flusso Dati E Gestione Dei Topic Visivi In ROS 2 22

3. CAPITOLO TRE. SVILUPPO DELLA PIPELINE VISIVA E SCELTA DELL'ARCHITETTURA NEURALE YOLO 24

3.1 Rete Neurale Per La Visione: Il Modello YOLO Per Object Detection E Segmentazione 24

3.1.1 Object Detection 24

3.1.2 YOLO: You Only Look Once 25

3.1.3 Dataset: MicroSoft Common Objects in COntext (MSCOCO) 25

3.2 Ricostruzione 3D Con Approccio Basato Su Nuvola Di Punti 26

3.2.1 Workflow Della Ricostruzione Spaziale Basata Su Nuvola Di Punti 26

3.2.2 Risultati Sperimentali: Instabilità E Limiti Fisici Del Sensore 28
3.3 Approccio Basato Su Ricostruzione Volumetrica..... 30
3.3.1 Elaborazione RGB E Segmentazione Semantica Bidimensionale 30
3.3.2 Calcolo Dimensionale 2D Invariante All'orientamento 31
3.3.4 Rendering Geometrico E Registrazione Dei Dati..... 33
3.4 Metriche Di Valutazione Delle Performance E Scelta Della Pipeline Di Ricostruzione 34
3.5 Valutazione Comparativa E Selezione Del Modello YOLO 38
3.5.1 Strumento Di Analisi Offline: Sviluppo Della GUI In Tkinter 39
3.5.2 Protocollo Di Test: Performance A Confronto..... 40
3.5.3 Analisi Comparativa Del Detection Rate 41
3.5.4 Confronto Dell'errore Dimensionale..... 41
3.5.5 Analisi Statistica Del Jitter Del Baricentro..... 45
4. CAPITOLO QUATTRO. SVILUPPO, SIMULAZIONE E VALIDAZIONE PRELIMINARE DELLA PIPELINE IN AMBIENTE ROS 2..... 49
4.1 L'ambiente Virtuale Gazebo 49
4.1.1 Impostazione Della Scena Di Simulazione Controllata 50
4.2 Implementazione E Integrazione Del Nodo Di Percezione Visiva..... 51
4.2.1 Trasformazioni Spaziali ROS 2 52
4.2.2 Registrazione Delle Immagini RGB-D..... 53
4.2.3 Generazione Dei Marker E Visualizzazione Della Scena Tramite Ambiente RViz..... 53
4.2.4 Risultati Ottenuti Dai Test Di Simulazione Per Il Nodo Di Visione 55
4.3 Sviluppo E Collaudo Del Nodo Di Pianificazione E Grasping 58
4.3.1 Trasformazioni Spaziali E Vincoli Di Orientamento: I Quaternioni 59
4.3.2 Approccio Cartesiano Lineare (Linear Approach) 59
4.3.3 Compensazione Cinematica: Applicazione Degli Offset All'end Effector 60
4.3.4 Risultati Ottenuti Dai Test Di Simulazione Per Il Nodo Di Grasping 61
4.4 Risultati Ottenuti Dalla Validazione Sperimentale Della Pipeline In Gazebo 62

5. CAPITOLO CINQUE. VALIDAZIONE SPERIMENTALE E TRASLAZIONE DELLA PIPELINE SU ROBOT FISICO IN LIVING LAB 63
5.1 Validazione Sperimentale del Nodo di Visione: Protocollo Di Acquisizione 64
5.1.1 Configurazione Cinematica E Assetto Dei Sensori 65
5.1.2 Metriche Di Validazione E Logica Di Campionamento 66
5.1.2.1 Errore Dimensionale Relativo 67
5.1.2.2 Stabilità Spaziale 3D: Jittering Del Centroide 71
5.1.2.3 Robustezza Della Percezione: Detection Rate 76
5.2 Validazione Del Nodo Di Grasping In Living Lab 78
5.2.1 Analisi Sperimentale E Discussione Dei Risultati Ottenuti 78
6 CAPITOLO SEI. CONCLUSIONI E SVILUPPI FUTURI 84
6.1 Conclusioni 84
6.2 Sviluppi E Prospettive Future 84

Introduzione

Il recupero funzionale e cognitivo a seguito di un evento neurologico acuto, con particolare riferimento a patologie invalidanti come la Negligenza Spaziale Unilaterale (USN), richiede percorsi riabilitativi complessi e prolungati. Questa condizione, in particolare, porta il paziente a ignorare e non esplorare adeguatamente lo spazio controlaterale alla lesione cerebrale (tipicamente lo spazio alla sinistra del paziente), con ricadute pesantissime sull'autonomia nelle attività quotidiane. Sebbene la medicina riabilitativa abbia consolidato nel tempo diversi protocolli di intervento, tra cui il training di esplorazione visiva, l'adattamento prismatico e la stimolazione sensoriale, la trasposizione di questi metodi nella pratica clinica quotidiana si scontra con limiti pratici e organizzativi non trascurabili. Da un lato, il carico di lavoro delle strutture sanitarie impone spesso di dover gestire molti pazienti, frammentando o riducendo il tempo effettivo dedicato alla singola seduta; dall'altro, gli esercizi proposti risultano frequentemente ripetitivi e monotoni. Questa carenza di dinamismo si traduce in un rapido calo dell'attenzione e della motivazione da parte del paziente, il quale, vivendo la terapia con noia e spesso frustrazione, finisce per inficiarne l'efficacia stessa, e abbandonare il percorso una volta superati i primi mesi di terapia.

Proprio per superare queste barriere e restituire centralità al coinvolgimento del paziente, la ricerca bioingegneristica sta guardando con sempre maggiore interesse all'introduzione della *robotica sociale e assistiva* (*SAR*) nei percorsi di cura. L'obiettivo non è sostituire la figura imprescindibile del terapista, ma fornirgli uno strumento avanzato capace di proporre una terapia controllata, quantificabile e coinvolgente. Un robot in grado di interagire fisicamente con l'ambiente, ad esempio passandogli degli oggetti, riordinando un piano di lavoro o sfidandolo in attività di manipolazione, può riaccendere l'interesse del paziente, prolungandone i tempi di attenzione. Affinché questo scenario diventi applicabile, è tuttavia necessario che il sistema robotico possieda una percezione spaziale estremamente affidabile, che gli consenta di operare e manipolare gli oggetti target in totale sicurezza.È in questo specifico snodo tecnologico che si inserisce il presente lavoro di tesi, il cui scopo è la progettazione, lo sviluppo e la validazione sperimentale di un'architettura *software* per l'esecuzione autonoma di task di *pick and place* da parte della piattaforma robotica *TIAGo* (sviluppata da *PAL Robotics*).

Il lavoro di tesi è stato svolto presso il laboratorio "*Living Lab*", del centro di eccellenza *CoE REDI* (*Center Of Excellence On Rehabilitatio Devices And Digital Instruments*) dell'università di Pavia. Il lavoro si inserisce nell'ambito della *Mission* 2 – *Activity* 5 (*Match Making* #53) del progetto *Fit4MedRob* (*Fit For Medical Robotics*), che ha l'obiettivo di sviluppare e validare un protocollo innovativo per la riabilitazione di pazienti post ictus affetti da emi inattenzione spaziale, in collaborazione con la

Fondazione Mondino. Per permettere al robot di afferrare oggetti con precisione, il primo e più critico passo è stato lo sviluppo di un sistema di visione in grado di ricostruirne l'ingombro e la posizione nello spazio tridimensionale. L'attività di ricerca e sviluppo, infatti, ha preso avvio con un'attenta valutazione delle metodologie di stima *3D*. Inizialmente, si è esplorato un approccio di visione basato sull'elaborazione dell'intera nuvola di punti (*Point Cloud*) della scena che viene presentata al robot, comprensiva quindi di piano di lavoro e oggetto target. La sperimentazione diretta ha però fatto emergere una serie di problematiche, legate principalmente all'instabilità del dato grezzo e ai limiti fisici intrinseci del sensore ottico, i quali generavano un rumore tale da compromettere l'esatto calcolo del baricentro dell'oggetto. Per aggirare questo ostacolo, si è scelto di implementare e validare un approccio alternativo di natura ibrida e volumetrica. Questo metodo prevede innanzitutto un'analisi bidimensionale dell'immagine per individuare l'oggetto e, solo in un secondo momento, estrae la profondità concentrandosi esclusivamente su una piccola regione di interesse attorno al suo centro geometrico. Dal punto di vista operativo, l'implementazione del *software* ha seguito un iter strutturato. Inizialmente, gli algoritmi sono stati scritti in *Python* e collaudati in configurazione *stand-alone* sfruttando direttamente la telecamera *RGB-D Orbbec Astra,* in dotazione anche sul robot. Questa fase preliminare si è rivelata cruciale non solo per perfezionare le logiche di estrazione della profondità, ma anche per condurre un'analisi comparativa su diverse architetture di reti neurali della famiglia *YOLO*, arrivando a selezionare il modello ottimale in base a metriche di *detection rate*, contenimento dell'errore dimensionale e riduzione del *jittering* spaziale del baricentro. Una volta consolidata la *pipeline* visiva, il sistema è stato traslato nel *framework* operativo *ROS 2*. Per garantire un passaggio sicuro alla macchina fisica, i nodi *software* per la visione e per il *grasping* sono stati dapprima sviluppati e testati all'interno dell'ambiente di simulazione virtuale *Gazebo*. Questa sperimentazione ha permesso di mettere a punto le trasformazioni spaziali e le logiche cinematiche necessarie all*'end-effector* per raggiungere il target senza incorrere in collisioni. La fase culminante del progetto ha visto infine la validazione dell'intero sistema sul robot *TIAGo* reale. Attraverso un protocollo di test che ha previsto la disposizione degli oggetti in molteplici configurazioni sul piano di lavoro, è stato possibile non solo confermare l'accuratezza del codice sviluppato, ma soprattutto mappare un cosiddetto *Safe Workspace*: un'area di lavoro ben definita sul tavolo all'interno della quale il robot garantisce prese sicure e ripetibili, ponendo le basi operative per un suo futuro impiego in sessioni riabilitative.

L'esposizione del lavoro è organizzata in sei capitoli, strutturati come segue:

Il *primo capitolo* delinea il quadro clinico della Negligenza Spaziale Unilaterale: vengono passati in rassegna gli approcci riabilitativi convenzionali, mettendone in luce le criticità pratiche, per poi introdurre le soluzioni e le prospettive offerte dall'innovazione tecnologica e dalla robotica.

Il *secondo capitolo* è dedicato all'architettura di sistema e al *background* tecnologico. Al suo interno vengono descritte le caratteristiche *hardware* e cinematiche del robot *TIAGo*, l'infrastruttura di comunicazione di *ROS 2*, il principio di funzionamento del sensore *RGB-D Orbbec Astra* e i fondamenti dell'*Object Detection* tramite il modello neurale *YOLO*.

Nel *terzo capitolo* si entra nel dettaglio dello sviluppo algoritmico. Viene ampiamente discusso il confronto metodologico tra la fallimentare ricostruzione tramite *Point Cloud* e la vincente strategia volumetrica adottata. Si illustrano inoltre gli strumenti sviluppati ad *hoc* per l'analisi dei dati offline e i criteri che hanno guidato la scelta definitiva.

Il *quarto capitolo* sposta il focus sull'implementazione del sistema nel *framework ROS 2* e sulla fase di simulazione in *Gazebo*. Qui vengono esaminati i nodi creati per la percezione visiva e per la pianificazione della traiettoria di presa, analizzando la gestione delle trasformazioni spaziali e i risultati preliminari ottenuti nell'ambiente virtuale.

Il *quinto capitolo* raccoglie i risultati della validazione sperimentale condotta sul robot reale. Vengono presentate le metriche di accuratezza spaziale e robustezza della percezione, discutendo criticamente i dati acquisiti e arrivando alla definizione e mappatura dell'area di lavoro sicura (*Safe Workspace*).

Infine, il *sesto capitolo* pone le conclusioni della tesi e gli sviluppi futuri, sintetizzando i traguardi ingegneristici raggiunti nel corso del progetto e delineando le possibili traiettorie di sviluppo futuro in ottica clinico-riabilitativa.

Capitolo Uno. Stato Dell'arte Sulle Metodologie Riabilitative Per La Negligenza Spaziale Unilaterale

In questo primo capitolo verrà discusso un primo inquadramento clinico della patologia, in cui verranno dettagliati i comportamenti e sintomi tipici dei pazienti che ne sono colpiti e alcune delle metodologie diagnostiche utilizzate. In seguito, verrà delineato lo stato dell'arte relativo alle metodologie di riabilitazione per la Negligenza Spaziale Unilaterale (*NSU*), attraverso una panoramica degli approcci terapeutici, partendo dalle tecniche cliniche tradizionali fino ad arrivare all'integrazione delle più recenti tecnologie innovative: i robot sociali assistivi.

1.1 Definizione Clinica E Quadro Patologico

L'ictus, sia esso ischemico o emorragico è una condizione debilitante e invalidante che può compromettere le funzioni cognitive, visuospaziali, linguistiche e motorie dei pazienti in funzione dell'area corticale interessata. Il *neglect* spaziale unilaterale è un disturbo neurologico caratterizzato da un deficit dell'attenzione, della percezione e dell'azione da parte del paziente a stimoli sul lato controlaterale alla lesione cerebrale, che nella maggior parte dei casi registrati di *neglect* si manifesta sul lato destro dell'emisfero cerebrale e porta, di conseguenza, ad un *neglect* sul lato sinistro dello spazio, (*Figura 1.1*) [1], [2].

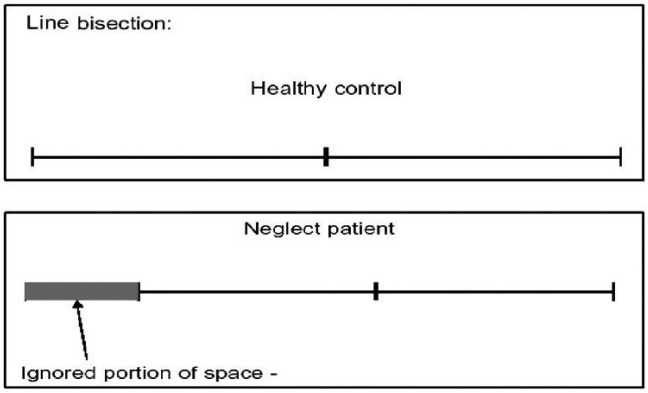


Figura 1.1 Confronto tra un soggetto sano e un paziente con *NSU* nel test di bisezione di linee. Al paziente viene chiesto di tracciare una linea al cento del segmento, indice della sua corretta percezione degli spazi destro e sinistro. Lo spostamento del segno verso destra da parte del paziente colpito da neglect evidenzia l'incapacità del paziente di elaborare correttamente gli stimoli provenienti dall'emispazio sinistro [1].

In letteratura, il *neglect* destro (causato da lesioni all'emisfero sinistro) viene spesso descritto come un disturbo più raro, meno grave e di durata inferiore rispetto a quello sinistro. In realtà, questa visione è spesso falsata da limiti diagnostici: i pazienti con danni all'emisfero sinistro soffrono frequentemente di disturbi del linguaggio (afasia) che

ostacolano la normale comprensione ed esecuzione dei test cognitivi, portando così a una forte sottostima del problema visivo. A livello anatomico, l'osservazione di danni ai circuiti fronto-parietali e alla materia bianca dell'emisfero sinistro dimostra che anche questa parte del cervello svolge un ruolo attivo nell'orientamento dell'attenzione spaziale. Queste evidenze più recenti invitano a riconsiderare i modelli neurologici classici, i quali attribuivano il controllo dell'attenzione quasi esclusivamente all'emisfero destro [3], [4].

Oltre al *neglect* sensoriale, può verificarsi un *neglect* motorio che si manifesta come un uso ridotto o nullo delle estremità contro lesionali durante la deambulazione o le attività bimanuali [5]. Tale deficit dato dal *neglect* non è da attribuire a compromissioni motorie o sensoriali primarie (ossia legate a lesioni degli organi di senso o delle vie motorie dirette) ma riflette, piuttosto, un'alterazione nei processi attenzionali di ordine superiore (complessi meccanismi di controllo attenzionale che si basano su reti neurali distribuite nel sistema nervoso centrale), che dunque non consentono al paziente di svolgere alcuni movimenti nella maniera corretta [6], [7].

1.1.1 Profilo Comportamentale E Diagnosi Preliminare Della Malattia

I pazienti colpiti da *neglect* presentano sintomatologie comuni: possono dirigere lo sguardo verso il lato della lesione quando, ad esempio, dialogano con una persona, al punto da non guardare l'interlocutore; possono consumare il cibo solamente in una metà del piatto, o anche prestare meno attenzione a un lato del proprio corpo durante l'igiene personale (condizione definita come *neglect* personale).

Tradizionalmente, i test clinici preliminari per la diagnosi di *neglect* impiegano test carta e matita (*test paper-and-pencil)* e includono test di cancellazione, la copia e il disegno di oggetti (*Figura 1.2* e *Figura 1.3*) [8]. Tali test richiedono ai soggetti di individuare dei target su un foglio posizionato centralmente su un piano di lavoro situato davanti al paziente: i soggetti colpiti da *neglect* tendono a iniziare la ricerca dal margine ipsilesionale (lato destro, il medesimo della lesione) del foglio omettendo del tutto i target controlesionali [9].

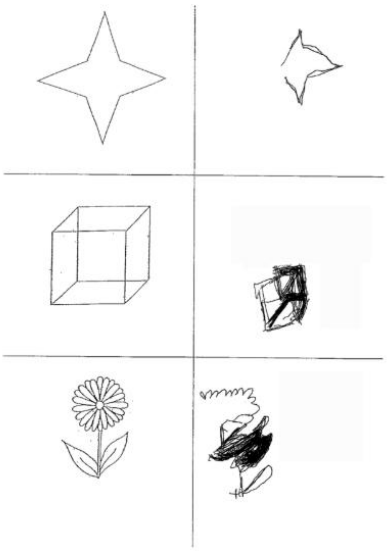


Figura 1.2 Test di copia: lo scopo è disegnare una copia del simbolo che si osserva sulla parte sinistra del foglio. I pazienti affetti da neglect tendono ad omettere gli elementi situati sul lato sinistro di ciascun oggetto [9].



Figura 1.3 Test di cancellazione: lo scopo è trovare un simbolo target tra altri simboli distrattori, in questo caso le figure a forma di stella più piccole. I pazienti affetti da neglect individuano bersagli esclusivamente sul lato ipsilesionale [8].

1.2 Approcci Riabilitativi Tradizionali Per La Negligenza Spaziale

La riabilitazione post-ictus comporta tipicamente un processo clinico che include tre fasi principali:

- Valutazione: Per identificare e quantificare i bisogni del paziente
- Definizione degli obiettivi: Per stabilire traguardi di miglioramento realistici e raggiungibili.
- Intervento: Per supportare il raggiungimento degli obiettivi concordati [10].

Gli interventi riabilitativi tradizionali includono lo sfruttamento delle funzioni cerebrali intatte residue, l'adattamento alla compromissione mediante l'uso di dispositivi prostetici o modifiche ambientali, o il riallenamento della funzione compromessa [1]. Il percorso di recupero *post ictus* è tipicamente classificato in tre stadi temporali, ciascuno caratterizzato da specifici bisogni riabilitativi: acuto, subacuto e cronico. Durante la fase acuta (dalle prime 24 ore fino a una settimana dall'esordio), l'obiettivo primario è la stabilizzazione medica. In questa finestra critica, l'introduzione di iniziative di riabilitazione precoce, già entro le prime 24/48 ore, risulta fondamentale per prevenire complicanze secondarie come contratture muscolari, ulcere da pressione e trombosi venosa profonda. Interventi come gli esercizi di *range-of-motion* passivo, ovvero la mobilizzazione passiva dell'arto controlesionale, la mobilizzazione precoce e la stimolazione sensoriale (tattile, uditiva o visiva) sono impiegati per innescare fin da subito la neuroplasticità nei circuiti cerebrali colpiti. Successivamente, durante la fase subacuta (da una settimana a 3-6 mesi), il paziente intraprende un percorso di riabilitazione tradizionale intensivo e multidisciplinare. Questo approccio si basa sulla sinergia di diverse discipline: la fisioterapia, focalizzata sul ripristino della mobilità, dell'equilibrio e della coordinazione, e la terapia occupazionale, mirata a restituire l'indipendenza nelle attività della vita quotidiana (*ADL, Activities of Daily Living*) attraverso allenamenti orientati al compito (*task oriented*). A queste si affiancano l'allenamento propriocettivo e gli esercizi a doppio compito (*dual task*), i quali richiedono al paziente di eseguire un compito motorio contemporaneamente a uno cognitivo, favorendo così un recupero funzionale e neurologico integrato. Durante la fase cronica, infine, che va dai 6 mesi successivi all'evento e si protrae a tempo indeterminato, l'attenzione si sposta sulla riabilitazione a lungo termine, sulla gestione delle disabilità residue e sul mantenimento attivo delle capacità funzionali, avvalendosi eventualmente di terapie avanzate come la riabilitazione robot assistita, la realtà virtuale e la stimolazione elettrica funzionale [11] .

1.2.1 Training Di Esplorazione Visiva

I primi metodi studiati per la riabilitazione tradizionale del *neglect* spaziale utilizzano tecniche empiriche per stimolare la direzione dello sguardo verso sinistra, tramite sforzo volontario del paziente che segue le istruzioni di un terapista specializzato [2], [5]. Sono stati sviluppati molti programmi che includono variazioni della natura degli stimoli, nel *feedback* o nella durata, nel numero e nella frequenza delle sessioni. La maggior parte dei programmi si articola in 4 settimane per un totale di 20 ore di esercizio alla settimana, in cui si riallenano compiti funzionali come leggere, scrivere, vestirsi o guidare una sedia a rotelle. Viene anche sfruttato l'uso di video per mostrare ai pazienti le loro omissioni durante lo svolgimento dei compiti videoregistrati [2]. Tra gli esercizi abbiamo ad esempio la presentazione di stimoli su foglio di carta o schermo, in cui il paziente deve individuare un oggetto o più di uno tra vari elementi di distrazione [5].

1.2.2 Adattamento Prismatico

L'adattamento prismatico è una tecnica riabilitativa molto diffusa, in cui il paziente indossa occhiali con lenti schermate da maschere opache (prismi ottici) che spostano lateralmente, verso il lato contro lesionale, il campo visivo durante compiti visuo-motori [6], [7]. Vengono eseguiti esercizi di puntamento in direzione di bersagli in varie posizioni dello spazio, in modo da studiare la plasticità delle funzioni sensomotorie e la capacità di adattamento del paziente [2], [5]. Attraverso l'esecuzione di tali compiti motori, il sistema nervoso centrale del paziente attua un processo di compensazione per correggere l'errore di puntamento, arrivando a riallineare con successo le mappe interne visive e propriocettive [12]. Se il paziente va in contro a adattamento, infatti, i movimenti di puntamento risulteranno spostati verso la direzione opposta alla distorsione introdotta dai prismi, dunque verso sinistra [2], [5]. Evidenze cliniche dimostrano che questo riallineamento sensomotorio non si limita a correggere la deviazione patologica della linea mediana soggettiva, ma si traduce in un miglioramento drastico delle principali manifestazioni cliniche del *neglect*. Come confermato dalle prestazioni in test neuropsicologici standardizzati, tra cui la bisezione di linee, i *task* di cancellazione, la copia di disegni e la lettura, i pazienti mostrano un recupero significativo dell'esplorazione spaziale verso l'emispazio controlesionale (*Figura 1.4*).

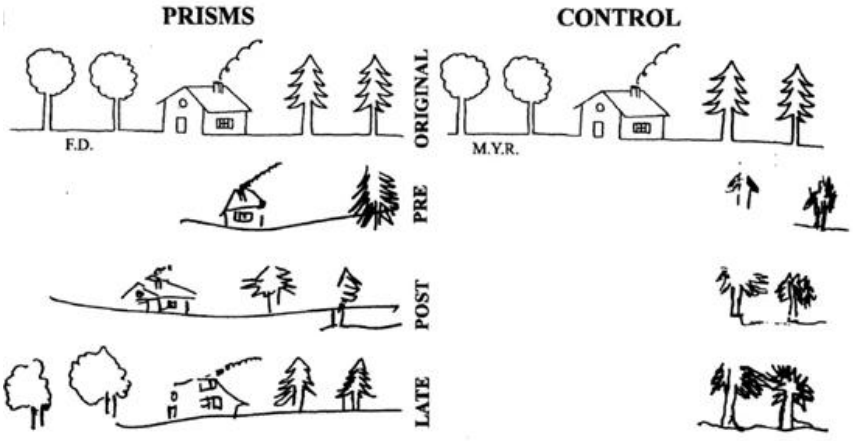


Figura 1.4 Confronto tra paziente trattato con adattamento prismatico (immagine a sinistra) e paziente che ha eseguito la riabilitazione con occhiali neutri (immagine a destra).

Nella *Figura 1.4*, nel riquadro di sinistra sono riportati i risultati di un paziente (F.D.) che ha effettuato la riabilitazione con adattamento prismatico. Possiamo notare per prima cosa il disegno realizzato prima dell'esposizione ai prismi (*pre-test*), in cui osserviamo la completa omissione di tre elementi, che nel disegno originale stanno sulla sinistra. Nel *post-test* (al momento della rimozione dei prismi), viene aggiunto un elemento al disegno del paziente, mentre nel test tardivo (*late-test*), vengono disegnati tutti gli elementi. Al contrario (riquadro di destra), la paziente (M.V.R.) esposta a occhiali neutri non ha mostrato alcun miglioramento nel *post-test* e nel *test* tardivo.

1.2.3 Stimolazione Sensoriale E Attivazione Motoria

Una tipica sintomatologia osservata nei pazienti affetti da *neglect* riguarda la presenza di disturbi dell'equilibrio [2] e una deviazione della posizione della testa e degli occhi che presentano un *bias* sistematico di 30 gradi verso il lato ipsilesionale (lato destro)[5]. Nei soggetti sani, infatti, le porzioni sinistra e destra dello spazio sono rappresentate in modo bilanciato in un sistema di riferimento centrato sul corpo (spazio egocentrico), e la lesione unilaterale può alterare questo equilibrio [2]. Queste tecniche mirano a ripristinare l'orientamento degli occhi, della testa e del corpo del paziente nella posizione di riposo verso la corretta posizione dritto-avanti [5]. Gli approcci si basano dunque sulla manipolazione dell'ambiente sensoriale del paziente, tramite stimolazioni vestibolari caloriche e galvaniche, stimolazioni opto cinetiche e stimolazione elettrica dei muscoli del collo [2], [5]. La stimolazione vestibolare calorica (*CVS*: *Caloric Vestibular Stimulation*) viene eseguita irrigando con acqua, alternativamente più calda o fredda rispetto alla temperatura corporea, l'orecchio contro lesionale che stimola il canale semicircolare orizzontale del sistema vestibolare. Propagandosi nell'orecchio interno, l'acqua va a provocare un moto convettivo dell'endolinfa (liquido contenuto nei canali

semicircolari) e induce un nistagmo vestibolare (movimento oscillatorio e involontario degli occhi), ovvero un riflesso vestibolo-oculare che il paziente percepisce accompagnato da una leggera e temporanea vertigine [5],[13]. Sperimentalmente, la *CVS* ha dimostrato di produrre miglioramenti significativi non solo nei classici test visuo-spaziali, ma anche nei punteggi delle scale di valutazione posturale (*Postural Assessment Scale for Stroke* - *PASS*), necessarie per monitorare i progressi motori e il recupero dell'autonomia e della deambulazione [13].

La stimolazione vestibolare galvanica (*GVS*: *Galvanic Vestibular Stimulation*) stimola elettricamente il sistema vestibolare tramite piccole intensità di corrente provenienti da due elettrodi applicati ai processi mastoidei sinistro e destro [5]. L'applicazione di una corrente elettrica a bassa intensità determina la depolarizzazione dei siti catodici (favorendo l'eccitazione) e l'iperpolarizzazione di quelli anodici (inducendo inibizione), potendo sfruttare forme d'onda incrementali, sinusoidali, impulsi brevi o rumore bianco a media nulla. Grazie alle sue proprietà di sicurezza, economicità e rapidità d'azione, la *GVS* si è dimostrata un trattamento complementare molto promettente per migliorare il controllo posturale, ridurre l'oscillazione corporea e stabilire risposte di bilanciamento in adulti affetti da diverse patologie neurologiche, tra cui la malattia di Parkinson, la vestibulopatia bilaterale o unilaterale e i disturbi dell'equilibrio post-ictus [5], [14].

La stimolazione optocinetica sfrutta la naturale integrazione delle informazioni visive, in particolare del movimento, nella percezione corporea e nel controllo posturale dello spazio, sottoponendo il paziente a stimoli dinamici mentre mantiene la posizione seduta su una piattaforma oscillante [5], [15]. La stimolazione visiva può essere generata, ad esempio, mediante una sfera rotante che proietta un flusso continuo di punti luminosi su una struttura cilindrica circostante a una velocità costante, ad esempio di 20° al secondo. Una rotazione antioraria della sfera produrrà anche in questo caso una risposta oculomotoria di tipo nistagmico, con una fase lenta diretta verso il lato negletto. La stimolazione vibratoria ripetuta dei muscoli del collo (*rNMV: Repeated Neck Muscle Vibration*) viene eseguita con il paziente in posizione seduta in una stanza buia, applicando un apposito vibratore a una frequenza di 80 Hz e un'ampiezza di 0.4 mm per una durata di 10 minuti mentre il soggetto è bendato. Il dispositivo viene posizionato manualmente dall'esaminatore sul lato sinistro del collo del paziente, individuando la sede precisa in base, ad esempio alla percezione di massima deviazione di un bersaglio visivo. L'effetto della vibrazione è un intervento che produce una percezione di deviazione dell'asse corporeo e riduce l'asimmetria nei pazienti con danno all'emisfero destro [5], [16].

1.3 Criticità e Limiti degli Approcci Riabilitativi Tradizionali

Le tecniche riabilitative tradizionali presentano importanti limitazioni dovute, tra i vari fattori, all'elevata eterogeneità clinica con cui la negligenza spaziale si manifesta nel singolo paziente; di conseguenza, l'applicazione del medesimo esercizio può produrre esiti terapeutici estremamente variabili. Le valutazioni cliniche, inoltre, non riescono a cogliere a pieno la complessità della negligenza spaziale unilaterale, rendendo difficile personalizzare gli interventi in modo efficace e proporre protocolli riabilitativi clinicamente rilevanti e specializzati per il singolo paziente. Gli interventi convenzionali mancano frequentemente di validità sperimentale, in quanto vengono erogati da terapisti attraverso interazioni dirette e non prevedono una quantificazione immediata e oggettiva dei progressi e di conseguenza un adattamento immediato della difficoltà degli esercizi per ogni specifico paziente [6]. Le tecniche riabilitative tradizionali presentano inoltre dei protocolli che spesso implicano lunghi tempi di esecuzione sotto la supervisione costante di professionisti qualificati. Per carenza di risorse questo non è sempre possibile, perciò viene ridotto il tempo disponibile per la riabilitazione. Gli esercizi tradizionali inoltre risultano monotoni e ripetitivi per i pazienti, che non si sentono stimolati e perdono interesse nello svolgere le sedute riabilitative [17].

1.4 Innovazione Tecnologica: La Riabilitazione Robotica

I progressi tecnologici hanno trasformato radicalmente il panorama della riabilitazione post ictus, introducendo approcci innovativi volti a migliorare il recupero motorio, le abilità funzionali e la qualità della vita complessiva dei pazienti colpiti dalla patologia. In particolare, nella riabilitazione dell'arto superiore, la robotica svolge un ruolo fondamentale offrendo una terapia precisa e intensiva attraverso dispositivi come i robot *end-effector* che forniscono esercizi su misura, guidando i pazienti attraverso movimenti mirati a migliorare il controllo motorio, la forza e il *range of motion* dell'arto colpito [11], [8]. Questo approccio, in generale, consente ai terapisti di personalizzare i piani riabilitativi sulla base delle specifiche problematiche del paziente in esame favorendo interventi più focalizzati e di maggiore impatto. I sistemi robotici, inoltre, forniscono ai terapisti capacità di monitoraggio e *feedback* in tempo reale, tracciando metriche prestazionali come, ad esempio, l'accuratezza del movimento durante un esercizio. L'integrazione di algoritmi adattivi basati sul *machine learning* e sull'intelligenza artificiale adatta gli interventi riabilitativi alle esigenze individuali, analizzando le risposte dei pazienti in tempo reale e consentendo di regolare i parametri terapeutici per massimizzare l'efficacia dei protocolli di riabilitazione [11]. L'utilizzo di trattamenti di riabilitazione robotica consente ai pazienti di eseguire dunque movimenti ripetitivi e intensi mirati al compito avvalendosi di appropriato supporto e fornendo un approccio

multidisciplinare completo, evitando al paziente frustrazione, noia e fornendo terapie ottimizzate [8].

1.4.1 I Robot Socialmente Assistivi (SAR)

I robot socialmente assistivi sono progettati per fornire ai pazienti assistenza e supporto sul piano sociale, svolgendo ruoli in vari ambiti, tra cui: monitoraggio e gestione dei sintomi della patologia e/o malattia, gestione di programmi di somministrazione farmaci, supporto ai pazienti in compiti terapeutici durante sedute riabilitative e riduzione dello stress [18], [19]. I robot socialmente assistivi sono utilizzati principalmente per migliorare le funzioni motorie e cognitive, motivare i pazienti e migliorare la qualità della vita complessiva, soprattutto per individui che si stanno riprendendo da un ictus, che gestiscono malattie croniche o che presentano disturbi muscoloscheletrici. Questi robot utilizzano tecniche di comunicazione verbale e gestuale e contatto visivo per coinvolgere i pazienti, guidandoli negli esercizi di riabilitazione [19]. Uno degli aspetti di maggiore importanza per i robot socialmente assistivi è riuscire a motivare il paziente durante gli esercizi. L'autoefficacia, ovvero la percezione che una persona ha delle proprie competenze, è correlata alla motivazione, ed un modo per influenzarla risiede nel fornire *feedback* continui di confronto sociale al paziente tramite il robot stesso [20]. Un esempio emblematico in letteratura è l'impiego del robot interattivo *PARO*, che presenta le sembianze di un cucciolo di foca, ed è dotato di un'intelligenza artificiale progettata per fornire stimolazione emotiva e relazionale. A livello tecnico, il dispositivo è equipaggiato con una rete di sensori che gli permettono di elaborare stimoli tattili e uditivi; grazie a un processore integrato, il robot risponde attivamente alle interazioni del paziente emettendo suoni e muovendo le pinne e gli occhi. In ambito riabilitativo, posizionando *PARO* nell'emispazio affetto da neglect, questa responsività multisensoriale incoraggia il paziente a orientare l'attenzione e a interagire fisicamente con il lato trascurato. Studi clinici hanno dimostrato come la stimolazione sociale offerta da questo dispositivo produca miglioramenti misurabili non solo nella partecipazione sociale del paziente, ma anche nei sintomi dell'eminegligenza visuospaziale [21].

Un altro approccio è rappresentato dai sistemi robotici dedicati in particolare al recupero motorio, per esempio dell'arto superiore (*Robot-Assisted Arm Training*), che integrano dinamiche tipiche dell'interazione tra uomo e robot. L'utilizzo di esoscheletri abbinati ad ambienti virtuali, *feedback* audiovisivi e *task* ludici (come il raggiungimento di bersagli a schermo) permette infatti di reindirizzare l'attenzione del paziente verso l'emispazio controlesionale in modo adattivo e stimolante [20]. In particolare, nei pazienti in fase subacuta, l'uso degli esoscheletri migliora significativamente il controllo motorio, la velocità di deambulazione e l'indipendenza nel cammino, con ricadute positive sulle attività della vita quotidiana e sulla partecipazione sociale. Inoltre, l'efficacia di questi

dispositivi varia in base alla loro architettura: mentre gli esoscheletri basati su *treadmill* (*tapis roulant*) risultano particolarmente efficaci per il miglioramento dell'equilibrio, gli esoscheletri *over-ground* (che permettono di camminare sul pavimento) offrono vantaggi superiori nell'ottimizzazione dei parametri del passo, come la lunghezza e la cadenza. Questi risultati sottolineano l'importanza di una prescrizione tecnologica mirata, in cui il dispositivo robotico viene selezionato in base alla specifica fase di recupero del paziente e agli obiettivi terapeutici prefissati [22].

La realtà virtuale immersiva, d'altra parte, sfrutta l'elevato livello di immersione in un mondo illusorio e l'interazione multisensoriale per far sentire il soggetto parte integrante dell'ambiente virtuale, migliorando la compliance al trattamento, incrementando la motivazione e offrendo benefici aggiuntivi sul recupero della destrezza degli arti superiori, delle prestazioni di marcia e dell'equilibrio dinamico rispetto alla sola terapia standard. I protocolli mirati agli arti superiori prevendono esercizi in cui i pazienti devono eseguire compiti di raggiungimento, presa e rilascio, ad esempio di una palla dentro un cesto, oppure giocano per allenare specificamente la funzione di presa e i movimenti di mani e avambracci. Per quanto riguarda gli arti inferiori e l'equilibrio, vengono impiegati programmi di allenamento sul tapis roulant integrati con ambienti virtuali e flussi ottici (*optic flow*), sfruttando grandi proiezioni o visori per favorire il recupero della marcia e il controllo posturale [23].

Un ultimo esempio che possiamo mettere in evidenza è rappresentato dai robot *end effector*, che intervengono guidando esclusivamente la parte distale dell'arto colpito (come la mano o il polso). Questa tecnologia è stata ampiamente impiegata negli studi clinici attraverso dispositivi come il braccio planare *Neuro-X* per indurre movimenti della spalla e del gomito, l'*InMotion 2* per il raggiungimento interattivo di bersagli, o l'*Amadeo Robotic Device* per l'allenamento specifico della mano (eseguendo, ad esempio, esercizi di presa e rilascio con le dita). La caratteristica centrale di questi trattamenti è l'associazione del movimento passivo, attivo-assistito o attivo dell'estremità paretica con interfacce ludiche e continui feedback visivi o acustici su monitor (come l'animazione dei movimenti della propria mano). Questa stimolazione combinata aiuta a richiamare i meccanismi di attenzione verso lo spazio controlesionale, contribuendo a migliorare la percezione della linea mediana valutata, ad esempio tramite il *Line Bisection Test* [24].

Capitolo Due. Architettura Di Sistema E Background Tecnologico

Nel presente capitolo verrà descritta l'infrastruttura tecnologica alla base dell'intero lavoro di tesi dal punto di vista *hardware* e *software*. In particolare, verrà ampliamente dettagliata la piattaforma robotica *TIAGo*, con particolare focus sulla telecamera *Orbbec Astra* tramite cui il robot può acquisire informazioni visive sul mondo esterno per eseguire task di vario genere, tra cui navigazione, *pick and place*, interazione sociale, monitoraggio del paziente e altri. Affinché il robot possa interagire autonomamente con l'ambiente esterno, deve comprendere ciò che lo circonda per compiere azioni mirate, come ad esempio afferrare un oggetto. Questo processo si basa su un ciclo continuo di operazioni che parte dalla percezione: attraverso i propri sensori, il robot acquisisce informazioni sull'ambiente circostante e costruisce una mappa dettagliata dello spazio operativo in cui deve agire. Una volta individuato un elemento di interesse, ad esempio un oggetto da afferrare, entra in gioco la pianificazione (o *path planning*), tramite la quale il sistema calcola un'eventuale traiettoria di navigazione per avvicinarsi all'oggetto e, dopo una nuova fase di percezione, la traiettoria ottimale che il braccio robotico deve seguire dalla sua posizione corrente fino al punto esatto in cui si trova il *target*. Infine, il nodo di controllo dei movimenti traduce questo percorso in comandi motori precisi, guidando fisicamente l'*end effector* del robot verso l'obiettivo. Tuttavia, per manipolare un oggetto con successo è necessario avere un accurato sistema di posizionamento, raccogliendo continuamento dati dai sensori; dunque, il sistema stima e aggiorna in tempo reale la postura del robot e le coordinate delle sue articolazioni, correggendo eventuali errori di misurazione. È proprio in questa delicata fase di osservazione e interazione che assumono un ruolo cruciale gli strumenti che verranno descritti nel seguente capitolo, tra cui la telecamera *RGB-D Orbbec Astra,* che funge da sistema di visione per il robot *TIAGo* e lo rende capace di identificare e localizzare gli oggetti *target*.

2.1 Piattaforma Robotica TIAGo: Architettura E Capacità Operative

Sviluppato dall'azienda *PAL Robotics*, *TIAGo* è un robot di servizio mobile e manipolatore, interamente basato sul software *ROS* (*Robot Operating Sysem*), supportato da simulazioni 3D e pacchetti *open source* per facilitarne la programmazione e la ricerca [25].

TIAGo è progettato per eseguire in sicurezza diversi *task*, che possono essere sintetizzati in tre aree fondamentali:

- Navigazione autonoma: Nella sua architettura sono presenti una base mobile e dei sensori laser (*LiDAR: laser range-finder*) che li consentono di mappare lo spazio che lo circonda, evitando ostacoli e muovendosi autonomamente in sicurezza negli spazi interni;

- Manipolazione di oggetti: Presenta un torso sollevabile che ne estende lo spazio di lavoro verticale e un braccio articolato a sette gradi di libertà (*DoF*) dotato di sensori per garantire continui feedback sulla posizione; perciò, può essere equipaggiato con diversi *end-effector* intercambiabili (*gripper*) o mani antropomorfe, per afferrare degli oggetti, che sono eventualmente dotati di sensori di forza;
- Percezione e interazione: La testa di *TIAGo* è dotata di una telecamera *RGB-D* per consentire il riconoscimento visivo e stimare correttamente la profondità dello spazio e degli oggetti, oltre a microfoni e autoparlanti per interagire con gli utenti (sensore *stereo microphone)*. Inoltre, è presente anche un *monitor touch* per consentire il controllo di *TIAGo* e garantire la possibilità di erogare esercizi [25].

2.1.1 Dotazione Hardware: Sensoristica Di Bordo e Cinematica Del Manipolatore

L'architettura *hardware* di *TIAGo* è composta da una molteplicità di sensori e componenti essenziali per il suo funzionamento, il cui *layout* complessivo è illustrato in *Figura 2.1*.

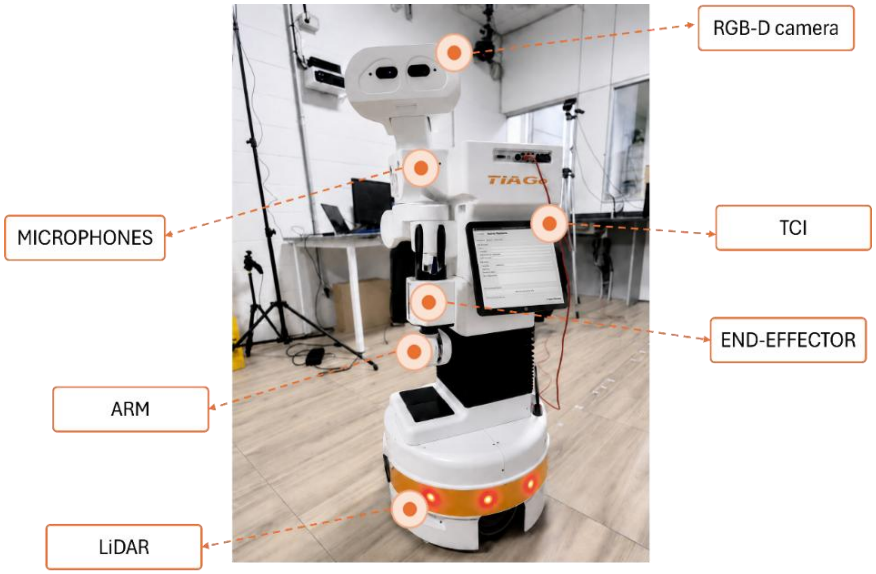


Figura 2.1 Componenti hardware principali del robot di servizio TIAGo utilizzato per il progetto di tesi.

La parte superiore di *TIAGo* è montata su un meccanismo *pan-tilt* a due gradi di libertà e dal punto di vista percettivo, la testa integra una telecamera *RGB-D* (modello *Orbbec Astra S*) ottimizzata per la visione ravvicinata, con un raggio di rilevamento della profondità compreso tra 0.4 e 2 metri. L'interazione vocale è invece garantita da un microfono stereo (*Andrea SuperBeam*) e da un autoparlante (*VISATON)* posizionato nella zona del collo. La struttura è concepita per essere espandibile, perciò la sommità del busto

presenta una superficie piatta dotata di punti di fissaggio che consentono l'installazione di sensori o hardware aggiuntivi, o se necessario consentono di posizionare un computer portatile per la gestione e l'utilizzo di *TIAGo* (*Figura 2.2*) [25].

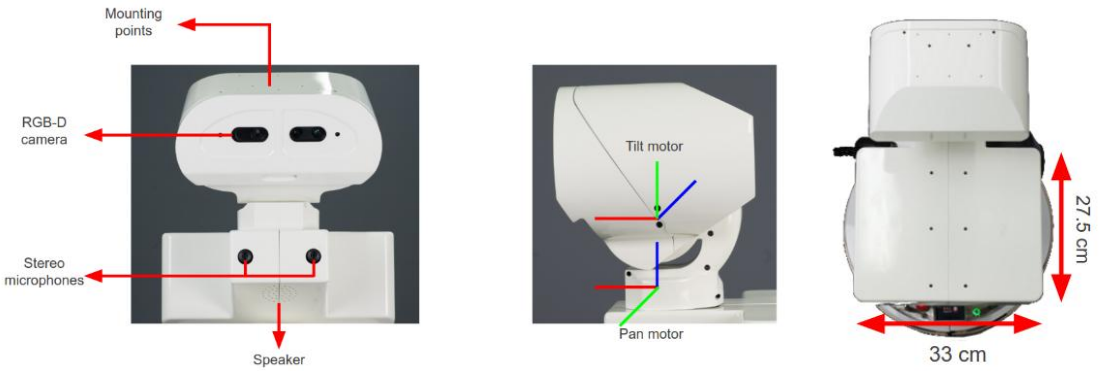
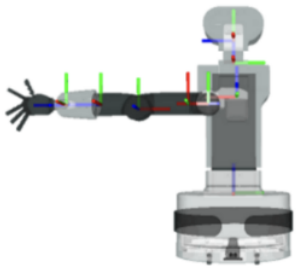


Figura 2.2 Componenti, cinematica e ingombri della parte superiore di TIAGo. Sulla sinistra vediamo i sensori visivi e uditivi (telecamera RGB-D, microfoni stereo e autoparlante) e i punti di ancoraggio sulla sommità. Il pannello centrale mette in evidenza la configurazione dei due giunti rotoidali (pan e tilt) che orientano la testa. Sulla destra, infine, viene mostrato lo spazio utile a disposizione (33 cm x 27,5 cm) per l'eventuale alloggiamento di un pc portatile o di hardware aggiuntivo [25].

Il torso di *TIAGo* costituisce l'elemento strutturale centrale del robot fungendo da supporto per il braccio manipolatore e la testa. Al suo interno integra un meccanismo di sollevamento lineare (*lifter*) che permette di variare dinamicamente l'altezza del robot da un valore minimo di 110 cm a un massimo di 145 cm.

Per pianificare correttamente i movimenti del robot ed evitare collisioni è fondamentale conoscere i limiti cinematici del sistema. La catena cinematica superiore di *TIAGo (Figura 2.3)* è composta da una combinazione di giunti rotoidali e un giunto prismatico, che permette l'escursione verticale. La testa di *TIAGo* possiede due gradi di libertà gestiti da giunti rotoidali (*head_1 e head_2*) utili per il puntamento dei sensori visivi, mentre il braccio robotico è costituito da sette giunti rotoidali (*arm_1 - arm_7*). La tabella in *Figura 2.3* riporta i limiti operativi inferiori e superiori (*lower e upper limit*) per ciascun asse: è importante notare che l'installazione del sensore di forza e coppia al polso riduce l'escursione massima del giunto *arm_6*, ma nella configurazione presente in *Living Lab (TIAGo-229*), il *parallel gripper* non è dotato di tale sensore [25].



Joint	Type	Lower limit	Upper limit
torso	prismatic	0 mm	350 mm
head_1	revolute	-75º	75º
head_2	revolute	-60º	45º
arm_1	revolute	0	157.5º
arm_2	revolute	-90º	62.5º
arm_3	revolute	-202.5º	90º
arm_4	revolute	-22.5º	135º
arm_5	revolute	-120º	120º
arm_6	revolute	-90º (*)	90º (*)
arm_7	revolute	-120º	120º

(*) Wrist version with force/torque sensor arm_6 range is [-81º, 81º]

Figura 2.3 Modello cinematico e limiti articolari del robot di servizio TIAGo. Sulla sinistra troviamo la rappresentazione dei sistemi di riferimento sui giunti; a destra, i limiti meccanici di escursione per il torso, la testa e il braccio robotico a sette gradi di libertà [25].

Nella versione di *TIAGo* presente in laboratorio, utilizzata per svolgere la tesi di laurea, il robot è dotato di un *end-effector* di tipo *parallel gripper* (*Figura 2.4*) che contiene due motori, ognuno dei quali controlla una delle due "dita" che lo costituiscono. Ogni "dito" ha una lunghezza di 4 cm, e consente la presa in sicurezza di oggetti tramite apertura e chiusura del *gripper* [25].



Figura 2.4 Parallel gripper in dotazione sul modello di TIAGo utilizzato per il lavoro di tesi [25].

La base mobile di *TIAGo* (*Figura 2.5*) rappresenta il nucleo per la navigazione e l'alimentazione del sistema. È progettata con un diametro di 54 cm e un'altezza di 30 cm e presenta un sistema di trazione differenziale che permette al robot di raggiungere una velocità massima di 1 m al secondo supportando un carico utile fino a 100 chilogrammi. Dal punto di vista dell'autonomia, la base è equipaggiata con una batteria agli ioni di litio da 36 volt e 20 Ah, con la possibilità di installarne una seconda per raddoppiare la capacità energetica (nel prototipo presente in laboratorio abbiamo due batterie), e contiene anche il computer di bordo e i moduli per la connettività wireless. La parte frontale ospita un

telemetro laser mentre la parte posteriore è dotata di tre sensori a ultrasuoni (*sonar*) direzionati per coprire il retro del robot, il pulsante di accensione, l'arresto di emergenza, l'interruttore principale e un piccolo display per monitorare lo stato della batteria. Dietro il sensore laser, abbiamo un accesso diretto alle porte del computer (*USB, HDMI*) utile per attività di debug o installazione. Collocato sotto la copertura superiore abbiamo il pannello di espansione che fornisce numerose interfacce per permettere l'integrazione di sensori aggiuntivi o componenti hardware personalizzati [25].

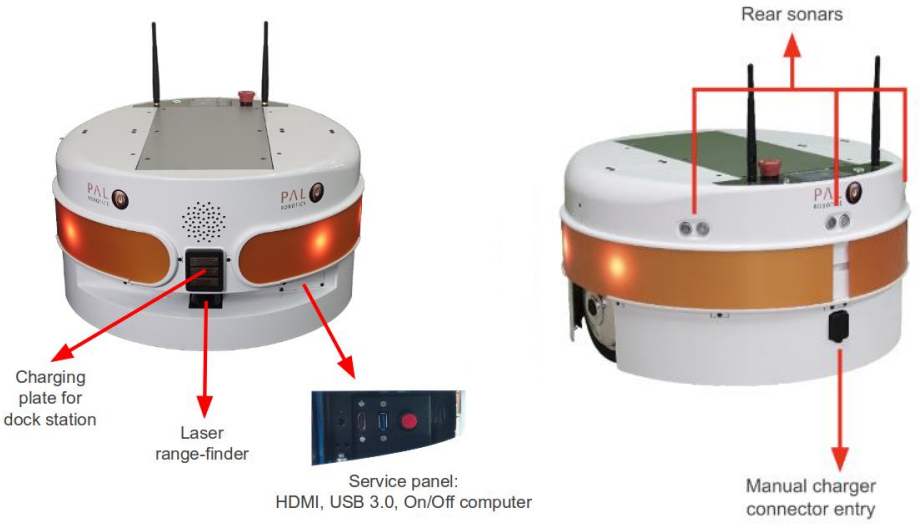


Figura 2.5 Vista frontale (figura di sinistra) e vista posteriore (figura di destra) della base mobile di TIAGo. L'immagine evidenzia la disposizione dei principali sensori di navigazione (telemetro laser e sonar posteriori) e delle interfacce di alimentazione e servizio [25].

2.1.2 Infrastruttura Software: Paradigmi Di Comunicazione Nel Framework ROS 2

Il *robot operating system* 2 (*ROS* 2) è il principale *framework software open source* per lo sviluppo della robotica ed è organizzato in numerosi pacchetti e protocolli di comunicazione al fine di costruire sistemi di controllo personalizzati [26], [27]. Più che un vero e proprio sistema operativo, *ROS* 2 funge da *middleware* creando un livello di astrazione tra l'*hardware* del robot e il codice di alto livello, facilitando lo sviluppo di sistemi complessi, modulari e adatti ad applicazioni in tempo reale. L'architettura *software* di *ROS* 2 si basa su un sistema distribuito, concettualizzabile come una rete o grafo, i cui elementi fondamentali sono chiamati nodi [26]. Un nodo è un'unità di elaborazione indipendente incaricata di svolgere un compito specifico, ad esempio, un nodo potrebbe gestire la lettura di un sensore mentre un altro potrebbe calcolare la traiettoria delle ruote del robot [15]. Per far funzionare l'intero sistema i nodi si scambiano continuamente dati attraverso interfacce di comunicazione standardizzate utilizzando un

modello *publish/subscribe* basato su *topic,* oppure utilizzando un modello *request/reply* basato su *service* [27], [28]. Il coordinamento operativo di *ROS* 2 è affidato agli esecutori, che gestiscono l'ordine e le tempistiche delle attività. Per ottimizzare le prestazioni, il sistema sfrutta la "composizione", che consiste nel raggruppare più nodi all'interno di un singolo processo. Questa tecnica, inoltre, abilita la comunicazione *zero copy* in cui i dati vengono scambiati direttamente via puntatori condivisi, aggirando il *middleware* ed eliminando le latenze di serializzazione. Dal punto di vista implementativo, l'aggregazione avviene dinamicamente a *run-time* stabiliti, tramite i *component containers* (eseguibili di base progettati per ospitare al loro interno più nodi) [26].

Per quanto riguarda la pianificazione del movimento del manipolatore (*motion planning*), lo standard di riferimento in ambito *ROS 2* è il *framework MoveIt*. A differenza della navigazione, che gestisce la locomozione della base mobile sul piano d'appoggio, *MoveIt* si occupa della cinematica del braccio robotico nello spazio tridimensionale. Sfruttando vincoli di posizione, velocità e accelerazione, il *software* calcola traiettorie compatibili con i giunti del braccio e prive di collisioni e le invia ai controller del braccio per eseguire operazioni complesse come il *grasping*. A livello architetturale, queste operazioni sono orchestrate tramite il nodo *MoveGroup*, che fornisce le interfacce per definire le pose obiettivo, calcolare i piani di moto e aggiornare la scena di pianificazione (*Planning Scene*), permettendo ad esempio di inserire gli oggetti presenti sul tavolo di lavoro come ostacoli da evitare. Dal punto di vista della comunicazione, il sistema opera in anello chiuso combinando *topic* e action di *ROS 2*. Per conoscere in tempo reale la configurazione fisica del robot, *MoveIt* si iscrive al *topic /joint_states*, sul quale vengono pubblicati costantemente i *feedback* di posizione e velocità effettiva letti dagli encoder degli attuatori. Una volta calcolata la traiettoria verso la posizione obiettivo (*goal position*), *MoveIt* la trasmette ai controller di basso livello (tramite interfacce d'azione dedicate come *FollowJointTrajectory*), i quali traducono il piano cinematico in comandi per muovere fisicamente i motori del braccio [27].

L'intero processo è supportato graficamente da *RVIZ*, il *software* impiegato per visualizzare i risultati di controllo e pianificare il percorso in tempo reale. Oltre a permettere l'inserimento diretto dei comandi di posizione dalla sua interfaccia, *RVIZ* monitora la sicurezza dell'operazione tramite, ad esempio, visualizzazione degli oggetti di collisione (visualizzabili come *marker*), (*Figura 2.6)*[27].

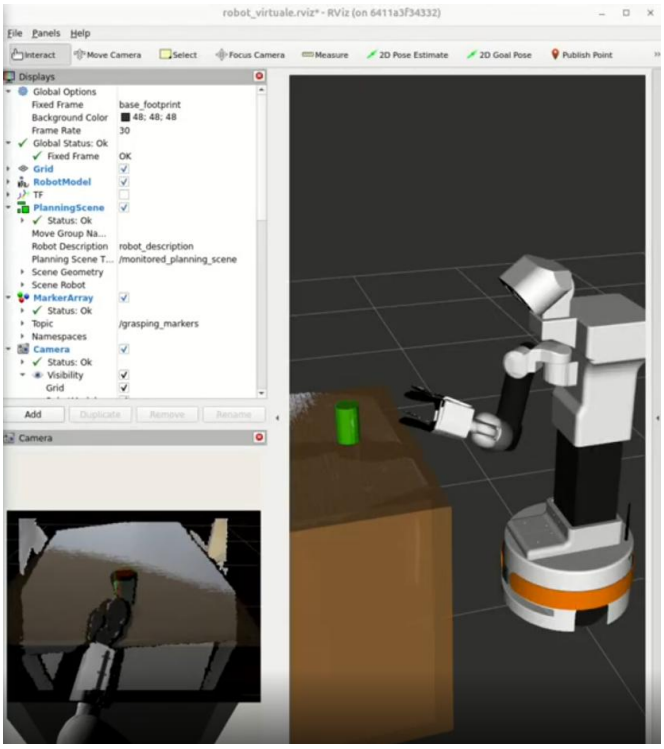


Figura 2.6 Visualizzazione della scena nel software RViz, dove possiamo osservare il modello cinematico del robot TIAGo sulla destra e il braccio manipolatore con il gripper aperto intento ad eseguire un task di pick and place. Notiamo poi la planning scene, tramite un blocco marrone che identifica l'ingombro del piano di lavoro e un cilindro verde sopra il piano che rappresenta il /grasping_markers corrispondente all'oggetto target.

2.2 Percezione Visiva: Il Sensore RGB-D Orbbec Astra

L'avvento di sensori *RGB-D* leggeri e a basso costo basati su luce strutturata ha rappresentato un punto di svolta per la robotica. Queste tecnologie hanno fornito una valida e potente alternativa ai tradizionali e costosi *scanner laser*, permettendo l'implementazione di sistemi *SLAM* (*Simultaneous Localization and Mapping)* robusti. La capacità di generare mappe 3D accurate dell'ambiente in tempo reale risulta infatti un requisito fondamentale per operazioni autonome avanzate, dalla navigazione dei robot mobili fino alla pianificazione di movimenti privi di collisioni (*collision-free motion planning*) per i bracci manipolatori [29]. La ricostruzione tridimensionale (3D) è il processo di acquisizione della forma e dell'aspetto degli oggetti reali, oltre che del volume e della posizione relativa dell'oggetto rispetto ad altri presenti nella scena. Per eseguire una ricostruzione 3D è possibile impiegare diversi metodi e tecnologie, tra questi possiamo citare i sensori *LIDAR*, la tecnologia di imaging a ultrasuoni, i dispositivi di acquisizione *RGB-D* e le immagini multi-vista [30].

2.2.1 Principi Di Acquisizione: Ottica RGB E Infrarossi Strutturati

I moderni sensori *RGB-D (Figura 2.7)* ricavano le informazioni di profondità affidandosi: al calcolo del tempo di volo (*Time of Flight*) della luce emessa dal dispositivo e riflessa dalla scena, oppure, come nel caso oggetto del nostro studio, alla tecnologia a luce strutturata (*structured light*). Il principio di funzionamento della tecnologia a luce strutturata si basa su tre fasi distinte: la prima è la codifica, in cui il sistema genera una sequenza di pattern noti, la seconda è l'acquisizione, in cui l'emettitore proietta il pattern sull'ambiente circostante mentre un sensore ottico dedicato acquisisce in modo continuo le immagini dalla scena illuminata, e la terza e ultima è la decodifica, in cui il processore interno elabora le immagini riconoscendo le distorsioni geometriche subite dal pattern a causa dei volumi degli oggetti fisici, individuando cosi i punti di corrispondenza esatti tra ciò che è stato proiettato e ciò che è stato acquisito [31].



Figura 2.7 Telecamera Orbbec Astra in configurazione stand alone [32].

Per tradurre la fase di decodifica in coordinate spaziali tridimensionali, il sistema si basa sul principio della triangolazione dei raggi che richiede una rigorosa calibrazione del dispositivo per determinare i parametri intrinseci delle lenti e i parametri estrinseci (ovvero la distanza fisica e l'angolazione tra il proiettore a infrarossi e la telecamera), tramite il punto di intersezione dei raggi luminosi riflessi il software ricava il valore di profondità associato a ciascun *pixel* [33].

Di fatti, un'immagine *RGB-D* è un'immagine a colori (*RGB)* combinata alla sua immagine di profondità, chiamata anche mappa di profondità (*depth map*), in cui il canale aggiuntivo riporta la distanza tra il piano dell'immagine e il corrispondente punto nell'immagine *RGB* [30], [31], [33]. La mappa di profondità, perciò, per ogni *pixel* contiene un valore di distanza, che aiuta il computer a interpretare la forma, la *texture* e ad eseguire la ricostruzione 3D della scena [33], [29]. Un elemento fondamentale della ricostruzione 3D è la nuvola di punti (*point cloud*) che rappresenta l'insieme di coordinate tridimensionali che definiscono la superficie esterna di un oggetto. Queste nuvole di punti possono essere ottenute in diversi modi, il più comune dei quali è tramite *scanner laser*. Le nuvole di punti ci aiutano a ricostruire l'oggetto poiché ogni punto della nuvola è composto dalle coordinate X, Y, e Z che definiscono rispettivamente larghezza, altezza e

profondità rispetto ad un'origine (0,0,0) situata esattamente nel centro ottico del sensore *RGB* o del sensore di profondità, a seconda della calibrazione specifica [33].

2.2.2 Flusso Dati E Gestione Dei Topic Visivi In ROS 2

All'interno dell'ecosistema *ROS 2*, l'integrazione del sensore *RGB-D* a bordo del robot *TIAGo* richiede una gestione rigorosa del flusso dati, al fine di convertire i segnali *hardware* grezzi in strutture dati standardizzate fruibili dagli algoritmi di percezione e manipolazione. Come riportato nella documentazione tecnica ufficiale fornita dal sito *PAL Robotics*, il sistema operativo del robot organizza i flussi visivi in specifici *topic*, raggruppando i dati del blocco ottico principale all'interno del *topic:* /head_front_camera [25].

Quando la telecamera è attiva, il relativo nodo *ROS* si occupa di pubblicare i dati sfruttando le interfacce standard di libreria. Nello specifico, per quanto concerne il flusso visivo a colori (*RGB*), il sistema espone due canali di comunicazione primari:

1 */head_front_camera/color/image_raw/**: Questo *topic* veicola la matrice di pixel *RGB* in formato non compresso, impiegando il tipo di messaggio *sensor_msgs/msg/Image*. Inoltre, la presenza dell'asterisco finale indica che il *driver* sfrutta il *middleware image_transport* di *ROS 2* per esporre contestualmente dei sotto-topic compressi (ad esempio */head_front_camera/color/image_raw/compressed*); questo meccanismo codifica i *frame* in formato *JPEG* o *PNG*, permettendo la trasmissione del flusso video al computer di controllo senza saturare la larghezza di banda della rete locale.

2 */head_front_camera/color/camera_info*: Questo *topic* trasmette i metadati essenziali della telecamera tramite il messaggio *sensor_msgs/msg/CameraInfo*. Al suo interno sono contenute le matrici geometriche di calibrazione: i parametri intrinseci (lunghezza focale e centro ottico), la matrice di proiezione e i coefficienti di distorsione. Tali informazioni sono imprescindibili per il controllo del braccio robotico, poiché permettono di tradurre le coordinate 2D dell'immagine in misurazioni metriche spaziali [25].

Parallelamente al flusso a colori, l'architettura prevede la gestione dei dati di profondità per ricavare la terza dimensione dello spazio operativo. Sia il sensore ottico a colori (*RGB*) sia il sensore di profondità acquisiscono i rispettivi flussi di dati a una risoluzione di 640x480 pixel, operando a una frequenza di aggiornamento costante di 30 fotogrammi al secondo (*fps*). Il sensore a infrarossi emette la propria lettura metrica sul *topic*: */head_front_camera/depth/image_rect_raw* (veicolando messaggi di tipo *sensor_msgs/msg/Image*), e il valore di ogni singolo *pixel* non esprime un'intensità cromatica, ma la distanza ortogonale (tipicamente misurata in millimetri) della scena

rispetto al piano ottico della telecamera. Anche in questo caso, i parametri intrinseci necessari per la correzione della lente a infrarossi vengono esposti su un *topic* dedicato: */head_front_camera/depth/camera_info*. Una delle problematiche principali nella gestione combinata dei dati *RGB-D* risiede nel disallineamento spaziale dell'obiettivo a colori e del sensore di profondità, che sono posizionati a una determinata distanza l'uno dall'altro. Per risolvere questo *offset* prospettico, il *driver ROS 2* del robot *TIAGo* implementa un processo di registrazione spaziale (*depth-color registration*). Sfruttando le matrici di trasformazione estrinseca fornite dai messaggi *CameraInfo*, il sistema riproietta la mappa di profondità nel sistema di riferimento della telecamera a colori. Il risultato di questa complessa operazione matematica viene pubblicato sul topic */head_front_camera/aligned_depth_to_color/image_raw*.

Infine, l'unione dei dati visivi e di profondità registrati confluisce nella generazione del *topic /head_front_camera/depth_registered/points*. Tale canale pubblica messaggi di tipo *sensor_msgs/msg/PointCloud2*, ovvero dense nuvole di punti tridimensionali colorate. Questa struttura dati rappresenta l'apice del flusso di elaborazione visiva, poiché costituisce l'input diretto per i *framework* di pianificazione del movimento (come *MoveIt*), consentendo al manipolatore robotico di eseguire la ricostruzione degli ostacoli, la stima della posa e la sintesi sicura delle prese (*grasping*) [25].

Capitolo Tre. Sviluppo Della Pipeline Visiva E Scelta Dell'Architettura Neurale YOLO

Durante la prima fase del progetto di tesi mi sono concentrata sullo sviluppo e validazione dell'algoritmo di visione ottimale per il mio lavoro di tesi utilizzando e testando il codice *Phyton* scritto a mano, tramite telecamera *RGB-D Orbbec Astra* nella configurazione *stand-alone* disponibile nel laboratorio. L'obiettivo primario del lavoro è consistito nell'analisi comparativa di due distinte metodologie di ricostruzione 3D. Inizialmente è stato valutato un approccio *depth-based*, fondato sul dato di profondità come input primario. Successivamente, l'analisi si è concentrata su un metodo di ricostruzione volumetrica puramente *RGB-based*, capace di generare il modello tridimensionale a partire da feature geometriche 2D estratte dalle immagini, prescindendo totalmente dalle mappe di profondità iniziali.Sono state eseguite delle acquisizioni iniziali molto semplici per testare i due codici andando a confrontare la ricostruzione da un punto di vista qualitativo, mentre in un secondo momento una volta scelto il codice migliore sono state eseguite delle valutazioni quantitative. Gli oggetti valutati sono oggetti di uso comune, tra cui bicchieri di plastica classificati dalla rete *YOLO* come *cup*, piccole palline da gioco classificate come *sportsball* e alcuni oggetti di forma assimilabile ad un parallelepipedo, tra cui per esempio un telecomando, in cui la classe *YOLO* è *remote* [34]. Dopo aver scelto il codice di ricostruzione sono stati confrontati diversi modelli *YOLO* per determinare quale sia il più opportuno per il caso specifico analizzato nel lavoro di tesi e dunque per una ricostruzione 3D ottimale al fine di eseguire esercizi di *pick and place* tramite il robot *TIAGo*.

3.1 Rete Neurale Per La Visione: Il Modello YOLO Per Object Detection E Segmentazione

Il rilevamento degli oggetti (*object detection*) è un campo importante nel dominio della visione artificiale (*computer vision*), migliorato e ottimizzato tramite processi di apprendimento automatico (*machine learning*) e profondo (*deep learning*). Nello specifico, questi algoritmi supervisionati di *machine learning* o *deep learning*, in cui il modello viene addestrato su un ampio *dataset* etichettato, ci permettono di classificare le immagini o gli oggetti presenti in esse in categorie predefinite [35], [34].

3.1.1 Object Detection

Il rilevamento degli oggetti può essere considerato come una combinazione di classificazione, localizzazione e segmentazione (*Figura 3.1*) [35]. Data un'immagine lo scopo *dell'object detection* è quello di determinare se vi siano o meno istanze di oggetti appartenenti a categorie predefinite e, se presenti, restituire la posizione spaziale e l'estensione di ciascuna istanza [36].

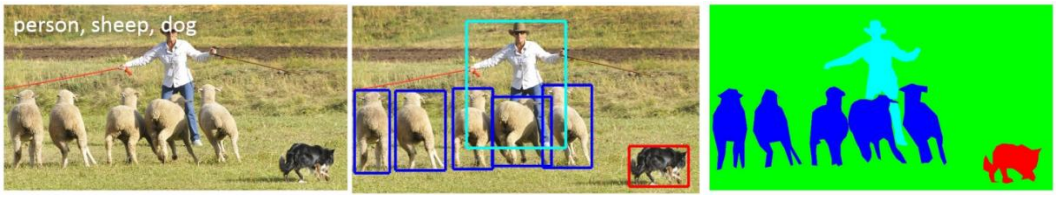


Figura 3.1 Object detection. Nell'ordine, da sinistra verso destra, osserviamo: classificazione, localizzazione (tramite bounding box) e segmentazione della stessa immagine di riferimento [34].

La localizzazione degli oggetti (*object localization*) è il compito di determinare la posizione di un singolo o di molteplici oggetti in un'immagine o fotogramma con l'aiuto di un riquadro rettangolare localizzato attorno all'oggetto, comunemente noto con il nome di *bounding box* [35]. Tramite il *bounding box* si ottiene una delimitazione rettangolare che racchiude l'oggetto in modo approssimativo. In alternativa, per un'identificazione più accurata, è possibile estrarre una precisa maschera di segmentazione in grado di ricalcare perfettamente i contorni dell'oggetto (si ottiene un *bounding box* perfettamente aderente ai contorni dell'oggetto, unendo in pratica localizzazione e segmentazione) [36]. La segmentazione delle immagini (*image segmentation*), infine è il processo di partizionamento di un'immagine in segmenti multipli, in cui un segmento può contenere un oggetto completo o una parte di esso o, ancora, essere parte dello sfondo. I pixel all'interno di un segmento possiedono un insieme di caratteristiche comuni, come intensità e texture [35].

3.1.2 YOLO: You Only Look Once

YOLO (*You Only Look Once*) è un rilevatore che produce in output il *bounding box* e le probabilità specifiche delle classi per gli oggetti sottostanti esaminando tutte le dimensioni spaziali di un'immagine in un solo colpo. L'algoritmo *YOLO* basa il suo processo di rilevamento sulle reti neurali convoluzionali (*CNN*), considerate in letteratura come i modelli più rappresentativi del *deep learning*. Questa architettura consente al sistema di apprendere le caratteristiche delle immagini in modo diretto e automatico sfruttando una struttura gerarchica: i livelli iniziali della rete estraggono le feature di basso livello (come i bordi degli oggetti), mentre i livelli successivi elaborano rappresentazioni di complessità crescente per ottenere astrazioni sempre più profonde [35].

3.1.3 Dataset: MicroSoft Common Objects in COntext (MSCOCO)

Il dataset *MicroSoft Common Objects in COntext (MSCOCO)* è uno dei dataset più popolari nei compiti di visione artificiale, contenente scene complesse della vita quotidiana con molteplici oggetti in esse, collocati nel loro contesto naturale [35], [34], [36]. Il dataset presenta un numero limitato di categorie ma comprende più istanze (abbiamo diversi esempi per ogni classe) in ciascuna categoria: nello specifico include 91

diverse categorie di oggetti come ad esempio: persona, cane, treno e altri oggetti comunemente presenti in una scena di vita quotidiana. Oltre all'elevato numero di istanze in ciascuna categoria, il dataset presenta anche molteplici istanze con caratteristiche diverse per ogni singola immagine, perciò in una singola immagine composta per esempio da quattro persone, due automobili e un semaforo queste diverse istanze possono avere caratteristiche diverse all'interno della stessa fotografia perciò alcune possono essere in parte occluse da altri oggetti, alcune piccole e sfocate perché si trovano vicino allo sfondo, altre in primo piano e magari con angolazioni e illuminazioni differenti, etc. [35]. In particolare, per il lavoro di tesi è stato utilizzato un modello di rete neurale *YOLO* pre-allenato sul dataset MS*COCO*, e sono stati selezionati nello specifico alcuni oggetti (dunque alcune classi di MS*COCO*) tra cui: *cup*, *bottle*, *remote*, sports ball, e altre.

3.2 Ricostruzione 3D Con Approccio Basato Su Nuvola Di Punti

Il primo approccio esplorato per la ricostruzione 3D si è basato su un'estrazione diretta delle coordinate spaziali della mappa di profondità (*depth map*) [31] , [37]. In questa architettura iniziale le maschere di segmentazione bidimensionale generate dalla rete *YOLO* venivano sovrapposte direttamente al flusso infrarossi della telecamera in modo da avere una separazione tra oggetto e sfondo. I pixel di profondità isolati venivano poi proiettati nello spazio per generare una nuvola di punti locale sulla quale venivano calcolate le metriche geometriche dell'oggetto, dunque calcolate nel tridimensionale. L'elemento caratterizzante di questa implementazione risiedeva proprio nella sua dipendenza quasi esclusiva dai valori di profondità grezzi per la generazione del modello tridimensionale. Viene delineata di seguito la pipeline dettagliata con i vari passaggi approfonditi.

3.2.1 Workflow Della Ricostruzione Spaziale Basata Su Nuvola Di Punti

La pipeline sviluppata per questo primo approccio di ricostruzione si articola in una sequenza di fasi ben definite, progettate per trasformare dati grezzi provenienti dai sensori *hardware* in metriche tridimensionali strutturate. Il processo di elaborazione opera in tempo reale (*loop* continuo *frame per frame*) e si divide nei seguenti passaggi fondamentali:

- *Acquisizione E Registrazione Hardware*: Il flusso inizia con l'acquisizione simultanea dei flussi video *RGB* e di profondità tramite il sensore *Orbbec Astra*, con attivazione della registrazione *hardware* spaziale. Poiché la lente a colori e il proiettore a infrarossi sono fisicamente distanti, il driver della telecamera viene configurato per deformare e allineare prospetticamente la matrice di profondità, garantendo una corrispondenza esatta, *pixel per pixel*, con il fotogramma *RGB* [37].

- *Percezione E Segmentazione Semantica 2D*: Il fotogramma a colori viene passato in input al modello *YOLO* (architettura scelta arbitrariamente per una prima valutazione qualitativa: *versione 8 segmentation nano*), il quale identifica gli oggetti nella scena associandoli a delle classi in relazione al dataset *MSCOCO* [36]. Nel codice viene inoltre implementata una divisione degli oggetti in tre classi predefinite (sfera, cilindro e parallelepipedo), in modo tale che ogni oggetto venga collocato in una classe in relazione alle sue caratteristiche geometriche per poi essere ricostruito in modo opportuno a seconda della sua morfologia, attraverso delle funzioni specifiche implementate nel codice. Il modello restituisce le coordinate matematiche di un *bounding box* aderente ai contorni dell'oggetto. Allo scopo di separare l'oggetto dallo sfondo per isolare la nuvola di punti, viene generata una maschera di segmentazione binaria ad alta definizione che permette di isolare i pixel esatti che compongono il target escludendo completamente lo sfondo (valore 255 corrispondente all'oggetto, valore 0 corrispondente allo sfondo).
- *Proiezione Spaziale E Filtraggio*: La maschera bidimensionale ottenuta viene sovrapposta alla mappa di profondità allineata. In questo modo il sistema estrae esclusivamente i valori di distanza pertinenti all'oggetto. Utilizzando le equazioni del modello *Pinhole Camera,*

$$X = \frac{(u-c_x)*Z}{f_x} \qquad Y = \frac{(v-c_y)*Z}{f_y}$$

  i parametri intrinseci della lente (lunghezze focali, f_x ed f_y , e centro ottico di coordinate c_x e c_y) opportunamente relazionati con le coordinate immagine in pixel (*u,v*), e il valore di profondità Z, possiamo ricavare le coordinate metriche spaziali (*X,Y*). Al fine di mitigare gli errori di lettura del sensore i punti generati vengono sottoposti ad un filtraggio statistico, che elimina i valori anomali (*outlier*) e isola i punti attorno al valore di profondità mediano.

- *Analisi Geometrica E Stima Dimensionale*: Le coordinate spaziali estratte compongono una nuvola di punti locale sulla quale il sistema applica algoritmi geometrici differenti in base alla classe dell'oggetto (sfera, cilindro o parallelepipedo). Per oggetti sferici viene calcolato il cerchio minimo circoscritto alla maschera bidimensionale, convertendo il raggio da pixel a millimetri, sfruttando la distanza media e dunque, ottenendo il diametro reale e ricavando il baricentro dell'oggetto. Per quanto riguarda oggetti classificati come cilindro o parallelepipedo, viene eseguita un'analisi delle componenti principali (*PCA*) tramite il calcolo della matrice di covarianza. Estraendo autovalori e autovettori il sistema individua l'orientamento spaziale reale dell'oggetto, e proiettando i punti

della nuvola su questi assi principali è possibile misurare con precisione altezza, larghezza e profondità reali e calcolare il baricentro rispetto alla normale dell'oggetto. Il baricentro viene poi shiftato di una quantità pari a mezzo spessore o al raggio della figura, per poterlo posizionare correttamente all'interno dell'oggetto, ottenendo così il baricentro reale dell'oggetto 3D.

- *Fitting Visivo E Registrazione Delle Metriche*: Nell'ultima fase, i dati matematici vengono passati ad un motore di *rendering 3D* (*Figura 3.2*).



Figura 3.2 Esempio di ricostruzione 3D basata sulla mappa di profondità: l'immagine evidenzia la presenza di pixel fluttuanti (flying pixels) lungo i margini dell'oggetto e l'instabilità dinamica nel calcolo del baricentro, fenomeno visibile come continue fluttuazioni nei video di ricostruzione.

Parallelamente, tutte le metriche dimensionali, le coordinate del baricentro, le oscillazioni delle misure (differenze dei valori ottenuti tra un frame e l'altro) e le frequenze operative vengono registrate e salvate in un file *CSV* per l'analisi offline delle performance.

3.2.2 Risultati Sperimentali: Instabilità E Limiti Fisici Del Sensore

Durante la fase di test e validazione condotte su diverse sequenze video, l'approccio basato sull'estrazione della nuvola di punti ha evidenziato criticità sostanziali. L'analisi dei dati sperimentali ha fatto emergere un problema sistematico legato ai limiti fisici del sensore di profondità. La tecnologia ad infrarossi alla base della telecamera, infatti, risulta fortemente sensibile alle proprietà dei materiali inquadrati: superfici caratterizzate da particolari indici di assorbimento ottico e superfici curve impediscono una corretta lettura della distanza. Questo limite hardware si è tradotto nella presenza di ampie zone con valori di profondità nulli, veri e propri "*buchi*" (dati mancanti) all'interno della mappa spaziale dell'oggetto. Al fine di valutare quantitativamente l'affidabilità del sensore di profondità, è stata sviluppata un'analisi visiva degli errori di acquisizione. Confrontando la maschera di segmentazione ideale generata dalla rete *YOLO* (che isola fedelmente la sagoma 2D dell'oggetto) con i dati grezzi del sensore, è stata generata un'apposita immagine di diagnostica (*debug missing*), (*Figura 3.3*). In tale rappresentazione, i pixel assenti dalla zona colorata di bianco (zone dove ci si aspetta bianco ma si vede nero)

evidenziano in modo diretto i 'buchi' informativi: si tratta di zone in cui l'algoritmo rileva correttamente la presenza dell'oggetto, ma per le quali il sensore a infrarossi non restituisce alcuna misurazione (profondità nulla) a causa della deflessione del raggio su superfici curve o a determinati indici di assorbimento.

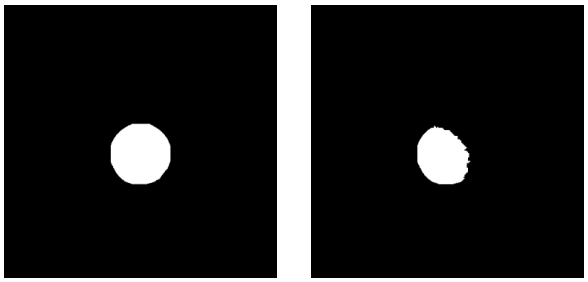


Figura 3.3 Confronto tra la segmentazione 2D ideale (a sinistra) e l'immagine di debug (a destra), dove i pixel bianchi mancanti, che rendono la figura incompleta, evidenziano la perdita di segnale di profondità sulla superficie curva dell'oggetto.

La carenza di questi dati corrompeva irrimediabilmente la struttura della nuvola di punti generata (*Figura 3.4*) e di conseguenza, gli algoritmi di analisi delle componenti principali ed il successivo *fitting* delle primitive 3D risultavano fortemente instabili.



Figura 3.4 Esempio di ricostruzione di un oggetto cilindrico per focus sul problema dei dati mancanti nella ricostruzione basata su depth.

L'analisi visiva della ricostruzione tridimensionale in *Figura 3.4* evidenzia i limiti intrinseci dell'acquisizione tramite sensori a infrarossi, distinguendo chiaramente tra le occlusioni geometriche e la perdita di dati locali. Come si osserva nei risultati, la telecamera *RGB-D* cattura esclusivamente la superficie anteriore rivolta verso l'ottica, lasciando non mappato il volume posteriore. Inoltre, in corrispondenza dei profili curvi dell'oggetto, i raggi infrarossi subiscono fenomeni di deflessione anziché tornare al ricevitore, generando discontinuità e 'buchi' di profondità nei punti critici.

La ricostruzione visiva presentava, dunque, artefatti evidenti e sfarfallii continui, portando il sistema a restituire stime dimensionali errate e un calcolo del baricentro del tutto inaffidabile che variava continuamente locazione tra i vari *frame*. Nel tentativo di mitigare questo difetto, studiando la letteratura e lo stato dell'arte della ricostruzione tridimensionale è stato integrato nella pipeline un modulo di ricostruzione digitale

dell'immagine (*inpainting*) [38], [39], [40]. Nello specifico, è stato applicato un algoritmo basato sulle equazioni della fluidodinamica di *Navier Stokes*, con l'obiettivo di interpolare fluidamente i valori di profondità mancanti partendo dai pixel validi adiacenti [39], [40].

L'area che presentava i "buchi" veniva trattata come un fluido incomprimibile utilizzando l'algoritmo *cv2.INPAINT_NS*, che sfrutta una finestra locale di vicinato (5x5) e impone le informazioni di profondità dei pixel affidabili adiacenti come condizione al contorno. Tramite la propagazione del *Laplaciano* lungo le isofote (linee di ugual valore), l'equazione diffonde la pendenza superficiale all'interno del punto "vuoto". Nelle zone piane la viscosità del "fluido" è bassa per accelerare il riempimento, mentre in prossimità dei bordi geometrici a forte gradiente la viscosità aumenta non linearmente, arrestando la diffusione e preservando intatti i contorni netti del solido [40]. Tuttavia, i risultati sperimentali hanno dimostrato che tale correzione *software* non apportava miglioramenti significativi: l'interpolazione riempiva i *pixel* vuoti in modo matematicamente continuo, ma i valori generati non rispettavano la reale curvatura e geometria tridimensionale dell'oggetto, mantenendo le misurazioni finali al di fuori del margine di tolleranza richiesto per la manipolazione robotica.

A fronte di questi risultati è emersa la necessità di un cambiamento di paradigma e si è scelto di scartare questa prima *pipeline* e di abbandonare l'approccio puramente tridimensionale basato sulla mappa di profondità come punto di partenza per l'estrazione morfologica.

3.3 Approccio Basato Su Ricostruzione Volumetrica

Per superare i limiti strutturali del sensore a infrarossi e l'instabilità delle nuvole di punti riscontrate nella pipeline basata sul dato grezzo della profondità, è stato sviluppato un nuovo approccio metodologico di tipo ibrido, basato su una ricostruzione volumetrica. Invece di effettuare un fitting geometrico su dati 3D spesso mancanti e dunque inaffidabili, questo nuovo metodo inverte il paradigma di calcolo affidando l'intera analisi morfologica (riconoscimento della forma, calcolo di larghezza e altezza, e stima del baricentro) allo spazio bidimensionale (*RGB* ad alta risoluzione). La mappa di profondità viene interrogata solo in modo mirato durante la stima del baricentro, in modo da ottenere dei valori affidabili. Questa strategia svincola il calcolo dimensionale dalla qualità specifica del sensore a infrarossi, garantendo un'elevata stabilità e precisione per le successive operazioni di manipolazione robotica.

3.3.1 Elaborazione RGB E Segmentazione Semantica Bidimensionale

Il primo stadio di questa nuova architettura richiama la precedente, andando a elaborare come primo passo il flusso a colori *RGB*, e mantenendo anche in questo caso la registrazione *hardware* spaziale. Successivamente, il fotogramma *RGB* viene processato

dalla rete neurale convoluzionale *YOLO* (architettura *v8 segmentation nano*), che anche in questo caso, classifica nelle tre categorie scelte gli oggetti (sfera, cilindro o parallelepipedo) e ne estrae i contorni esatti. Viene generata nuovamente la maschera binaria per separare oggetto e sfondo, che in questo caso però non funge solo da filtro per la pulizia della nuvola di punti globale come nel codice precedente, ma costituisce il vero e proprio nucleo matematico per il successivo calcolo delle dimensioni dell'oggetto e del suo baricentro.

3.3.2 Calcolo Dimensionale 2D Invariante All'orientamento

Il sistema calcola il centroide bidimensionale dell'oggetto, ricavato come media delle coordinate dei pixel appartenenti alla maschera bidimensionale, non interrogando la nuvola di punti completa, ma estraendo una piccola *ROI* (*region of interest*) di dimensione *5x5 pixel* attorno al centroide bidimensionale direttamente dalla mappa di profondità. Scartando eventuali valori nulli o fuori scala, il sistema calcola la mediana delle letture sulla mappa di profondità all'interno di questa *ROI*, fornendo una quota Z (in millimetri) estremamente robusta e rappresentativa della superficie frontale dell'oggetto. Ottenuta la distanza Z, il sistema procede al calcolo degli ingombri geometrici analizzando esclusivamente la distribuzione dei *pixel* nella maschera binaria.

Per garantire che le misurazioni siano invarianti rispetto alla rotazione dell'oggetto nell'immagine, vengono applicati due metodi differenziati in base alla morfologia dell'oggetto in esame:

- Per gli oggetti sferici viene applicato un algoritmo di calcolo del cerchio minimo circoscritto (*minimum enclosing circle*) al contorno della maschera, che ci permette di ottenere un raggio in *pixel* indipendente da qualsiasi orientamento.
- Per cilindri e parallelepipedi i *pixel* della maschera vengono trattati con l'analisi delle componenti principali (*PCA*), calcolando la matrice di covarianza e i relativi autovalori e autovettori. Il sistema in questo modo individua automaticamente l'asse maggiore e l'asse minore della distribuzione dei *pixel* (corrispondenti rispettivamente all'altezza/lunghezza e alla larghezza dell'oggetto). L'estensione lungo questi assi restituisce gli ingombri esatti in *pixel* indipendenti da come l'oggetto sia inclinato e posizionato sul piano di lavoro.

Infine, le dimensioni ricavate in *pixel* (raggio, altezza e larghezza) vengono convertite nelle reali dimensioni metriche (in millimetri) attraverso le equazioni del modello *Pinhole Camera,* descritte nei paragrafi precedenti. Poiché la quota di profondità del baricentro Z campionata appartiene alla superficie visibile (frontale) dell'oggetto, la coordinata di profondità viene traslata verso l'interno di una quantità pari al mezzo spessore calcolato (raggio o mezza larghezza), per ricavare il baricentro tridimensionale. Questo metodo

garantisce che il *Tool Center Point* stimato si trovi esattamente al centro del volume teorico dell'oggetto e non sulla superficie frontale. In particolare, questo ci permette di individuare correttamente il baricentro all'interno di oggetti che presentano simmetria radiale, come sfere o cilindri, mentre con oggetti parallelepipedi si tratta di una approssimazione.

3.3.3 Retroproiezione Geometrica E Filtraggio RANSAC

Il sistema successivamente, procede all'estrazione della superficie tridimensionale visibile dell'oggetto andando ad effettuare una retroproiezione geometrica che consente di mappare i *pixel* dell'immagine 2D nel reale spazio metrico 3D. Per ogni *pixel* di coordinate (u,v) identificato dalla maschera come appartenente all'oggetto, viene campionato il corrispondente valore di distanza dalla mappa di profondità. Il sistema scarta i valori nulli o fisicamente implausibili. Successivamente, sfruttando il modello matematico della *Pinhole Camera* e i parametri intrinseci del sensore (le lunghezze focali e il centro ottico), i punti bidimensionali vengono proiettati nello spazio tridimensionale.

L'insieme delle coordinate spaziali (X,Y,Z) e dei rispettivi valori cromatici campionati dal fotogramma visivo va a costituire una nuvola di punti grezza, soggetta a rumore e artefatti. Al fine di isolare l'oggetto target, la nuvola di punti viene prima analizzata tramite l'algoritmo probabilistico *RANSAC* (*Random Sample Consensus*) integrato nella libreria *Open3D* [41], [42], [43]. L'algoritmo modella matematicamente il piano dominante della scena, corrispondente alla superficie del tavolo d'appoggio, estraendone i coefficienti cartesiani per definire l'equazione generale del piano:

$$Ax + By + Cz + D = 0$$

Questa equazione viene impiegata come filtro geometrico di *background* calcolando la distanza ortogonale punto-piano di ciascun vettore della *point cloud* attraverso la relazione:

$$d = \frac{|A_{X_0} + B_{Y_0} + C_{Z_0} + D|}{\sqrt{A^2 + B^2 + C^2}}$$

tutti i punti appartenenti o adiacenti al tavolo (caratterizzati da una distanza d inferiore a una soglia prestabilita) vengono eliminati dall'oggetto. La nuvola residua viene poi rifinita utilizzando un filtro statistico spaziale. Il filtro va ad analizzare l'intorno locale di ciascun punto della nuvola (valutando i 40 vicini più prossimi) e calcola la distanza media da essi. I punti in cui la distanza supera una determinata soglia di tolleranza, basata sulla deviazione standard globale della distribuzione, vengono classificati come rumore ed eliminati, permettendo di ottenere una superfice tridimensionale pulita, coerente e priva di artefatti.

3.3.4 Rendering Geometrico E Registrazione Dei Dati

L'ultima fase della *pipeline* è dedicata alla visualizzazione tridimensionale in tempo reale e alla raccolta delle metriche calcolate in un file *CSV* dedicato. Sfruttando le dimensioni reali (diametro, altezza e larghezza) e le coordinate spaziali del baricentro ricavate nei passaggi precedenti, l'algoritmo istanzia dinamicamente una primitiva geometrica ideale (sfera, cilindro o parallelepipedo) all'interno dell'ambiente virtuale generato dalla libreria *Open3D*. Per conferire un aspetto realistico a questo volume teorico, il sistema esegue un'avanzata operazione di mappatura cromatica (*color mapping*) sfruttando la nuvola di punti reale filtrata precedentemente. Nello specifico, la *mesh* della primitiva viene inizialmente suddivisa per aumentarne la risoluzione superficiale (*funzione subdivide_midpoint*). Successivamente, implementando un algoritmo di ricerca spaziale ad alta efficienza basato su *KD-Tree* (*KDTreeFlann*), il sistema interroga la nuvola di punti: per ciascun vertice del modello teorico viene individuato il punto reale più vicino nello spazio 3D, trasferendone il valore cromatico (*RGB*). Questa soluzione permette di ottenere un solido matematicamente perfetto nelle dimensioni, ma visivamente simile all'oggetto reale inquadrato dalla telecamera. All'interno della scena 3D (*Figura 3.5*), in corrispondenza del baricentro volumetrico calcolato, viene sovrapposto un sistema di assi cartesiani, che rappresenta un riferimento visivo di cruciale importanza: indica l'esatto *Tool Center Point* (*TCP*) stimato, ovvero il punto *target* che verrà comunicato all'eventuale braccio robotico per le operazioni di presa (*grasping*).

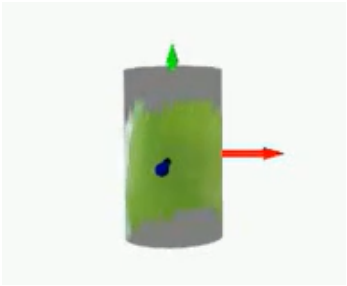


Figura 3.5 Risultato del fitting geometrico basato sul modello volumetrico.

Rispetto all'approccio basato unicamente sulla *depth*, l'approssimazione geometrica offerta dal metodo di ricostruzione basato su volume garantisce una stima delle dimensioni più corretta, una superficie pulita e una stabilità dinamica del baricentro completamente priva di fluttuazioni (fenomeno osservabile nei video in *output*). Parallelamente al *rendering* visivo, l'algoritmo effettua un tracciamento continuo di tutte le grandezze operative. Ad ogni fotogramma, le coordinate del baricentro, le estensioni dimensionali e i tempi di esecuzione (espressi in Hz) vengono immagazzinati in memoria. Al termine della misurazione (o quando l'oggetto esce dal campo visivo), il sistema elabora un'analisi statistica della sessione: calcola le dimensioni medie, le oscillazioni

massime degli ingombri e, soprattutto, il *jitter* spaziale del baricentro (la variazione millimetrica massima lungo gli assi X, Y e Z), che fornisce un indice diretto della stabilità della misurazione tridimensionale. Tutti i record, suddivisi sia in formati *frame per frame* che in riassunti statistici, vengono esportati automaticamente in file *CSV*, corredati dalle registrazioni video (*RGB* e ambiente 3D), permettendo così una rigorosa validazione offline delle performance del sistema.

3.4 Metriche Di Valutazione Delle Performance E Scelta Della Pipeline Di Ricostruzione

Al fine di determinare l'architettura definitiva per il sistema di visione, è stata condotta una campagna di test comparativa tra la prima pipeline realizzata (modello *depth*) e la successiva architettura ibrida (modello *volume*). La validazione si è basata sul confronto tra le misurazioni estratte in tempo reale dall'algoritmo (i cui dati completi sono riportati nella tabella sottostante) e un *ground truth* fisico precedentemente misurato in laboratorio. Il set di validazione ha incluso oggetti sferici (palline di vario diametro), cilindrici/conici (bicchieri di plastica colorati e in legno) e parallelepipedi (mouse e telecomando).

Oggetto target	*Ground Truth*		*Depth*			*Volume*		
	L/D (cm)	***H (cm)***	***L/D (cm)***	***H (cm)***	***Baricentro (cm)***	***L/D (cm)***	***H (cm)***	***Baricentro (cm)***
Cup giallo	7	12	3.16	14.29	X = - 10.64 Y = 8.69 Z = 96.57	6.84	11.86	X = - 10.1 Y = 9.39 Z = 97.40
Cup rosso	7	12	3.23	13.92	X = - 10.83 Y = 8.52 Z = 96.60	6.99	11.87	X = - 10.21 Y = 9.22 Z = 97.46
Cup verde	7	12	3.08	14.48	X = - 10.13 Y = 8.80 Z = 96.07	6.95	11.88	X = - 9.66 Y = 9.49 Z = 96.98
Cup viola	7	12	3.15	14.11	X = - 9.79 Y = 8.90 Z = 96.06	6.93	11.93	X = - 9.31 Y = 9.58 Z = 96.67
Cup legno	7	12	4.17	12.33	X = - 10.23 Y = 10.46 Z = 98.08	6.93	11.02	X = - 8.98 Y = 10.76 Z = 98.76
Remote	5.5	12	9.58	11.29	X = - 0.83 Y = 22.88 Z = 85.10	5.17	9.19	X = 0.41 Y = 20.35 Z = 86.10
Mouse	6	12	6.89	11.09	X = - 1.51 Y = 14.00 Z = 98.88	5.32	6.38	X = 0.42 Y = 13.06 Z = 100.24
Sports ball bianca	5.5	0	5.53	0	X = - 10.15 Y = 13.11 Z = 97.42	5.67	0	X = - 10.05 Y = 13.15 Z = 100.03
Sports ball bianca e rossa	7.5	0	7.26	0	X = - 9.02 Y = 18.57 Z = 86.39	7.29	0	X = - 8.74 Y = 17.94 Z = 87.05

Tabella 3.1 Valutazione comparativa delle pipeline di visione (Depth vs Volume): stime dimensionali, ground truth delle dimensioni e stima del baricentro. Dove x corrisponde all'asse orizzontale, y all'asse verticale e z identifica la profondità, quindi la distanza oggetto-telecamera. Il sistema di riferimento risiede nel centro dell'obiettivo della Orbbec Astra.

Prendendo in esame la categoria dei cilindri (come i cup colorati in plastica, caratterizzati da un *ground truth* di 7.0 cm di diametro e 12.0 cm di altezza), il modello basato sull'estrazione diretta della nuvola di punti restituisce stime sistematicamente e gravemente sottodimensionate (registrando diametri medi compresi tra 3.08 cm e 3.23 cm). Questo errore è la diretta conseguenza dei "buchi" di profondità: il sensore a infrarossi fatica a leggere i bordi curvi o parzialmente riflettenti dell'oggetto, generando una nuvola di punti errata. Al contrario, il modello *volume*, ancorando il calcolo alla maschera semantica bidimensionale, azzera questo difetto: per il bicchiere rosso, ad esempio, restituisce un diametro di 6.99 cm e un'altezza di 11.87 cm, sfiorando l'esattezza teorica. Un comportamento analogo si riscontra sulle misurazioni sferiche, dove la pallina

bianca piccola (*ground truth* di 5.5 cm) viene stimata a 5.67 cm dal modello volumetrico ibrido.

Il divario più critico ai fini della manipolazione robotica emerge tuttavia nell'analisi della stabilità temporale, quantificata attraverso le oscillazioni dimensionali (la variazione del diametro/altezza tra un fotogramma e l'altro) e il *jitter* del baricentro spaziale (la fluttuazione del *Tool Center Point* tridimensionale), (*Tabella 3.2*). Il modello *depth* si dimostra altamente inaffidabile, con oscillazioni dimensionali che superano frequentemente i 2.5 cm sui cilindri e instabilità del baricentro che raggiungono i 6.6 cm sul telecomando e i 7.7 cm sul mouse. L'architettura "*volume"* abbatte questo rumore di un intero ordine di grandezza: l'oscillazione dimensionale sul bicchiere rosso crolla da 2.58 cm a 0.02 cm, mentre la stabilità del baricentro si assesta su variazioni inferiori a 0.4 cm per gran parte dei target geometrici semplici.

Oggetto target	*Jitter baricentro (cm)*	*Oscillazione diametro o larghezza (cm)*	*Oscillazione altezza (cm)*	*Velocità (Hz)*
Cup giallo depth	2.992	2.55	2.76	15.8
Cup giallo volume	0.643	0.33	0.24	17.9
Cup rosso depth	3.066	2.58	2.73	15.8
Cup rosso volume	0.323	0.02	0.2	17.6
Cup verde depth	0.609	0.28	2.34	16.3
Cup verde volume	0.543	0.18	0.39	17.8
Cup viola depth	0.617	0.18	1.88	15.5
Cup viola volume	0.321	0.02	0.32	18.0
Cup di legno depth	2.572	1.81	2.64	16.4
Cup di legno volume	0.329	0.16	0.16	17.6
Ball bianca e rossa depth	2.278	0.38	0	21.7
Ball bianca e rossa volume	0.332	0.26	0	23.2
Ball gialla depth	1.195	0.27	0	21.7
Ball gialla volume	0.269	0.12	0	21.2
Ball bianca depth	6.343	0.5	0	21.3
Ball bianca volume	0.14	0.19	0	23.9
Telecomando depth	6.665	3.54	3.26	15.5
Telecomando volume	4.576	0.33	0.48	17.1
Mouse depth	7.712	2.32	1.94	16.2
Mouse volume	1.976	0.14	0.2	17.4

Tabella 3.2 Valutazione comparativa delle pipeline di visione (Depth vs Volume): oscillazione tra i vari frame delle stime dimensionali (diametro, larghezza e altezza) e jitter del baricentro (vedi equazione paragrafo 3.5.2 Protocollo Di Test: Performance A Confronto).

Anche sul fronte dell'efficienza computazionale, la nuova architettura dimostra prestazioni superiori. Sebbene la pipeline *volume* includa elaborazioni analitiche bidimensionali aggiuntive (come il calcolo della *PCA* 2D e l'estrazione della *ROI* di profondità), la rinuncia all'elaborazione geometrica di intere nuvole di punti instabili permette al sistema di operare a frequenze costantemente maggiori (raggiungendo picchi di 23.9 Hz, contro una media di 15-16 Hz del modello precedente), garantendo una fluidità ideale per applicazioni in tempo reale.

Alla luce di questa evidente superiorità in termini di aderenza al *ground truth*, drammatica riduzione del *jitter* spaziale ed efficienza di calcolo, l'approccio ibrido volumetrico è stato adottato come architettura definitiva per le successive fasi del progetto.

3.5 Valutazione Comparativa E Selezione Del Modello YOLO

Al fine di selezionare l'architettura neurale *YOLO* più idonea per la pipeline di *pick and place*, è stata condotta una campagna sperimentale di acquisizione e analisi volta a confrontare le prestazioni di tre diverse generazioni del modello: *YOLOv8*, *YOLOv11* e *YOLOv26*. Per ciascuna di esse sono state valutate sia la variante *Nano* che la variante *Small*, impiegando esclusivamente le versioni ottimizzate per la segmentazione delle istanze.

Il set di oggetti fisici impiegato per i test è il medesimo già adottato per la validazione dell'algoritmo di ricostruzione 3D. Tuttavia, per valutare concretamente la robustezza dei modelli, le acquisizioni sono state effettuate variando sistematicamente le condizioni operative e ambientali. Nello specifico, sono state alterate la posizione spaziale degli oggetti sul piano di lavoro, l'intensità dell'illuminazione e la distanza dal sensore visivo, con l'obiettivo di sottoporre la *pipeline* di visione a uno *stress test* rappresentativo dei reali imprevisti operativi possibili. In un'applicazione robotica reale, infatti, il sistema non opererà mai in condizioni perennemente ideali, ed è fondamentale quantificarne i limiti di fallimento. In particolare, le acquisizioni si sono svolte nelle seguenti condizioni:

- Condizioni ottimali (con base d’appoggio chiara: tavolo da laboratorio)
- Luce scarsa
- Base del piano d’appoggio scura
- Maggiore distanza dalla telecamera con base d’appoggio chiara
- Maggiore distanza dalla telecamera con base d’appoggio scura

La configurazione *condizioni ottimali* (a una distanza nominale di 95 cm) è stata utile per stabilire un *baseline* prestazionale, ovvero il *Best-Case Scenario*. Le variabili di disturbo sono state poi introdotte per isolare specifiche vulnerabilità hardware e software. L'utilizzo di una *base del tavolo scura* aveva lo scopo di forzare il mimetismo cromatico, testando la capacità delle reti neurali di non confondere i bordi dell'oggetto (specialmente

quelli a basso profilo come il mouse) con le ombre o con lo sfondo. Similmente, l'acquisizione in *luce scarsa* è stata inserita per valutare la tenuta del sistema al diminuire della definizione dei gradienti di colore e all'aumentare del rumore ottico (la "grana" del sensore), che tipicamente innesca instabilità nella maschera di segmentazione. Infine, le acquisizioni a *distanza maggiore* (portate a 155 cm, testate sia su base chiara che scura) rappresentano il *Worst-Case Scenario*. Allontanando il *target*, si riduce drasticamente la densità di *pixel* utili per definirne i contorni. Combinando questa carenza di risoluzione con ostacoli come una superficie scura, ho potuto verificare quale architettura di rete (*Nano* o *Small*) possedesse la reale capacità di mantenere un *Detection Rate* affidabile, evitando distorsioni della *Bounding Box* o la totale perdita dell'oggetto prima della fase di avvicinamento del braccio robotico.

3.5.1 Strumento Di Analisi Offline: Sviluppo Della GUI In Tkinter

Al fine di condurre la campagna di validazione dei modelli neurali in modo rigoroso, sistematico ed efficiente, è stato progettato e sviluppato un applicativo *software* dedicato, dotato di un'interfaccia grafica (*GUI*) realizzata tramite la libreria standard *Tkinter*. La necessità di questo strumento nasce dall'esigenza di slegare la fase di valutazione prestazionale dai limiti fisici e temporali dell'acquisizione in tempo reale. L'applicativo, infatti, è stato concepito per eseguire un'elaborazione in *batch* (seriale e automatizzata) di acquisizioni fisiche preregistrate, operando in modalità interamente *offline*. Attraverso la finestra di dialogo dell'interfaccia, l'utente è in grado di definire agilmente due parametri di *input* fondamentali: la *directory* radice (la "cartella madre" contenente l'intero dataset delle sessioni di test registrate in laboratorio) e la specifica architettura neurale da mettere alla prova (selezionabile da un menù a tendina che include le varianti *Nano* e *Small* delle generazioni *YOLOv8, YOLOv11 e YOLOv26*).

Una volta inizializzato il processo, il *software* automatizza l'intera *pipeline* di calcolo. Invece di interrogare i sensori *hardware* della telecamera, l'algoritmo itera in modo autonomo all'interno delle sottocartelle del dataset, sincronizzando la lettura del flusso video a colori con il caricamento delle matrici di profondità grezze, precedentemente salvate fotogramma per fotogramma in formato numerico (*.npy*).

L'applicativo esegue quindi la segmentazione visiva e il calcolo della ricostruzione volumetrica su ogni singolo *frame* del video testato. Al termine dell'elaborazione di ciascuna cartella, il sistema genera automaticamente un *output* completo e strutturato: esporta le registrazioni video dell'inferenza (sia in 2D che nell'ambiente virtuale 3D) e compila in automatico i *report* analitici in formato *CSV*. Tali file contengono sia lo storico metrico *frame by frame*, sia le misurazioni globali aggregate (come le medie dimensionali, le oscillazioni di lettura e il *jitter* del baricentro spaziale), fornendo così la

base di dati quantitativa essenziale per un confronto oggettivo tra i vari modelli di intelligenza artificiale.

3.5.2 Protocollo Di Test: Performance A Confronto

Il protocollo di valutazione è stato strutturato per misurare quantitativamente l'affidabilità e l'applicabilità pratica di ciascun modello neurale (*YOLOv8, YOLOv11* e *YOLOv26*) all'interno della *pipeline* volumetrica. L'analisi comparativa si è concentrata su tre metriche prestazionali fondamentali: la continuità del riconoscimento visivo, l'accuratezza della stima geometrica e la stabilità tridimensionale del punto di presa (baricentro dell'oggetto).

La prima metrica indagata è stata il *Detection Rate* (tasso di rilevamento). Questo parametro esprime la capacità della rete neurale di riconoscere e segmentare il *target* in modo continuo lungo l'intera durata dell'acquisizione, ed è calcolato come il rapporto percentuale tra il numero di fotogrammi in cui l'oggetto è stato rilevato con successo e il numero totale di fotogrammi validi del video. Un *Detection Rate* prossimo al 100% è indice di un modello visivo altamente robusto, capace di mantenere il tracciamento anche a fronte di variazioni prospettiche, riflessi o condizioni di illuminazione subottimali.

La seconda fase del protocollo ha valutato la precisione morfologica dell'algoritmo. Le stime dimensionali (diametro, altezza o larghezza) restituite dal *software* per ciascun fotogramma sono state mediate sull'intera sessione di test e confrontate con il *ground truth* fisico degli oggetti. Per quantificare l'affidabilità della misurazione, l'errore è stato valutato calcolando l'errore relativo percentuale rispetto alla dimensione reale:

$$Errore\ \% = \left|\frac{M_{stimata} - M_{reale}}{M_{reale}}\right| * 100$$

Dove $M_{stimata}$ rappresenta la dimensione (altezza o larghezza/diametro) media (in millimetri) calcolata dall'architettura e M_{reale} costituisce la misura di riferimento verificata manualmente con strumenti di precisione (*ground truth*). L'analisi di questo errore ha permesso di stabilire quale modello neurale offrisse le maschere di segmentazione più conformi all'ingombro reale dell'oggetto.

Infine, la metrica più critica ai fini di un'operazione di *pick and place* robotico è risultata essere la stabilità spaziale del baricentro stimato, che coincide con il *Tool Center Point* (*TCP*) *target*. Fisiologicamente, a causa del rumore visivo e di micro-variazioni nei bordi della maschera di segmentazione tra frame adiacenti, le coordinate del baricentro calcolato possono subire delle lievi fluttuazioni. Questa instabilità meccanica, definita *jitter*, è stata quantificata registrando l'escursione massima (la differenza tra il valore massimo e il valore minimo) lungo i tre assi cartesiani $(\Delta X, \Delta Y, \Delta Z)$ durante l'intera

acquisizione. Il *jitter* tridimensionale complessivo è stato quindi calcolato tramite la norma euclidea di tali oscillazioni:

$$Jitter_{3D} = \sqrt{\Delta X^2 + \Delta Y^2 + \Delta Z^2}$$

Valori minimi di $Jitter_{3D}$ indicano una stabilità spaziale eccellente. Un baricentro stabile, infatti è un requisito ingegneristico inderogabile: un'eccessiva fluttuazione del punto *target*, genererebbe traiettorie di avvicinamento a scatti, compromettendo la precisione del robot e aumentando il rischio di collisioni o prese instabili.

3.5.3 Analisi Comparativa Del Detection Rate

Le reti di taglia "*Nano*" si sono rivelate inadeguate per un impiego robotico autonomo (tasso di successo del 56-60%), fallendo sistematicamente nell'identificazione di oggetti piatti o a basso contrasto. Il passaggio alla taglia "*Small*" (come *YOLOv8*) ha incrementato i rilevamenti oltre l'85%, introducendo tuttavia criticità di *over bounding*: il modello tendeva a fondere gli oggetti chiari con i riflessi del tavolo, restituendo stime tridimensionali errate. Il compromesso ingegneristico ottimale è stato raggiunto solo con i modelli "*Small*" di ultima generazione (*v11 e v26*). Pur attestandosi su un tasso di rilevamento leggermente inferiore (circa 80%), queste reti eliminano i falsi positivi e garantiscono al robot *TIAGo* quella stabilità volumetrica e spaziale indispensabile per pianificare ed eseguire le traiettorie di presa in totale sicurezza.

3.5.4 *Confronto Dell'errore Dimensionale*

Per valutare la robustezza dell'algoritmo non è sufficiente considerare l'errore medio aggregato, ma risulta imprescindibile analizzare la varianza e la stabilità temporale delle stime *frame* per *frame*. A tale scopo, le misurazioni esportate dall'applicativo di *batch analysis* come file *CSV*, sono state rielaborate sotto forma di *Box Plot*, evidenziando il comportamento delle architetture nelle diverse condizioni operative. Nella *Figura 3.6* sono riportati i risultati con i *Box Plot* aggregati per modello *YOLO* utilizzato, per una visione complessiva del comportamento dei vari modelli rispetto agli errori dimensionali compiuti dal sistema (sulla stima di larghezza/diametro e altezza). Nel grafico l'asse verticale viene troncato al 20% per poter apprezzare meglio la distribuzione degli errori generale. Possiamo infatti osservare che solamente due modelli compiono errori maggiori o vicini al 20% rispetto, in particolare, alla stima dell'altezza; perciò, andando a spostare la vista verso gli errori medi possiamo osservare meglio la distribuzione dei box plot degli altri modelli.

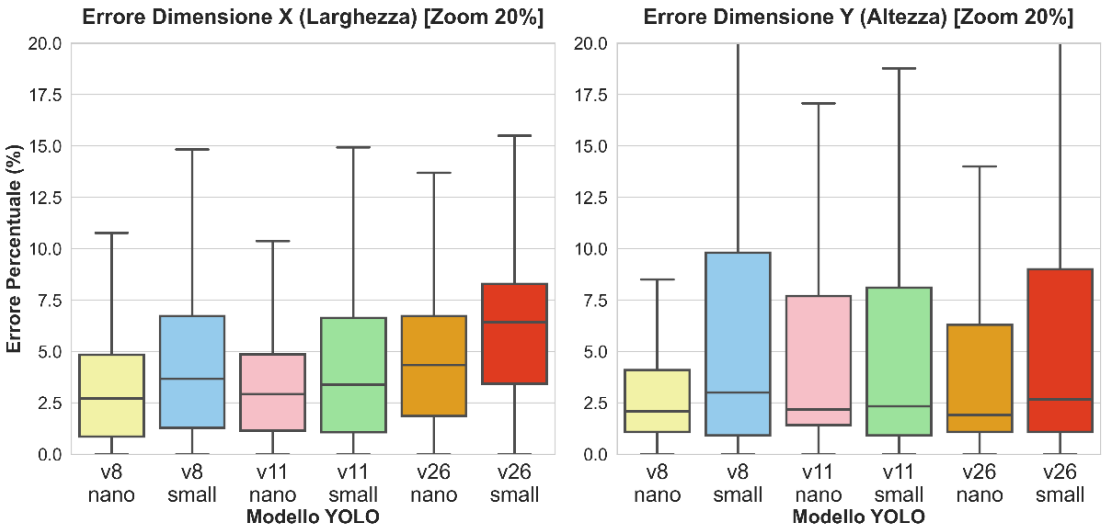


Figura 3.6 Box plot dell'errore relativo dimensionale dei modelli YOLO: vista di dettaglio con limite di scala al 20% per agevolare il confronto della tendenza centrale.

Osservando le stime relative alla larghezza (X) e all'altezza (Y), si conferma la tendenza delle reti *Nano* a mantenere errori mediani inferiori e una minore dispersione statistica, un vantaggio teorico che viene tuttavia vanificato dalla loro osservata incapacità di rilevamento negli scenari a basso contrasto o con luce scarsa. Spostando il focus sui modelli *Small*, i grafici svelano in modo inequivocabile le criticità delle singole versioni. In particolare, il modello *YOLOv8 Small* (box azzurro) presenta un'elevatissima varianza spaziale, con scatole molto ampie e *whiskers* prolungati che testimoniano un *jittering* severo e un'incertezza cronica nella definizione dei bordi. All'estremo opposto, *YOLOv26 Small* (box rosso) mostra distribuzioni relativamente più compatte ma centrate sui valori mediani di errore planare più alti dell'intero set, confermando visivamente la sua tendenza sistematica alla dilatazione conservativa della maschera (*over-bounding*).

Osservando la *Figura 3.7*, possiamo vedere un esempio degli errori dimensionali compiuti dalle varie reti *YOLO* in relazione agli oggetti specifici, in particolare ho scelto di riportare la condizione di acquisizione denominata come “condizione ottimale”, che va a richiamare quella che idealmente sarebbe la condizione di lavoro tipica da laboratorio in uno scenario di *pick and place*, quindi con presenza di luce ottimale e posizione *standard* dell’oggetto a distanza medio-vicina dalla telecamera sul piano di lavoro.

Nello specifico, l'analisi delle distribuzioni statistiche tramite *Box Plot* evidenzia che gli oggetti caratterizzati da geometrie a basso profilo e materiali ad elevato assorbimento ottico (quali il mouse e il telecomando) presentano errori dimensionali significativamente più elevati e una marcata instabilità temporale (con errori stimati tra il 30 e il 40 %). Tale criticità si manifesta graficamente con scatole molto ampie e *whiskers* (baffi) estesi, indice di un forte rumore di misura riconducibile a due fattori concorrenti: da un lato,

l'incertezza della rete neurale nel discriminare i bordi di oggetti molto piatti (che genera *jittering* della maschera 2D); dall'altro, la difficoltà fisiologica del sensore a infrarossi nel rilevare in modo stabile la quota Z su superfici piatte e assorbenti. Abbiamo altri due casi anomali rispetto alle Sports Ball Bianca e Bianca e Rossa, due palline da gioco che sono poco contrastate rispetto al piano di lavoro e di conseguenza la rete *YOLO* fatica a rilevare correttamente, generando dei *bounding box* che non hanno stabilità tra i vari frame e di conseguenza una errata lettura delle dimensioni degli oggetti, con errori altissimi.

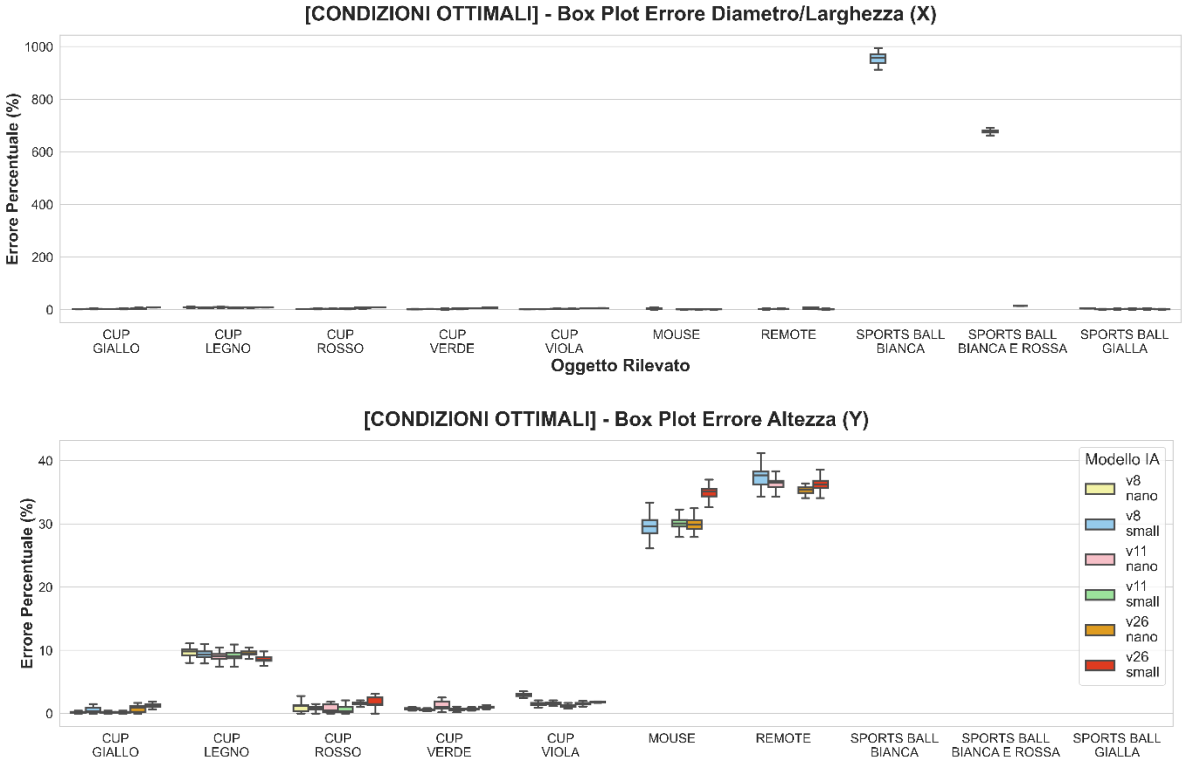


Figura 3.7 Box plot degli errori dimensionali nella condizione di acquisizione “condizioni ottimali”: posizione dell’oggetto rispetto alla telecamera pari a 95 cm di distanza.

L'osservazione delle distribuzioni statistiche troncate alla soglia del 15% (*Figura 3.8*) permette di isolare il reale comportamento geometrico delle diverse architetture *YOLO*, fornendo le evidenze necessarie per la selezione del modello definitivo, perché questo ci permette di escludere i valori outlier dalla visualizzazione e apprezzare meglio le distribuzioni. Tale accorgimento si è reso necessario per escludere i valori anomali (*outlier*) generati da specifici casi limite (come la sfera bianca o il *remote*), che avrebbero altrimenti compromesso la leggibilità dei grafici. Considerando che l'errore fisiologico del sistema si attesta intorno al 12%, la soglia del 15% consente di apprezzare chiaramente le distribuzioni nominali, scartando esclusivamente i picchi anomali estremi.

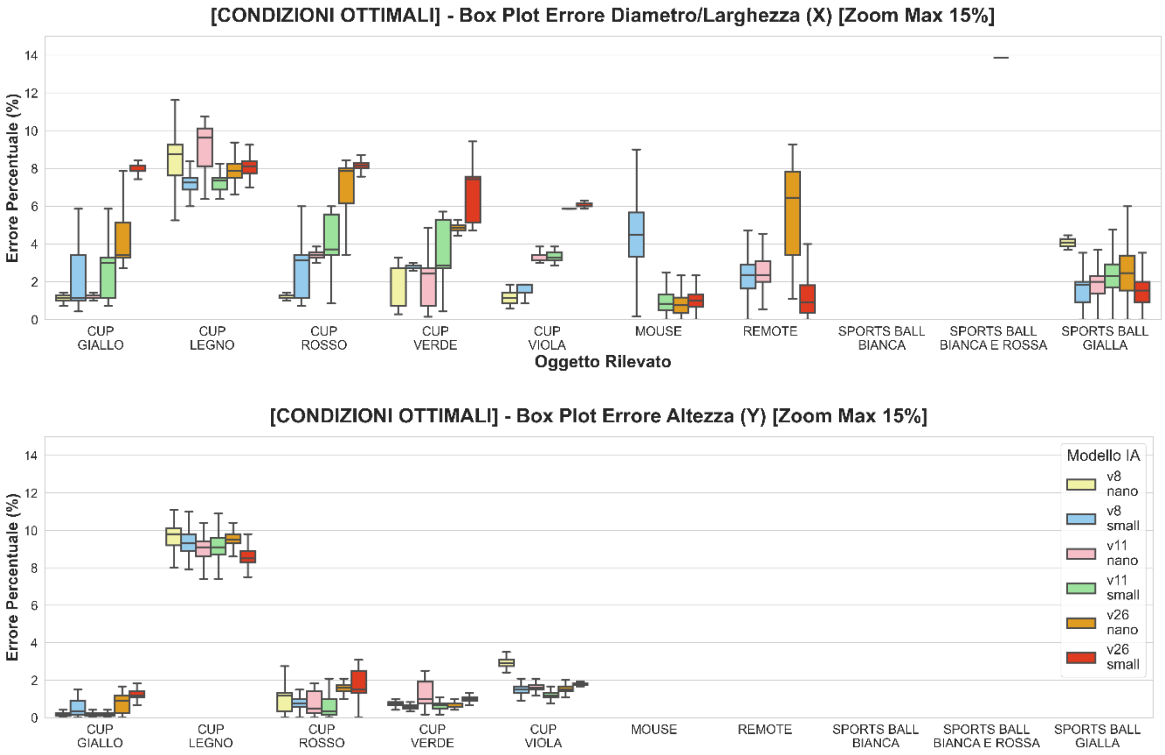


Figura 3.8 Box plot degli errori dimensionali della condizione di acquisizione “condizioni ottimali”: posizione dell’oggetto rispetto alla telecamera pari a 95 cm di distanza. Con zoom al 15%. I trattini orizzontali isolati nei grafici indicano campioni a varianza nulla, mentre le scatole mancanti segnalano veri fallimenti di rilevamento.

L'analisi dei solidi cilindrici ad alto contrasto mostra in generale ottimi risultati, con errori quasi nulli e basso jittering, ad eccezione del bicchiere di legno, dove il basso contrasto cromatico con il piano d'appoggio (entrambi bianchi) porta la maschera *YOLO* a faticare nell'aderire ai contorni, generando errori intorno al 10%. Sull'asse X (diametro/larghezza) emerge invece un fattore discriminante tra i modelli: le reti *Nano* sono precise ma fragili su profili piatti o poco illuminati, mentre la *YOLOv26 Small* tende a un *over bounding* sistematico (maschere dilatate ed errori sul 4-8%) pur di non perdere mai il *target*. I *Box Plot* degli scenari critici (luce scarsa, distanza crescente, base scura) non sono stati inclusi, ma l'analisi sui dati *CSV* grezzi ha comunque confermato la gerarchia dei modelli: al peggiorare delle condizioni visive, le reti *Nano* collassano (*Detection Rate* anche pari a zero), rendendo necessario l’utilizzo di una rete *Small*, mentre all'aumentare della distanza, crescono inevitabilmente gli errori dimensionali per i limiti dell'infrarosso nella misura della profondità e per la maggiore instabilità delle maschere tra un frame e l'altro. Sulla base di questi risultati, *YOLOv11 Small* si conferma la scelta ottimale per il *pick and place* del robot *TIAGo*: unisce la robustezza tipica della famiglia *Small* alla precisione dei bordi e alla fedeltà dimensionale, senza la dilatazione eccessiva di *v26 Small*.

3.5.5 Analisi Statistica Del Jitter Del Baricentro

Per quantificare matematicamente il fenomeno del *jittering* (il "tremolio" della maschera di segmentazione tra fotogrammi adiacenti), è stata condotta un'analisi statistica basata sul calcolo della deviazione standard delle coordinate tridimensionali del baricentro. Un valore di deviazione standard ridotto indica un *bounding box* geometricamente rigido e affidabile. Al fine di non alterare la metrica a causa delle diverse distanze fisiche di acquisizione, i dati sono stati raggruppati per condizione operativa. È stato sviluppato uno *script* dedicato che: estrae le coordinate *frame per frame* dai file *CSV*, calcola media e deviazione standard dei valori dei singoli video e aggrega i risultati per scenario e modello neurale (*Tabella 3.3* e *Tabella 3.4*).

Condizione	*Modello YOLO*	*Media ± Dev. Std X (cm)*	*Media ± Dev. Std Y (cm)*	*Media ± Dev. Std Z (cm)*
Condizioni Ottimali	YOLOv8_nano	0.020 ± 0.006	0.023 ± 0.004	0.096 ± 0.048
Condizioni Ottimali	YOLOv8_small	0.160 ± 0.275	0.192 ± 0.502	0.538 ± 0.896
Condizioni Ottimali	YOLOv11_nano	0.035 ± 0.032	0.026 ± 0.012	0.202 ± 0.291
Condizioni Ottimali	YOLOv11_small	0.021 ± 0.010	0.024 ± 0.008	0.129 ± 0.109
Condizioni Ottimali	YOLOv26_nano	0.032 ± 0.034	0.024 ± 0.008	0.230 ± 0.297
Condizioni Ottimali	YOLOv26_small	0.032 ± 0.039	0.024 ± 0.007	0.234 ± 0.331
Base Piano Scura	YOLOv8_nano	0.018 ± 0.005	0.026 ± 0.009	0.094 ± 0.043
Base Piano Scura	YOLOv8_small	0.027 ± 0.035	0.035 ± 0.029	0.223 ± 0.374
Base Piano Scura	YOLOv11_nano	0.019 ± 0.005	0.029 ± 0.009	0.093 ± 0.053
Base Piano Scura	YOLOv11_small	0.033 ± 0.035	0.038 ± 0.026	0.277 ± 0.486
Base Piano Scura	YOLOv26_nano	0.019 ± 0.008	0.031 ± 0.012	0.109 ± 0.049
Base Piano Scura	YOLOv26_small	0.030 ± 0.045	0.041 ± 0.039	0.317 ± 0.652
Luce Scarsa	YOLOv8_nano	0.062 ± 0.098	0.088 ± 0.104	0.800 ± 1.954
Luce Scarsa	YOLOv8_small	0.073 ± 0.103	0.064 ± 0.088	0.853 ± 1.750
Luce Scarsa	YOLOv11_nano	0.056 ± 0.093	0.062 ± 0.095	0.710 ± 1.839
Luce Scarsa	YOLOv11_small	0.200 ± 0.349	0.092 ± 0.116	1.927 ± 3.313
Luce Scarsa	YOLOv26_nano	0.057 ± 0.099	0.065 ± 0.101	0.799 ± 1.940
Luce Scarsa	YOLOv26_small	0.120 ± 0.151	0.103 ± 0.134	1.332 ± 1.951

Tabella 3.3 Analisi dell'instabilità spaziale (Jittering): medie e deviazioni standard lungo gli assi X, Y e Z registrate dai diversi modelli YOLO negli scenari ambientali di test Condizioni Ottimali, Base Piano Scura e Luce Scarsa.

Condizione	*Modello YOLO*	*Media ± Dev. Std X (cm)*	*Media ± Dev. Std Y (cm)*	*Media ± Dev. Std Z (cm)*
Maggiore Distanza su Piano con Base Chiara	YOLOv8_nano	0.033 ± 0.014	0.059 ± 0.007	0.211 ± 0.097
Maggiore Distanza su Piano con Base Chiara	YOLOv8_small	0.044 ± 0.022	0.066 ± 0.039	0.305 ± 0.313
Maggiore Distanza su Piano con Base Chiara	YOLOv11_nano	0.032 ± 0.012	0.062 ± 0.026	0.237 ± 0.162
Maggiore Distanza su Piano con Base Chiara	YOLOv11_small	0.053 ± 0.033	0.079 ± 0.056	0.397 ± 0.382
Maggiore Distanza su Piano con Base Chiara	YOLOv26_nano	0.033 ± 0.005	0.060 ± 0.008	0.246 ± 0.042
Maggiore Distanza su Piano con Base Chiara	YOLOv26_small	0.056 ± 0.043	0.092 ± 0.064	0.460 ± 0.480
Maggiore Distanza su Piano con Base Scura	YOLOv8_nano	0.825 ± 1.787	1.545 ± 3.354	7.464 ± 16.528
Maggiore Distanza su Piano con Base Scura	YOLOv8_small	0.554 ± 1.326	1.025 ± 2.507	4.986 ± 12.257
Maggiore Distanza su Piano con Base Scura	YOLOv11_nano	0.687 ± 1.610	1.341 ± 3.035	6.332 ± 14.910
Maggiore Distanza su Piano con Base Scura	YOLOv11_small	0.897 ± 1.370	1.162 ± 2.672	5.989 ± 12.912
Maggiore Distanza su Piano con Base Scura	YOLOv26_nano	0.035 ± 0.011	0.048 ± 0.021	0.205 ± 0.078
Maggiore Distanza su Piano con Base Scura	YOLOv26_small	0.559 ± 1.311	0.958 ± 2.511	4.704 ± 12.294

Tabella 3.4 Analisi dell'instabilità spaziale (Jittering): medie e deviazioni standard lungo gli assi X, Y e Z registrate dai diversi modelli YOLO negli scenari ambientali di test Maggiore Distanza su Piano con Base Chiara e Scura.

Nella *Tabella 3.4* riportata sopra, il primo valore quantifica il *jittering* medio (in centimetri) subìto dal baricentro all'interno della singola acquisizione, mentre il secondo

valore misura invece la varianza di tale tremolio tra video differenti appartenenti alla medesima categoria.

Nelle acquisizioni nominali a 95 cm dal *target* (*Condizioni Ottimali* e *Base Piano Scura*), il *jittering* medio si attesta su valori sub millimetrici o di pochi millimetri, con un errore medio sull'asse Z inferiore a 0.3 cm, ed anche la varianza associata risulta altrettanto contenuta. Questo dato certifica la correttezza matematica della ricostruzione, di fatti, in assenza di degrado visivo ambientale, il modello genera maschere estremamente rigide, garantendo coordinate spaziali precise e altamente ripetibili. L'analisi evidenzia, infatti, che la scarsa illuminazione riduce la capacità delle reti di operare sui gradienti cromatici, generando un aumento del *jittering* sull'asse Z (tra 0.7 e 1.9 cm) e un comportamento "a scalino" dovuto al rilevamento intermittente dell'oggetto. Il degrado prestazionale massimo si osserva combinando un'elevata distanza (155 cm) con il mimetismo cromatico: in questo scenario limite, modelli come *YOLOv8 Nano* mostrano totale inaffidabilità.

3.5.6 Scelta del modello YOLO

Dal confronto comparativo, *YOLOv11 Small* emerge come l'architettura più performante e robusta. A differenza delle versioni "*Nano*" (instabili in condizioni avverse), di *YOLOv8 Small* e della variante *v26 Small* (caratterizzata da maggiore dispersione), *YOLOv11 Small* minimizza il *jittering* spaziale in ogni condizione. Mantenendo deviazioni standard contenute anche con luce scarsa o distanze elevate, questa rete garantisce il miglior compromesso tra accuratezza metrica e coerenza temporale, confermandosi la scelta ottimale per il sistema.

Capitolo Quattro. Sviluppo, Simulazione E Validazione Preliminare Della Pipeline In Ambiente ROS 2

Al fine di testare i due nodi necessari al completamento del *task* di *pick and place* da parte del robot manipolatore *TIAGo*, sono state eseguite delle prove di simulazione utilizzando il software *Gazebo* tramite il *framework ROS 2*. I due nodi realizzati si suddividono in: un primo nodo che consente a *TIAGo* di vedere l'ambiente davanti a sé e quindi individuare potenzialmente degli oggetti che si trovano sul piano di lavoro, riconoscendoli e calcolandone le dimensioni e la posizione del baricentro, e, in seguito, un secondo nodo dedicato al *task* di *pick and place.* Quest'ultimo, sulla base dei dati pubblicati dal primo nodo (dimensioni di oggetto e piano di lavoro e rispettivi baricentri), comanda l'approccio all'oggetto attraverso l'*end effector*, prende (*grasp*), solleva e riposiziona l'oggetto completando il task di *pick and place*. La fase di simulazione è volta all'applicazione della *pipeline,* una volta validata, su robot reale presente nel laboratorio *Living Lab* del centro di eccellenza *REDI* dell'Università di Pavia. Durante la fase di validazione sperimentale sono stati usati i software *Gazebo* e *RViz*. Gazebo è un motore di simulazione fisica 3D che ricrea le dinamiche del mondo reale, e tramite il quale si genera la vera e propria scena di simulazione con piano di lavoro e oggetto, e si collaudano i nodi realizzati; RViz è uno strumento di visualizzazione grafica che permette di ispezionare visivamente ciò che il robot "percepisce", mostrando in tempo reale il contenuto dei vari topic come *Point Cloud* e *Marker*.

4.1 L'ambiente Virtuale Gazebo

Il simulatore di robotica 3D *open source Gazebo* (*Figura 4.1*) è uno strumento che consente di creare mondi tridimensionali complessi con l'utilizzo di un gran numero di oggetti e robot in dinamica e statica, offrendo ampie opportunità di ricerca agli utilizzatori [44].

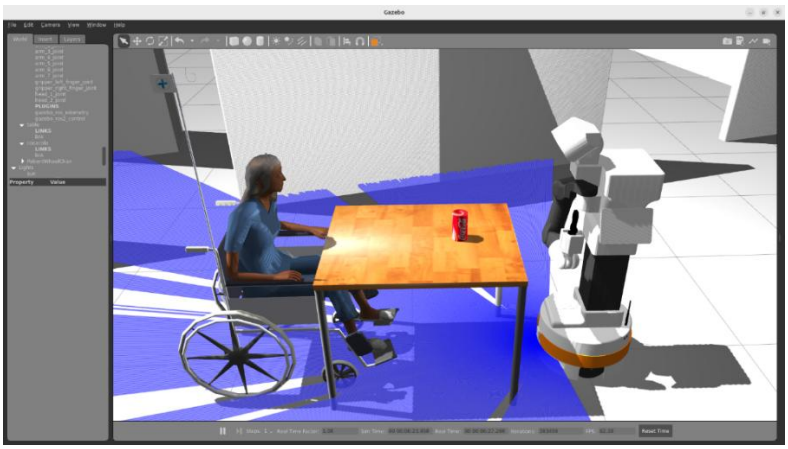


Figura 4.1 Esempio di scenario realizzabile attraverso l'ambiente simulato Gazebo.

Attraverso la simulazione è possibile testare i robot e la *pipeline* realizzata per il loro funzionamento evitando infortuni agli esseri umani e danni alle superfici, gli oggetti e i robot stessi riducendo il carico sull'*hardware* reale del robot e il dispendio di tempo durante sessioni di laboratorio reali [44]. Dopo aver eseguito la simulazione e aver validato i codici è possibile eseguire direttamente sul robot fisico reale in laboratorio tutta la *pipeline* senza modifiche sostanziali [45]. All'interno della simulazione tutti gli oggetti simulati possiedono massa, velocità, attrito e numerosi altri attributi che consentono loro di comportarsi in modo realistico quando vengono spinti, tirati, rovesciati o trasportati [46], [47]. L'impiego congiunto dell'ambiente di simulazione *Gazebo* e dell'interfaccia *RViz* permette di sviluppare e validare architetture software che possono essere implementate senza modifiche sul robot reale, sfruttando *RViz* per un'analisi visiva e diagnostica del comportamento del sistema [45].

4.1.1 Impostazione Della Scena Di Simulazione Controllata

Al fine di eseguire le prove del nodo di visione e, successivamente, del nodo di *grasping*, il primo passaggio eseguito è rappresentato dall'apertura di *Gazebo* tramite il comando a terminale su *Visual Studio Code*: *ros2 launch tiago_229_gazebo tiago_gazebo.launch.py*. Il comando esegue un file di lancio *ROS 2* che avvia l'ambiente di simulazione virtuale *Gazebo*, genera al suo interno il modello tridimensionale del robot *TIAGo* e ne inizializza tutti i motori e i sensori virtuali, rendendolo pronto a interagire con gli algoritmi di visione e grasping. A seguito dell'avvio del simulatore *Gazebo*, l'interfaccia grafica restituisce la visualizzazione tridimensionale della scena, all'interno della quale è possibile inserire e manipolare dinamicamente vari modelli. Per replicare lo scenario operativo del *task* di *pick and place*, la scena viene popolata inserendo un piano di lavoro (corrispondente al modello *table0m8*) su cui vengono disposti degli oggetti target, come ad esempio una lattina di Coca Cola (*Figura 4.2*).

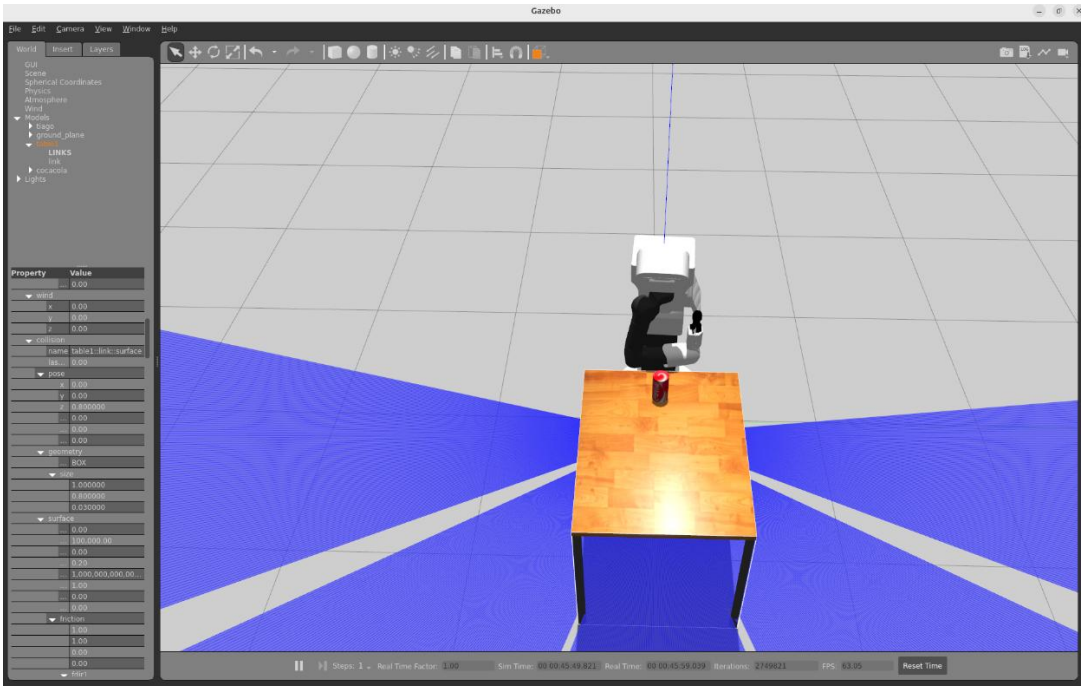


Figura 4.2 Visualizzazione su Gazebo della scena di simulazione controllata, popolata manualmente selezionando gli oggetti disponibili del sistema.

L'ambiente virtuale offre inoltre strumenti di ispezione diretta, permettendo di accedere alle proprietà dei modelli (nella sezione *Geometry*) per verificarne le dimensioni esatte, un passaggio fondamentale per ottenere il "*Ground Truth*" necessario a validare l'accuratezza dell'algoritmo di visione. Per configurare l'assetto operativo iniziale del robot, vengono inviati singoli messaggi di traiettoria (*JointTrajectory*) direttamente ai *controller* del busto e della testa tramite il terminale. Questa operazione è fondamentale per sollevare il torso di *TIAGo* e inclinare la telecamera verso il basso, garantendo così un'inquadratura ottimale e priva di occlusioni del piano di lavoro prima di attivare gli algoritmi di percezione visiva.

4.2 Implementazione E Integrazione Del Nodo Di Percezione Visiva

Il nodo *ROS 2* denominato *tiago_camera_node* funge da modulo di percezione visiva e ricostruzione tridimensionale, tramite il quale il robot è in grado di osservare la scena davanti a sé, riconoscendo eventuali oggetti *target* disposti sopra il piano di lavoro, calcolandone dimensioni e baricentro. Il nodo in particolare sottoscrive i *topic* della telecamera frontale per acquisire in tempo reale flussi video *RGB* (con *topic: /head_front_camera/rgb/image_raw*) e mappe di profondità (con *topic: /head_front_camera/depth_registered/image_raw*).

Il flusso di elaborazione si divide in due fasi principali:

- Il nodo genera la nuvola di punti a partire dai dati di profondità e applica l'algoritmo *RANSAC* per identificare l'equazione matematica del piano d'appoggio, estraendone il centro geometrico e le dimensioni spaziali, che rappresenteranno un dato utile al fine di costituire un oggetto di collisione considerato dal nodo di *grasping.* È infatti fondamentale informare il robot della presenza di oggetti di collisione nella scena, se questo non viene fatto, una volta che si elabora la *pipeline* in sede di simulazione reale di laboratorio il robot potrebbe scontrarsi con le gambe del tavolo o la sua stessa superficie, portando a danni agli oggetti o a sé stesso.
- Utilizzando il modello di segmentazione neurale *YOLO v11 segmentation*, vengono parallelamente individuati gli oggetti *target* sul piano d'appoggio e classificati in base alla loro morfologia (sfera, cilindro o parallelepipedo). Una volta estratti il baricentro e le dimensioni dell'oggetto, il nodo di visione pubblica queste coordinate sui rispettivi *topic*, affinché possano essere processate dal nodo deputato al *grasping*.

Come descritto nei paragrafi precedenti, anche in questo caso viene eseguito un filtraggio della maschera bidimensionale dell'oggetto sfruttando il piano d'appoggio precedentemente calcolato dall'algoritmo *RANSAC*, in modo da avere una maschera libera da interferenze causate da sfondo o ombre. Inoltre, viene applicata l'analisi delle componenti principali (*PCA*) sulle coordinate dei *pixel* per il calcolo delle dimensioni, viene estratta la regione di interesse sulla mappa di profondità per la corretta misurazione del baricentro, che viene successivamente *shiftato* di una quantità pari a mezzo spessore o raggio dell'oggetto, per posizionarlo correttamente all'interno dell'oggetto stesso.

4.2.1 Trasformazioni Spaziali ROS 2

Una volta ricavate le dimensioni e i baricentri degli oggetti e del piano di lavoro, tramite la libreria *TF2*, tutte le coordinate estratte vengono rototraslate dal sistema di riferimento ottico della telecamera (*head_front_camera_color_optical_frame*) al sistema di riferimento globale del robot (*base_footprint*), e infine pubblicate su appositi *topic* pronti per il nodo di *grasping*. In particolare, nell'ottica di simulazione e raccolta di dati, il nodo di visione è stato ingegnerizzato con una routine di valutazione statica integrata eseguendo una fase iniziale di *warm up* di 50 *frame* per consentire la stabilizzazione dell'albero delle trasformazioni. Al fine di garantire una base statistica completa e omogenea per ogni oggetto in esame, l'algoritmo non si interrompe dopo un numero fisso di interazioni, ma subordina la chiusura del test al raggiungimento di esattamente cento rilevamenti corretti. Ciò significa che il sistema continua a campionare il flusso video scartando i *frame* non validi (che corrispondono a mancati rilevamenti dell'oggetto da

parte della rete *YOLO*), elaborando di conseguenza un numero complessivo di *frame* potenzialmente superiore ai cento, per poter accumulare i campioni utili richiesti per un'analisi comparabile. Al termine dell'acquisizione, il sistema calcola automaticamente le metriche statistiche descritte precedentemente, dunque oscillazione spaziale del baricentro (*jitter*) ed errori percentuali rispetto alle dimensioni reali (*ground truth*), esportando i risultati in appositi file *CSV*.

4.2.2 Registrazione Delle Immagini RGB-D

Poiché l'ottica della telecamera *RGB* e quella del sensore a infrarossi (per la lettura della profondità), del robot *TIAGo* sono disallineate a causa delle loro posizioni fisiche differenti sul dispositivo (così come nella configurazione *stand alone*), è stato necessario applicare un nodo dedicato alla registrazione delle due immagini. Questo nodo in particolare, durante una prova di simulazione viene attivato come primo passaggio e successivamente viene attivato il nodo di visione che quindi sarà in grado di ricevere l'immagine registrata. Il risultato di questa elaborazione geometrica è la generazione di una nuova mappa di profondità perfettamente allineata che viene pubblicata in tempo reale sul *topic: /head_front_camera/depth_registered/image_raw*. A valle di questo processo il nodo principale di visione attinge direttamente a tale *topic* registrato garantendo una corrispondenza spaziale rigorosa e puntuale tra i *pixel* della maschera bidimensionale estratta dalla rete *YOLO* sull'immagine *RGB* e i corrispondenti valori metrici di profondità. Tale prerequisito si rivela fondamentale per consentire al sistema di calcolare in modo affidabile il baricentro spaziale e le dimensioni volumetriche 3D degli oggetti, azzerando gli errori di parallasse prima dell'invio dei dati al modulo di *grasping*.

4.2.3 Generazione Dei Marker E Visualizzazione Della Scena Tramite Ambiente RViz

L'architettura del nodo di visione prevede la generazione e la pubblicazione dinamica di una primitiva geometrica sotto forma di *marker* (tramite il messaggio di tipo *marker array* pubblicato sul topic: /grasping_markers). L'obiettivo dell'utilizzo dei *marker* è quello di abilitare monitoraggio e *debugging* visivo in tempo reale all'interno dell'ambiente di visualizzazione *RViz (Figura 4.3):* strumento di visualizzazione tridimensionale che permette di importare file, visualizzare modelli, analizzare le collisioni possibili tra oggetti e robot, controllare i dati dei sensori presenti (telecamere, laser, etc) e monitorare lo stato dei giunti tramite nodi come *robot_state_publisher* e *joint_state_publisher,* [47].

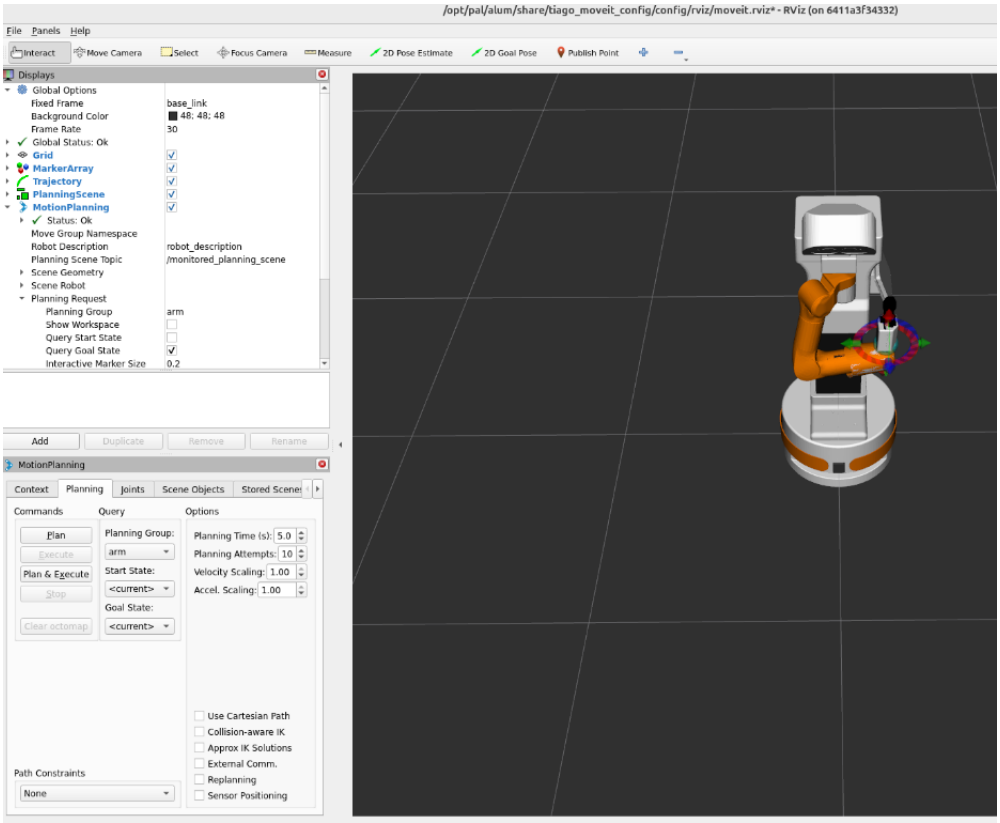


Figura 4.3 Software RViz.

Questa funzionalità rappresenta uno strumento diagnostico molto importante per validare la correttezza della *pipeline* di percezione prima dell'esecuzione di qualsiasi comando di manipolazione.

Nello specifico il sistema si occupa di istanziare e aggiornare due tipologie diverse di ostacoli virtuali:

- *Marker del piano di lavoro*: Sfruttando i dati calcolati a valle dell'algoritmo *RANSAC* e la stima del baricentro nel sistema globale rispetto al sistema *base_footprint*, il nodo genera un parallelepipedo che corrisponde all'ingombro rappresentato dal tavolo su cui verranno poggiati gli oggetti per il *task* di *pick and place*. L'ostacolo virtuale riproduce fedelmente l'ingombro spaziale del tavolo con le sue specifiche larghezza, profondità e altezza reale, e viene inoltre visualizzato con una colorazione semitrasparente che permette all'operatore di verificare visivamente il corretto filtraggio del piano.
- *Il marker dell'oggetto target:* Parallelamente, per l'oggetto individuato dalla rete *YOLO* il nodo genera un secondo *marker*, la cui geometria si adatta in modo intelligente alla classe morfologica di appartenenza (*Marker.SPHERE* per le sfere, *Marker.CYLINDER* per i cilindri e *Marker.CUBE* per gli oggetti parallelepipedi).

Anche in questo caso, le dimensioni e le coordinate spaziali di questo *marker* vengono aggiornate *frame per frame* in base ai risultati dell'analisi delle componenti principali e la profondità media, posizionando l'oggetto correttamente sul piano di lavoro.

4.2.4 Risultati Ottenuti Dai Test Di Simulazione Per Il Nodo Di Visione

Per validare il corretto funzionamento del nodo di visione, è stata condotta una prova di simulazione utilizzando due modelli offerti da *Gazebo*, nello specifico un oggetto con geometria cilindrica e uno di geometria sferica. Come illustrato nella *Figura 4.4*, l'oggetto *target* è stato posizionato sul tavolo di fronte al robot *TIAGo* all'interno dell'ambiente *Gazebo*. Per quantificare oggettivamente l'accuratezza dell'algoritmo, è stato estratto il *ground truth* rispetto alle dimensioni e la posizione dell'oggetto sfruttando direttamente l'interfaccia nella zona sinistra di *Gazebo*. Sono stati registrati i parametri esatti del *link* associato all'oggetto selezionato, leggendo dalla sezione dedicata alla geometria ("*geometry*") le dimensioni reali del modello e le sue coordinate spaziali assolute che variano con la posizione scelta sul piano di lavoro. Questi valori di riferimento sono stati poi confrontati con le stime prodotte dal nodo di visione in esecuzione e visualizzate in *RViz*, permettendo così di misurare con precisione l'errore del sistema nella stima dimensionale e spaziale dell'oggetto.

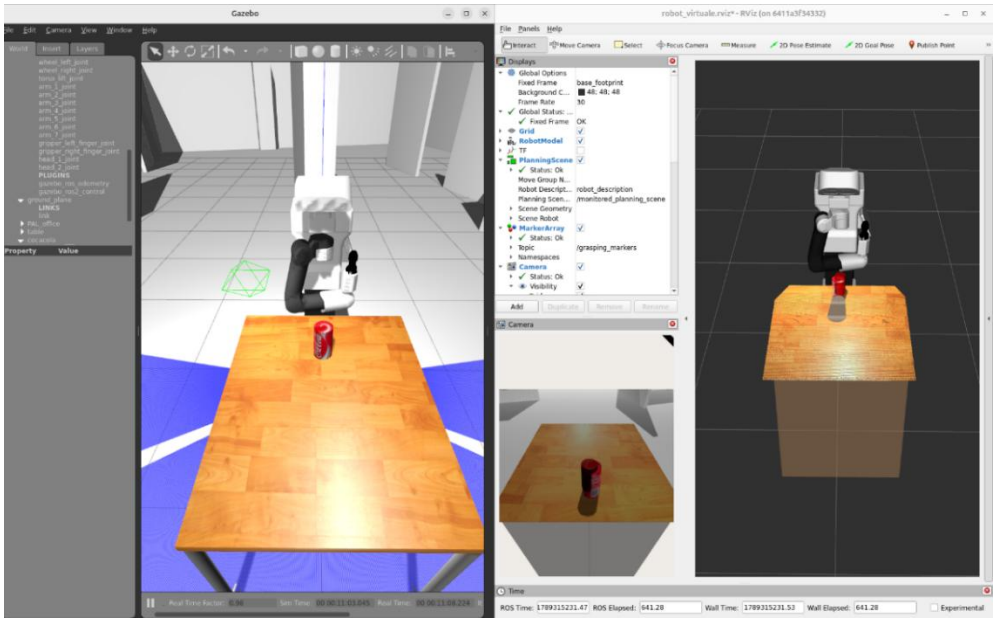


Figura 4.4 Test di validazione del nodo di visione in simulazione nella piattaforma Gazebo. A sinistra (in Gazebo) possiamo osservare la scena di simulazione con l'oggetto target posto sul piano di lavoro e il robot TIAGo posizionato davanti al tavolo. A destra troviamo invece la piattaforma RViz in cui possiamo osservare l'oggetto sottoforma di marker rosso sovrapposto alla vista RGB di TIAGo.

Prima di procedere con l'analisi dei dati quantitativi, è fondamentale chiarire la relazione spaziale tra i sistemi di riferimento utilizzati per il calcolo dell'errore. Il simulatore *Gazebo* restituisce le coordinate assolute degli oggetti basandosi sull'origine globale dell'ambiente virtuale (*world_frame*). Al contrario, la pipeline di visione sviluppata calcola la posa dell'oggetto convertendo i dati ottici rispetto al baricentro del robot (*base_footprint*) tramite l'albero delle trasformazioni (*TF2*), in modo da fornire coordinate direttamente utilizzabili dai nodi di *motion planning*. Nelle prove sperimentali condotte, il robot *TIAGo* è stato inizializzato esattamente nell'origine della simulazione (coordinate X=0, Y=0, Z=0 e rotazioni nulle); di conseguenza, il sistema di riferimento locale del robot e quello globale del simulatore risultano perfettamente coincidenti. Questa configurazione strategica ha permesso di confrontare in modo diretto le coordinate dell'oggetto stimate dal nodo visivo con il *Ground Truth* della posizione in Gazebo.

Sono stati effettuati due diversi test per ogni oggetto scelto, posizionando l'oggetto *target* in due posizioni diverse nel piano di lavoro, i cui risultati sono presentati nella *Tabella 4.1*:

Test	***Oggetto target***	***Ground Truth stima dimensionale (cm)***	***Diametro/ spessore stimato (cm)***	***Altezza stimata (cm)***	***Baricentro (m)***	***Ground Truth baricentro (m)***
1	Cilindro (lattina *cocacola*)	Diametro = 8 Altezza = 15	6.35	12.15	X = 0.619 Y = 0.011 Z = 0.936	X = 0.627 Y = 0.000 Z = 0.815
2	Cilindro (lattina *cocacola*)	Diametro = 8 Altezza = 15	6.44	12.47	X = 0.665 Y = 0.012 Z = 0.928	X = 0.674 Y = 0.000 Z = 0.815
3	Sfera (pallina)	Diametro = 6	6.15	0	X = 0.818 Y = - 0.035 Z = 0.856	X = 0.815 Y = - 0.055 Z= 0.847
4	Sfera (pallina)	Diametro = 6	6.1	0	X = 0.748 Y = 0.027 Z = 0.860	X = 0.742 Y = 0.017 Z = 0.847

Tabella 4.1 Risultati ottenuti dalle prove di validazione del nodo di visione condotte sui due oggetti target disponibili nel sistema (cilindro e sfera). Dove x corrisponde alla misura di profondità (distanza oggetto-base_footprint), y corrisponde all'asse orizzontale (escursione laterale) e z identifica l'asse verticale (altezza). Il sistema di riferimento risiede in base_footprint.

Le prove condotte hanno evidenziato comportamenti differenti nella stima volumetrica in base alla geometria dell'oggetto *target*. Per quanto riguarda il modello cilindrico (lattina di *Coca Cola*), si conferma una sistematica sottostima del volume: estraendo la media dalle iterazioni, il diametro stimato si attesta a circa 6.40 cm (con un errore del 20% rispetto agli 8 cm reali), mentre l'altezza è valutata a circa 12.30 cm (rispetto ai 15 cm

nominali). Al contrario, le misurazioni effettuate sul modello sferico (pallina, *Figura 4.5*) si sono rivelate estremamente accurate, con un diametro stimato compreso tra 6.10 e 6.15 cm a fronte dei 6 cm reali (un errore marginale compreso tra l'1.6% e il 2.5%).

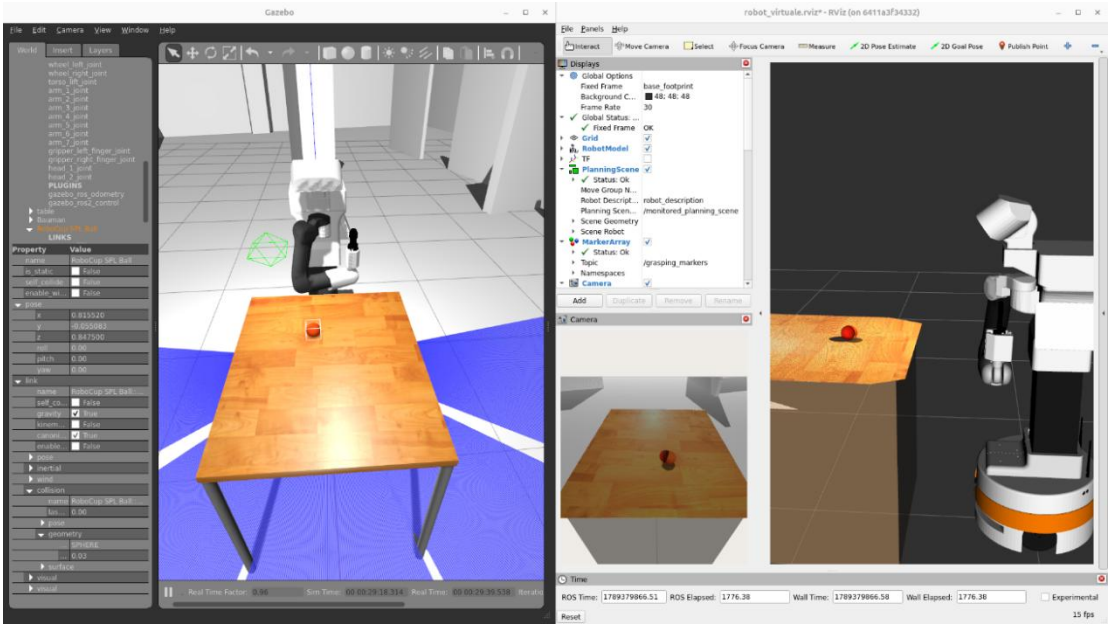


Figura 4.5 Test di validazione del nodo di visione con oggetto target sferico. A sinistra osserviamo la scena del software Gazebo, mentre a destra abbiamo la rappresentazione in RViz.

La discrepanza dimensionale osservata per il cilindro non è da imputarsi alla logica dell'algoritmo di visione, bensì ai limiti intrinseci dell'ambiente di simulazione stesso. I plugin di *Gazebo* impiegati per emulare i sensori di profondità presentano inevitabili approssimazioni rispetto a uno scenario reale. La generazione virtuale della *depth map* introduce spesso artefatti di *rendering* e rumore simulato sui contorni (specie su geometrie cilindriche). Questi limiti del motore di *rendering* alterano la resa volumetrica del modello, generando una *Point Cloud* meno fedele rispetto a quella ottenibile da un sensore fisico reale e inducendo l'algoritmo a sottostimare il volume dell'oggetto.

Nelle prove con la sfera, l'errore sulla stima della profondità (asse X) è risultato pressoché nullo (inferiore a 1.3 cm). Nei test con il cilindro (lattina), invece, si è registrato uno scostamento apparente di circa 11/12 cm. Tale deviazione non è tuttavia imputabile a un errore di stima, bensì al sistema di coordinate originario di *Gazebo*: l'origine (0,0,0) del modello virtuale della lattina è situata alla sua base (piano di appoggio) e non al centro di massa. Poiché l'algoritmo mira correttamente al centro fisico del volume per ottimizzare la presa del *gripper*, la discrepanza rilevata riflette fedelmente la distanza tra la base dell'oggetto e l'effettivo punto di *grasping* calcolato dal sistema visivo.

La reale robustezza del sistema è infine certificata da un *Detection Rate* perfetto (100% di rilevamento su 100 frame utili per ciascuna prova) e da un'elevatissima stabilità del dato spaziale, con oscillazioni (*Jitter*) misurate in appena 0.003 cm. Questi risultati sub centimetrici confermano che i *target* spaziali calcolati dalla *pipeline* sono altamente affidabili per alimentare con successo la successiva manovra di presa, dimostrando la capacità del nodo di controllo di operare con precisione nonostante le imperfezioni e i limiti tipici dell'ambiente di simulazione.

4.3 Sviluppo E Collaudo Del Nodo Di Pianificazione E Grasping

Il nodo ROS 2 *tiago_pick_place* rappresenta il modulo esecutivo per la manipolazione degli oggetti target e funge da ponte tra l'output della *pipeline* di visione (*tiago_camera_node*) e il *framework* di controllo del movimento *MoveIt 2*. Nello specifico, il nodo sottoscrive i *topic* contenenti il centro e le dimensioni del piano di lavoro *(/table_center, /table_dimensions*) e del *target* (*/object_centroid, /object_dimensions*). Per garantire la stabilità delle stime visive, i centroidi ricevuti vengono accumulati in un *buffer* circolare di 8 campioni: la posa effettiva viene calcolata come mediana delle letture e validata solo se la deviazione massima risulta inferiore a 1 cm (*target_stability_threshold* = 0.01). Ottenute le dimensioni, il nodo esegue una classificazione geometrica dell'oggetto (*determine_object_shape*), distinguendo tra sfera, cilindro e parallelepipedo. Utilizzando questi dati, aggiorna dinamicamente la *planning* scene tridimensionale, istanziando il tavolo e il *target* tramite primitive geometriche *MoveIt* 2 (*SolidPrimitive*) con margini di sicurezza di circa 5 mm per prevenire collisioni indesiderate. Una volta validato il *target*, il nodo effettua una sequenza automatizzata di *pick and place*: interroga i server cinematici per verificare la fattibilità della posa di pre-presa (*Inverse Kinematics*), esegue un avvicinamento cartesiano lineare (*GetCartesianPath*) lungo l'asse di approccio locale dell'*end effector*, gestisce la chiusura in retroazione del *gripper* e governa le successive fasi di sollevamento (*lift*), riposizionamento (*place*) e ritorno in configurazione di *home*. La logica per la presa dell'oggetto (*grasping*) è implementata riducendo gradualmente l'apertura delle dita (con una chiusura a piccoli incrementi di 2 mm alla volta), e confrontando, a ogni passo, l'apertura desiderata con quella reale letta dai giunti. Quando l'errore tra questi due valori supera una specifica soglia di tolleranza, il sistema deduce che l'oggetto sia bloccato fisicamente e la presa sia conclusa, per cui arresta istantaneamente il movimento. Inoltre, il nodo integra un sistema di tracciamento delle prestazioni della pipeline che esporta in un file CSV l'esito di ogni singola sottofase operativa.

4.3.1 Trasformazioni Spaziali E Vincoli Di Orientamento: I Quaternioni

La corretta localizzazione spaziale e l'allineamento dell'*end-effector* rappresentano il fondamento della sequenza di manipolazione. Il nodo di *grasping* sfrutta la libreria *TF2* di *ROS* per mappare dinamicamente le coordinate del baricentro dell'oggetto, nel sistema globale del robot (*base_footprint*). L'orientamento della pinza varia in base alla forma dell'oggetto, calcolando un quaternione d'assetto dedicato:

- Oggetti parallelepipedi e piatti: Viene adottata una strategia di presa dall'alto (top-down) imponendo un *pitch* nominale di $90°$. Per facilitare il raggiungimento della posa, il sistema di *check IK* analizza gradi di libertà aggiuntivi testando campionamenti discreti di *roll* ($\pm 90°, \pm 45°$).
- Oggetti sferici: Onde evitare un approccio puramente verticale (rigido e che rischierebbe impatti contro la superficie), il nodo impone una presa diagonale. Il *pitch* viene valutato a partire da un set di candidati decrescenti ($60°, 50°, 40°, 30°$): il solutore assegna la prima inclinazione utile che garantisca una soluzione *IK* priva di collisioni. In aggiunta, la verifica cinematica esegue una ricerca estesa su vari angoli di *roll* (da $\pm 17°$ fino a $\pm 180°$).
- Oggetti cilindrici: L'approccio avviene sul piano orizzontale con *pitch* nullo (0 rad) e *roll* nullo, mantenendo l'asse della pinza parallelo al piano di lavoro.

Per tutte le geometrie, lo *yaw* di approccio viene calcolato dinamicamente lungo l'asse radiale, ovvero l'asse che collega idealmente *end effector* del robot e oggetto. Tuttavia, se l'orientamento radiale risulta troppo frontale ($|\text{Yaw}| < 20°$), il sistema forza un angolo minimo di attacco laterale ($\text{MIN_LATERAL_YAW} = 20°$). Questa correzione previene collisioni frontali del braccio contro il bordo del tavolo o ostacoli adiacenti [48], [49].

4.3.2 Approccio Cartesiano Lineare (Linear Approach)

A differenza dei movimenti di trasferimento liberi, che vengono calcolati nello spazio dei giunti e possono generare traiettorie ad arco imprevedibili, la delicata fase di avvicinamento finale all'oggetto è gestita tramite una interpolazione rigorosamente cartesiana. Sfruttando il servizio */compute_cartesian_path*, il nodo impone alla pinza di coprire gli ultimi centimetri di discesa (definiti dal *target_z_offset*) seguendo una linea retta perfetta lungo l'asse Z, con un passo di campionamento massimo di 5 mm (*max_step* = 0.005). Questa scelta algoritmica impedisce movimenti laterali parassiti che potrebbero urtare l'oggetto. A garanzia della sicurezza operativa, il codice valuta la frazione della traiettoria calcolata (*result.fraction*): se il solutore non riesce a interpolare almeno il 95% del percorso lineare in totale assenza di collisioni, l'operazione viene abortita in sicurezza, prevenendo danni *hardware* al robot reale [49].

4.3.3 Compensazione Cinematica: Applicazione Degli Offset All'end Effector

Per tradurre il baricentro visivo in un punto di presa fisicamente raggiungibile, il nodo implementa una logica di compensazione tramite *offset* spaziali, che tengano conto dell'ingombro dato dalla pinza del robot e soprattutto dal fatto che il sistema di riferimento dell'*end effector* è situato nel "polso" robotico (*gripper_grasping_frame*) e non all'interno del "palmo" della mano, perciò se non applicassimo un *offset* porteremmo il robot ad approcciare l'oggetto sbattendo con il "polso" nella superfice esterna dello stesso. Nella funzione *build_target_pose*, viene applicato un offset traslazionale pari a *GRIPPER_CENTER_OFFSET* = 0.05 m, ovvero 5 cm lungo l'asse X locale dell'end-effector. Questo scostamento locale viene convertito nel frame globale *base_footprint* moltiplicandolo per la matrice di rotazione associata al quaternione d'orientamento della pinza (*apply_gripper_offset*).

Per gli oggetti caratterizzati da geometria parallelepipeda o da un profilo piatto, l'algoritmo applica correzioni mirate sia sul piano orizzontale sia lungo l'asse verticale per compensare i limiti intrinseci della percezione visiva e prevenire impatti indesiderati. In particolare, per bilanciare il sistematico errore di stima della profondità tipico dei sensori *RGB-D* nell'individuazione di superfici piatte o poco riflettenti, viene introdotto un *offset* radiale negativo (BOX_RADIAL_OFFSET $= -0.03$ m) che richiama il punto di attacco di 3 cm verso la base del robot. Contestualmente, la quota Z del *target* viene elevata di 3 cm rispetto al baricentro visivo stimato; tale accorgimento garantisce una quota di sicurezza anti-impatto, evitando che le estremità della pinza collidano con il piano d'appoggio o urtino la superficie durante la chiusura verticale (*top-down*).

Nel caso delle geometrie sferiche, le correzioni rispondono invece all'esigenza di centrare perfettamente l'inviluppo delle dita sulla curvatura dell'oggetto, applicando uno scostamento radiale (BALL_RADIAL_OFFSET $= -0.04$ m) e un micro-sollevamento verticale (BALL_Z_OFFSET $= 0.002$ m).

Il nodo, infine, integra un meccanismo di protezione dal piano di lavoro (TABLE_CLEARANCE), indispensabile nelle manovre d'approccio inclinate o dall'alto (Pitch > 0), dove la proiezione verso il basso dell'*end effector* rischia di danneggiare la struttura del piano d'appoggio. Il sistema calcola analiticamente la quota minima di sicurezza ($Z_{\min}$) sommando all'altezza del tavolo (Z_{table}) il franco libero desiderato (TABLE_CLEARANCE $= 0.015$ m) e la componente verticale dell'ingombro delle dita, determinata trigonometricamente come FINGER_TIP_LENGTH $\cdot \sin(\text{pitch})$ con FINGER_TIP_LENGTH $= 0.05$ m. Qualora la quota di presa target risulti inferiore a $Z_{\min}$, il codice rialza automaticamente il punto di mira saturandolo alla quota soglia calcolata, garantendo l'integrità meccanica della pinza e azzerando il rischio di collisione con il tavolo.

4.3.4 *Risultati Ottenuti Dai Test Di Simulazione Per Il Nodo Di Grasping*

L'intera procedura di manipolazione è stata testata con successo all'interno dell'ambiente simulato utilizzando diversi oggetti target (con geometria cilindrica, sferica o parallelepipeda). Una volta acquisita e stabilizzata la posizione del *target* da parte del modulo di visione, il sistema ha eseguito autonomamente tutte le fasi operative del *task*. Inizialmente, il pianificatore ha calcolato la cinematica inversa (*IK*) per portare il braccio robotico in una posa sicura di *pre-grasp*, tenendo conto degli ingombri virtuali rappresentati dal piano di lavoro e dall'oggetto stesso, per evitare collisioni con il tavolo e scontri indesiderati con l'oggetto. Successivamente, *TIAGo* ha effettuato un avvicinamento cartesiano lineare e preciso verso l'oggetto, culminato con la chiusura adattiva del *gripper* fino al rilevamento del contatto fisico con il *target* (*Figura 4.6*).

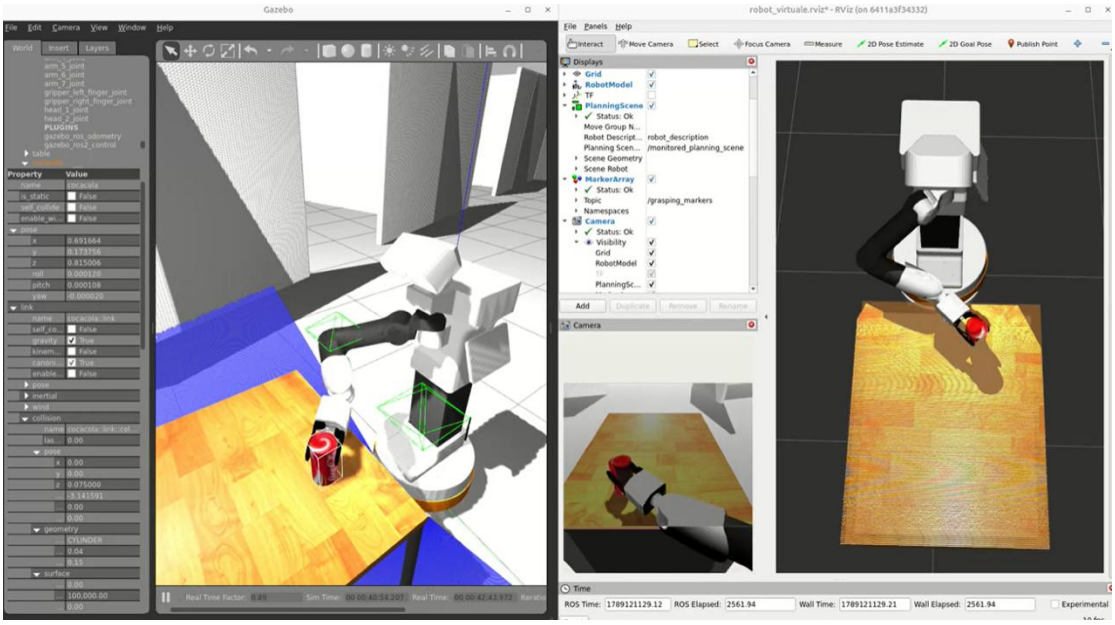


Figura 4.6 Fase di grasping all'interno dell'ambiente simulato. Il gripper di TIAGo esegue la chiusura adattiva attorno all'oggetto target cilindrico fino al rilevamento del contatto fisico. A destra (in RViz) è visibile la corretta pianificazione spaziale, mentre a sinistra (in Gazebo) avviene l'effettiva interazione fisica.

Infine, l'oggetto è stato sollevato verticalmente dal piano d'appoggio, dimostrando la perfetta sincronizzazione tra le coordinate stimate dal nodo visivo e i comandi di *motion planning,* per poi essere ripoggiato nella posizione di partenza nel piano di lavoro e terminando la sequenza di *pick and place*. Il task è stato portato a termine da TIAGo senza errori, confermando la solidità dell'integrazione tra percezione e controllo motorio.

4.4 Risultati Ottenuti Dalla Validazione Sperimentale Della Pipeline In Gazebo

Alla luce dei risultati ottenuti, l'integrazione tra il modulo di percezione visiva e il nodo di *grasping* ha superato con successo la fase di validazione in ambiente simulato. La precisione sub centimetrica garantita dalla stima spaziale ha permesso a *MoveIt 2* di calcolare traiettorie cinematiche sicure, portando a termine il *task* di *Pick & Place* in modo autonomo e ripetibile. Avendo confermato la solidità dell'intera architettura *software* su *Gazebo*, il passo successivo consisterà nel collaudo dell'intero sistema sul robot *TIAGo* fisico presente nel *Living Lab* del Centro di Eccellenza *REDI* dell'Università di Pavia.

Capitolo Cinque. Validazione Sperimentale E Traslazione Della Pipeline Su Robot Fisico In Living Lab

Nel seguente capitolo verranno descritte e discusse le prove sperimentali eseguite nel *Living Lab* del Centro di Eccellenza *REDI* dell'università di Pavia, utilizzando il robot *TIAGo* disponibile nel laboratorio. Sono state eseguite delle prime prove volte alla validazione sperimentale del nodo di visione realizzato in *ROS 2*, utilizzando il medesimo set di dieci oggetti sfruttato anche nelle prove con la telecamera *Orbbec Astra* in configurazione s*tand alone* precedentemente eseguite. Successivamente alle prove del nodo di visione, sono state eseguite le prove del nodo di *grasping*. Il passaggio da simulazione a test reali su robot fisico è stato dettato dalla trasposizione dei codici utilizzati in simulazione ai nuovi codici adoperati in "vivo", che ha previsto la modifica di alcuni parametri fondamentali, tra cui i parametri intrinseci della telecamera del robot e i *topic* specifici per la comunicazione tra i nodi.

Il passaggio dalla simulazione alla realtà rappresenta una criticità intrinseca nello sviluppo di sistemi robotici, in quanto molti approcci si basano su presupposti ideali che faticano a tradursi in applicazioni pratiche. Tra le principali discrepanze figurano l'assenza o la limitazione di sistemi di posizionamento globale (*GPS*), le restrizioni nei canali di comunicazione inter-robot e la mancanza di informazioni preventive sull'ambiente o sui *target*. Inoltre, l'utilizzo di ambienti simulati semplificati o privi di ostacoli non riflette le complessità e i rischi dei contesti reali, rendendo indispensabili ulteriori studi sul campo per adattare gli algoritmi alla navigazione basata su sensori e a condizioni operative complesse. Questo divario, noto come *reality gap*, è definito formalmente come la discrepanza tra l'ambiente simulato e quello reale in termini di dinamica e percezione. Poiché i simulatori si affidano ad approssimazioni, astrazioni numeriche e modelli fisici semplificati, come l'assunzione di corpi perfettamente rigidi o contatti ideali, non riescono a replicare pienamente la complessità e la stocasticità del mondo fisico. Tale divario si articola principalmente in quattro componenti critiche: le differenze nella dinamica e nei parametri fisici, le limitazioni percettive dei modelli sensoriali (che spesso ignorano rumori complessi e artefatti), le discrepanze negli attuatori e nei sistemi di controllo di basso livello, e i fattori legati al *design* di sistema come latenze di comunicazione e meccanismi di sicurezza. Superare questo disallineamento richiede strategie mirate, quali il miglioramento dell'identificazione del sistema, l'uso di modelli residuali o la *domain randomization*, al fine di garantire che le politiche addestrate in ambiente virtuale mantengano robustezza ed efficacia nella loro applicazione nel mondo reale [50], [51].

5.1 Validazione Sperimentale del Nodo di Visione: Protocollo Di Acquisizione

Allo scopo di garantire la ripetibilità degli esperimenti e una valutazione rigorosa delle *performance* ottenibili dalla *pipeline*, l'area di lavoro è stata opportunamente discretizzata definendo una griglia matriciale di test sulla superficie del piano di lavoro, identificando le seguenti coordinate spaziali, in cui verranno posizionati gli oggetti, uno alla volta, per poter effettuare la validazione del nodo di visione tramite il robot *TIAGo* (anche osservabili nella *Figura 5.1)*:

- Inner Center: (55, 0)
- Mid Center: (70, 0)
- Outer Center: (85, 0)
- Extreme Center: (100, 0)
- Near Right: (62, -20)
- Mid Right: (70, -30)
- Far Right: (78, -20)
- Near Left: (62, +20)
- Mid Left: (70, +30)
- Far Left: (78, +20)

Le posizioni sul piano di lavoro sono state scelte in modo da esplorare la capacità fisica operativa di *TIAGo* [25], con particolare attenzione alla massima estensione effettuabile da parte dell'*end effector* (*Figura 5.2*). In particolare, dunque, per l'asse X sono presenti la posizione massima di 100 cm, prossima alla massima distanza raggiungibile dal robot (lunghezza del braccio di *TIAGo* 86 cm circa e del *gripper* 10 cm), la posizione minima di 55 cm, scelta come distanza minima che *TIAGo* deve mantenere dal piano di lavoro per poter eseguire il movimento in modo corretto e senza collisioni con tavolo e/o oggetto, e diverse posizioni che esplorano anche lateralmente lo spazio disponibile per i *task* di *pick and place*.

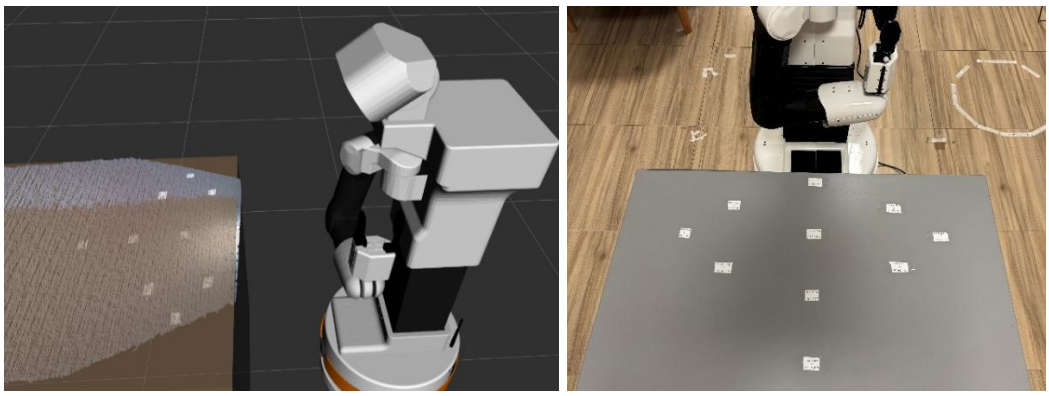


Figura 5.1 Visualizzazione delle posizioni scelte nel piano di lavoro contrassegnate da marker per la validazione del nodo di visione.

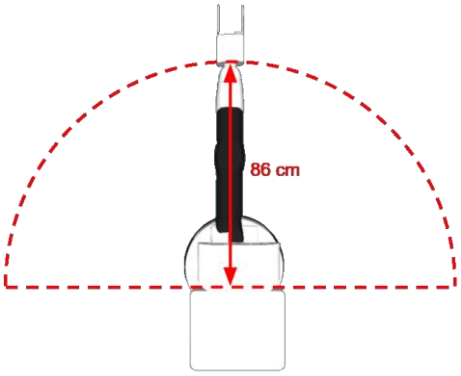


Figura 5.2 Massima escursione possibile del braccio manipolatore di TIAGo.

5.1.1 Configurazione Cinematica E Assetto Dei Sensori

Durante le fasi preliminari dei test di visione, si è constatato come l'orientamento della testa di *TIAGo* rappresenti un parametro cinematico critico per il corretto svolgimento del *task*. Mantenendo la telecamera in una posa fissa e puramente frontale, il limitato campo visivo (*Field of View*) del sensore genera delle zone cieche, precludendo il rilevamento degli oggetti collocati alle estremità del piano di lavoro. Per ovviare a questa limitazione, si è adottata una strategia di percezione attiva: l'articolazione del collo (*pan joint*) viene leggermente ruotata verso destra o verso sinistra in funzione della posizione stimata dell'oggetto *target*, garantendo così un'inquadratura ottimale e una copertura completa dell'area operativa. Nello specifico, per implementare questa strategia di inquadratura dinamica, l'elevazione del giunto prismatico del busto (*torso_lift_joint*) è stata mantenuta fissa a 0.35 m, garantendo un'altezza operativa ottimale rispetto al piano di lavoro. Per quanto riguarda l'orientamento della telecamera, i movimenti del collo di *TIAGo* sono stati scomposti nei due gradi di libertà fondamentali. L'angolo di inclinazione verticale (noto in cinematica come *pitch*), controllato dal giunto *head_2_joint*, è stato mantenuto costante per tutte le acquisizioni a -0.65 rad (corrispondenti a un'inclinazione verso il

basso di circa 37°). Al contrario, l'angolo rotazione laterale (definito come *yaw*), gestito dal giunto *head_1_joint*, è stato modulato in tre configurazioni predefinite in base alla posizione dell'oggetto sul tavolo: 0.35 rad (circa 20°) per l'esplorazione dell'emispazio sinistro, -0.35 rad (circa -20°) per l'emispazio destro e 0 rad (0°) per il puntamento frontale dell'area centrale (*Figura 5.3*).

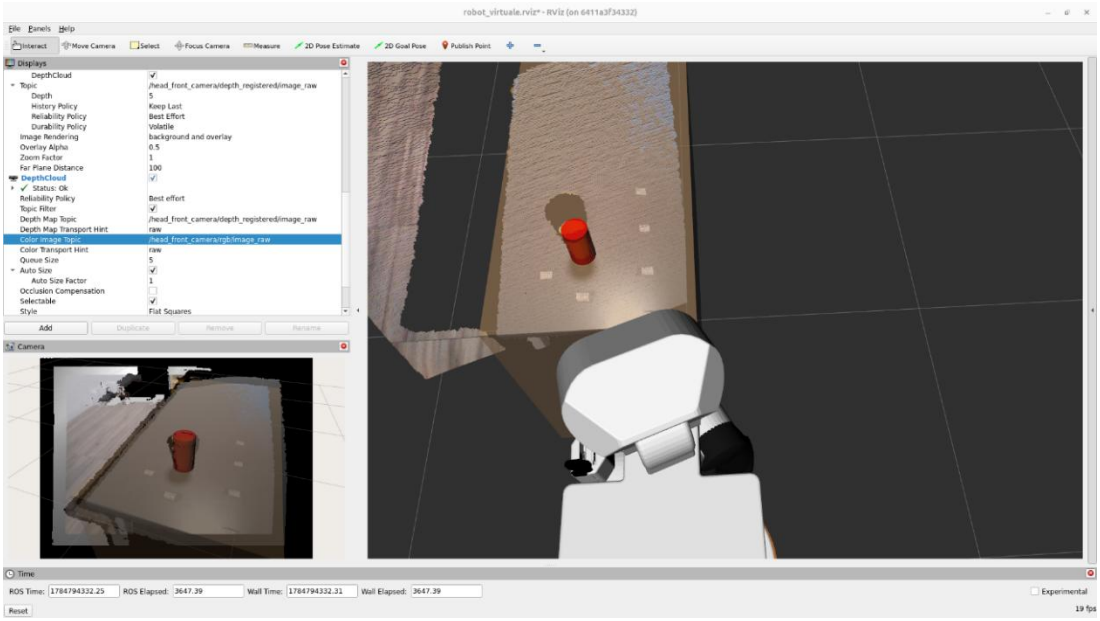


Figura 5.3 Esempio di esecuzione del test per il nodo di visione con inquadratura sulla sinistra: visualizzazione dell'output su RViz, in cui si evince il corretto posizionamento del marker dell'oggetto nello spazio operativo.

Nella *Figura 5.3* possiamo osservare che quando *TIAGo* mantiene un'inquadratura puramente frontale, il piano di lavoro rientra interamente all'interno del campo visivo del sensore, e al contrario, ruotando la testa verso i quadranti laterali (destra o sinistra), l'inquadratura subisce un taglio prospettico che riduce la porzione di tavolo percepita, focalizzando la vista esclusivamente sulla zona *target*. Questa perdita parziale del piano di lavoro non rappresenta tuttavia una criticità operativa, in quanto l'ingombro del tavolo viene comunque istanziato come oggetto di collisione fisso all'interno dell'ambiente *RViz*.

5.1.2 Metriche Di Validazione E Logica Di Campionamento

Al fine di garantire la significatività statistica delle misurazioni, il protocollo di acquisizione software è stato temporizzato e automatizzato. Per ogni singola posa di test, il nodo *ROS 2* di visione acquisisce un set predefinito di 100 frame validi, scartando i primi 50 cicli di esecuzione per consentire l'assestamento delle trasformazioni spaziali (fase di *warm-up,* processo anche descritto ed eseguito nel paragrafo dedicato alla simulazione). Al raggiungimento dei *frame target*, il sistema interrompe l'acquisizione e

genera automaticamente due report in formato *CSV*: un file dettagliato contenente i dati grezzi *frame per frame*, necessario per l'analisi del *jittering* spaziale, e un report riassuntivo. Quest'ultimo aggrega i dati calcolando le medie prestazionali, tra cui il *Detection Rate*, l'errore dimensionale relativo e le coordinate stimate del baricentro tridimensionale.

5.1.2.1 Errore Dimensionale Relativo

L'errore dimensionale relativo quantifica lo scostamento percentuale tra le dimensioni stimate dall'algoritmo e le dimensioni reali di *ground truth* note (larghezza, altezza o diametro). L'analisi di questo errore permette di evidenziare eventuali *Object Bias* (difficoltà di stima legate alla specifica morfologia o al colore dell'oggetto) e *Spatial Bias* (distorsioni ottiche e prospettiche all'aumentare della distanza dal sensore). Al fine di valutare l'errore dimensionale rispetto alle varie posizioni di lavoro scelte sul tavolo, sono stati realizzati dei grafici dedicati, suddivisi nelle tre grandezze stimate (diametro, altezza e larghezza), che riportano la percentuale di errore a seconda dei vari oggetti e della loro posizione nello spazio. L'impiego congiunto del *box plot* e dello *strip plot* offre una duplice prospettiva d'analisi: mentre il primo sintetizza la distribuzione statistica e la varianza dei dati per ogni posizione, il secondo permette di disaggregare l'informazione, evidenziando l'esatta collocazione delle singole rilevazioni. Attraverso una specifica codifica cromatica dei *marker*, è infatti possibile tracciare puntualmente le *performance* associate ai diversi oggetti *target*.

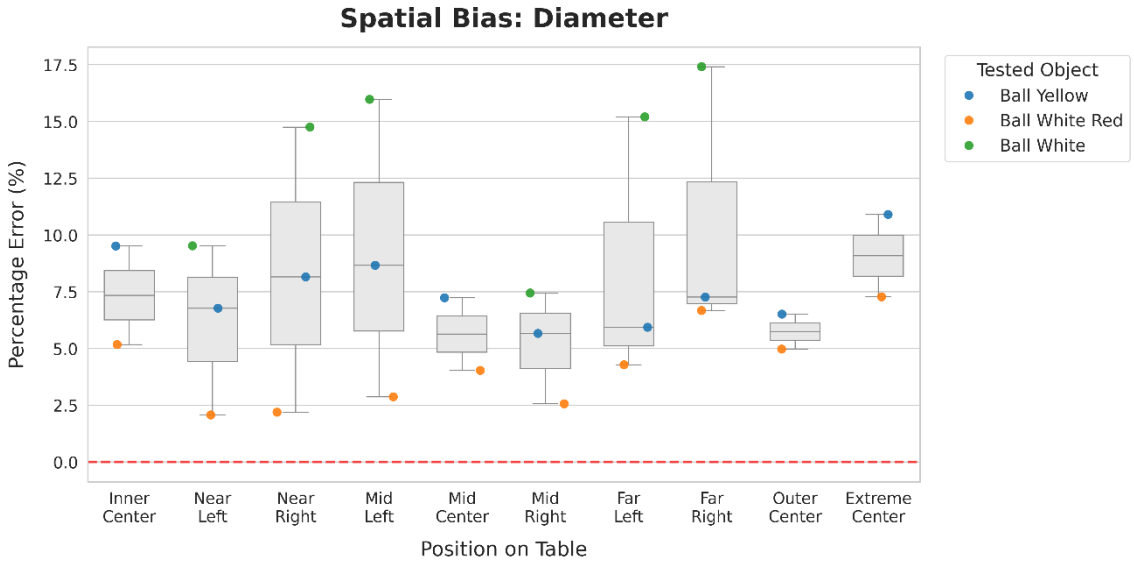


Figura 5.4 Errore percentuale dimensionale del diametro delle sfere rispetto alle varie posizioni sul piano di lavoro.

Osservando la linea rossa tratteggiata (che rappresenta l'errore nullo dello 0%), si osserva immediatamente che tutti i rilevamenti si posizionano al di sopra di essa, denotando un errore positivo e dunque una sovrastima dimensionale da parte del sistema, con un errore che va da un minimo di circa il 2% ad un picco massimo del 17.5%, per oggetti particolarmente complessi come la pallina bianca, che presenta poco contrasto con il piano di lavoro e di conseguenza un caso limite per il sistema. La problematica maggiore che emerge è quella legata al basso contrasto, da un'osservazione attenta dei *marker* colorati, possiamo infatti notare che, l'oggetto *Ball White* non solo rappresenta il *worst case scenario*, generando errori percentuali molto alti ma è anche l'unico oggetto che non viene registrato in alcune posizioni sul piano di lavoro. Questo particolare oggetto *target* (bianco) presenta un basso contrasto con il piano di lavoro e a causa di questa caratteristica e anche dell'illuminazione ambientale riflessa dal tavolo, non viene identificato in alcune posizioni sul piano di lavoro dalla rete *YOLO*, che non genera alcuna maschera (*bounding box*) (*Figura 5.5*).



Figura 5.5 Confronto tra pallina bianca (Ball White), sulla sinistra, e pallina gialla (Ball Yellow), sulla destra: possiamo osservare come, rispetto alla stessa posizione sul piano e la medesima illuminazione ambientale, YOLO identifichi correttamente la pallina gialla (grazie al suo contrasto rispetto al piano) e classifichi correttamente l'oggetto target (come sports ball), rispetto alla pallina bianca che, a causa dello scarso contrasto, non viene percepita dalla rete YOLO.

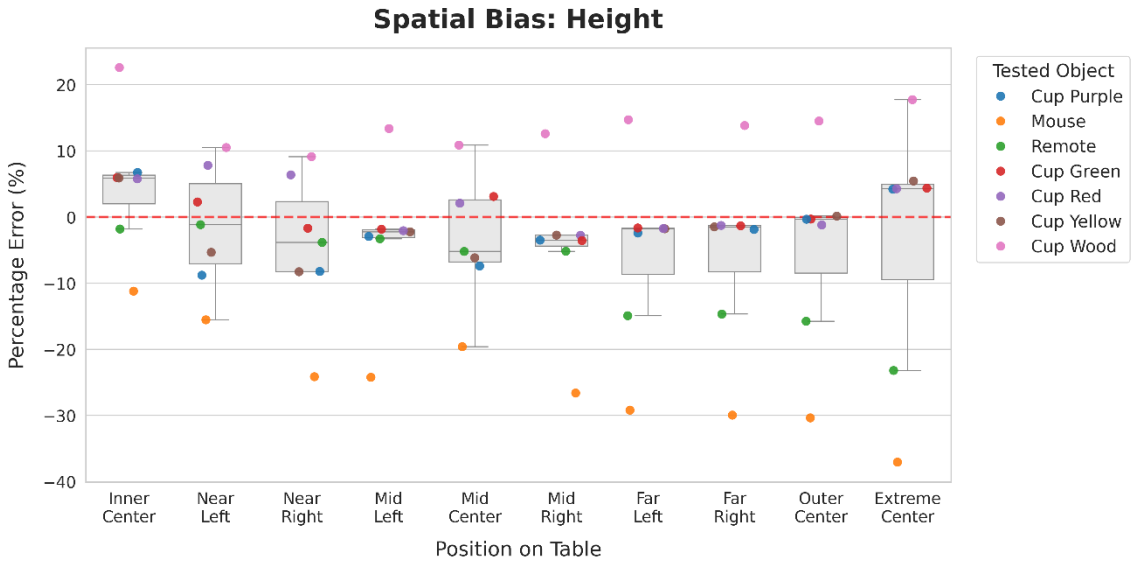


Figura 5.6 Errore percentuale dimensionale dell'altezza degli oggetti target (cilindri e parallelepipedi) rispetto alle varie posizioni sul piano di lavoro.

Rispetto all'analisi sul diametro, l'estrazione dell'altezza (*Figura 5.6*) presenta dinamiche profondamente diverse, in particolar modo influenzate dalla prospettiva della telecamera. Abbiamo infatti una tendenza generale alla sottostima: a differenza del diametro, che veniva sistematicamente sovrastimato, l'errore sull'altezza tende prevalentemente verso il basso (si concentra sotto la linea rossa dello 0% di errore). Questo fenomeno è legato all'angolo di incidenza visiva della telecamera posta sulla testa di *TIAGo*, che osservando il piano di lavoro dall'alto verso il basso genera una sottostima prospettica che la *PCA* fatica a seguire, rilevando un'altezza degli oggetti leggermente inferiore rispetto al *ground truth* fisico. Un altro fattore osservabile è l'elevato *Object bias* per le geometrie piatte, per cui alcuni oggetti subiscono un calo prestazionale dovuto alla loro specifica geometria. In particolare, oggetti come il *Mouse* (*marker* arancione) e il *Remote* (*marker* verde), che presentano una geometria complessa perché molto bassi e sottili, rappresentano i casi limite per il sistema di visione, che fatica a calcolare la loro altezza portano a errori dimensionali tra il -10% e il -40% nella zona centrale più a distanza dalla telecamera. Gli oggetti cilindrici mostrano una stabilità maggiore, con errori molto vicini alla linea dello 0% (in un range tollerabile tra il +5% e il -10%). Questi piccoli errori potrebbero derivare dalle peculiarità specifiche dei vari oggetti, come ad esempio la *texture*: per esempio il *Cup Wood*, è un oggetto cilindrico di legno bianco che può confondere la segmentazione di *YOLO* a causa del suo basso contrasto. Nelle posizioni più distanti, infine, si nota un leggero dilatamento delle *box* che denota maggiore varianza e un distacco netto tra le performance sui cilindri rispetto agli oggetti piatti.

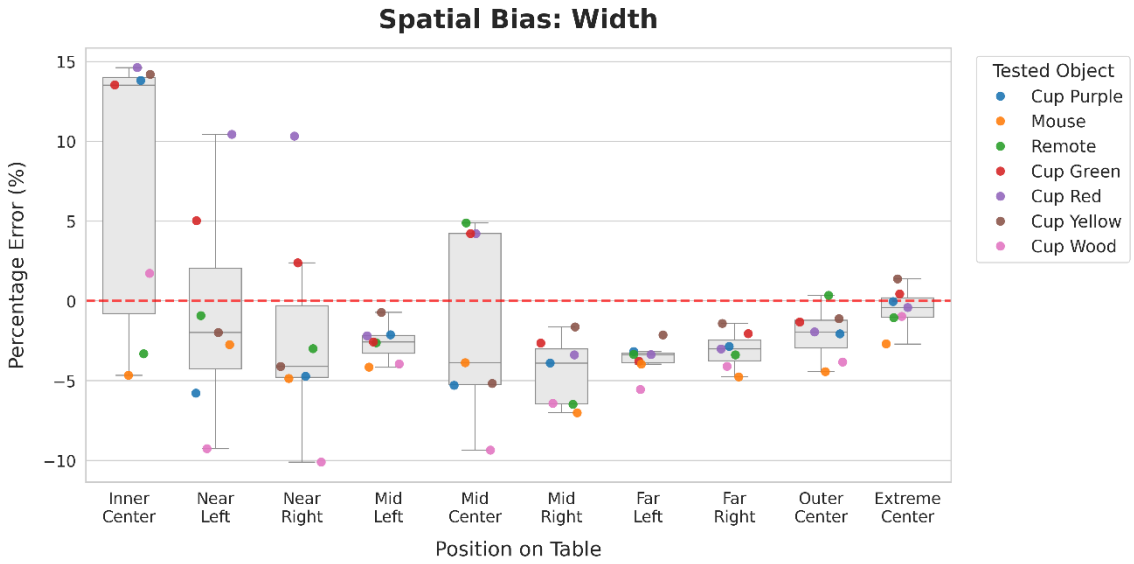


Figura 5.7 Errore percentuale dimensionale della larghezza degli oggetti target (cilindri e parallelepipedi) rispetto alle varie posizioni sul piano di lavoro.

Il grafico presentato nella *Figura 5.7* mostra l'errore percentuale nella stima della larghezza per i sette oggetti target cilindrici e/o parallelepipedi, valutati in precedenza rispetto all'altezza. Il dato più particolare riguarda le fasce più vicine al robot, in cui si verifica una distorsione focale della telecamera, infatti, quando un oggetto si trova molto vicino e in basso rispetto all'obiettivo, la proiezione prospettica fa percepire i contorni laterali molto più larghi di quanto non siano nella realtà. Perciò, la maschera generata da *YOLO* si allarga, portando la *PCA* a sovrastimare la larghezza. A partire dalla fascia intermedia, il comportamento del sistema cambia, infatti, possiamo osservare una diminuzione immediata della varianza con *box plot* più contenuti e un errore che si posiziona stabilmente tra il -2% e il -6%. Questo fenomeno avviene perché, allontanandosi dal sensore svanisce l'effetto "grandangolo", e l'oggetto viene visto in maniera migliore, ma allo stesso tempo, a lunghe distanze *YOLO* produce maschere leggermente conservatrici, portando a una sottostima strutturale della larghezza, dovuta al fatto che la risoluzione spaziale del sensore *RGB-D* diminuisce e i *pixel* della nuvola di punti si diradano.

In sintesi, l'analisi comparativa condotta sulle differenti geometrie evidenzia come l'accuratezza metrica della *pipeline* non dipenda unicamente dalla bontà degli algoritmi di proiezione tridimensionale, bensì dall'interazione tra le proprietà ottiche e geometriche del *target* e il punto di vista del sensore.

5.1.2.2 Stabilità Spaziale 3D: Jittering Del Centroide

Il *Jitter* spaziale quantifica il rumore intrinseco del sensore di profondità e l'oscillazione del *bounding box* generato dalla rete neurale. Matematicamente, è rappresentato dalla norma del vettore delle deviazioni standard delle coordinate (X, Y, Z) del baricentro, calcolate sull'intera finestra temporale di acquisizione (la formula è riportata nei paragrafi precedenti). In un'applicazione di manipolazione robotica governata da *MoveIt*, questa metrica è critica: un *jitter* elevato si traduce in un *target frame* instabile, che costringerebbe il pianificatore cinematico a ricalcolare continuamente le traiettorie del braccio, aumentando il rischio di collisioni con il piano di lavoro o prese fallite.

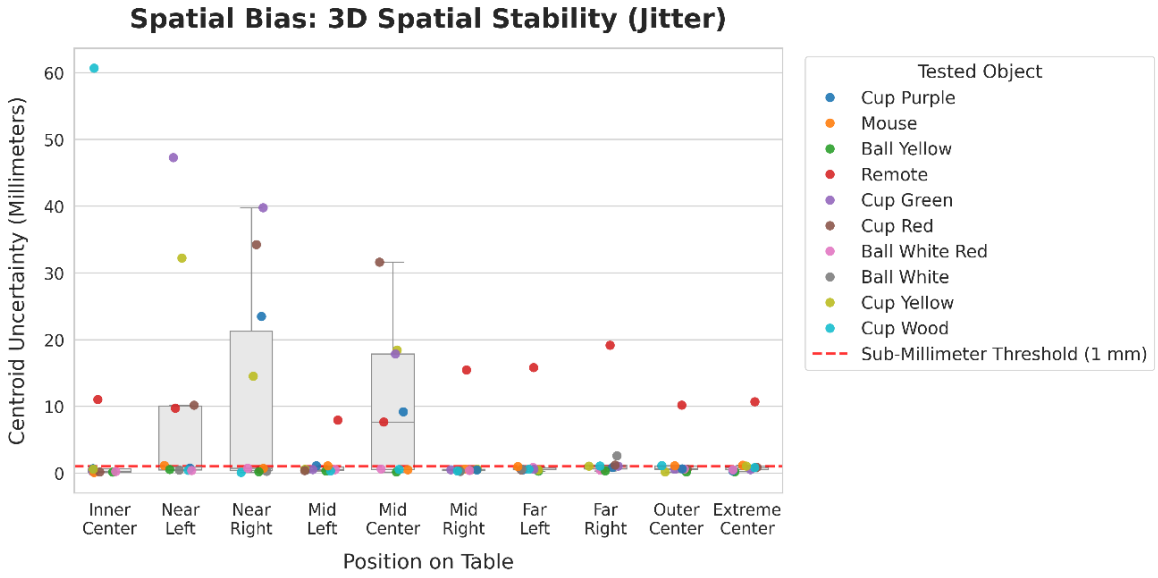


Figura 5.8 Stabilità del baricentro degli oggetti target al variare della loro posizione sul piano di lavoro.

Nella *Figura 5.8*, possiamo osservare il grafico relativo all'incertezza spaziale del baricentro 3D (misurata in mm) per tutti gli oggetti testati, mappata sulle 10 posizioni sul piano di lavoro. La linea rossa evidenzia la soglia sub millimetrica (1 mm) considerata ideale per una pianificazione cinematica sicura. Una prima attenta analisi del grafico evidenzia un'affidabilità sub millimetrica generalizzata: la maggior parte delle misurazioni si concentra al di sotto o in prossimità della soglia critica di 1 mm. D'altra parte, in alcune posizioni specifiche (*Near Left, Near Right e Mid Center*) emerge un comportamento anomalo sistematico: la conformazione liscia e curva di alcuni oggetti porta a riflessioni e/o deviazioni dell'infrarosso proiettato dal sensore, che genera delle "irregolarità" nella lettura della profondità che quindi subisce misurazioni con dati corrotti o intermittenti, con perdita della corretta lettura delle coordinate del baricentro tra i vari *frame*. Inoltre, da questa prima figura, possiamo osservare anche un

comportamento specifico legato all'oggetto *Remote*, identificato dal *marker* rosso: essendo un oggetto con un *low profile*, presenta una maggiore instabilità a causa dell'interazione dell'infrarosso con questa particolare morfologia, e di conseguenza è caratterizzato da un baricentro che oscilla continuamente tra i vari *frame*.

Per una visualizzazione migliore delle performance dei vari oggetti nelle posizioni in cui i *box plot* si attestano al di sotto della soglia massima di 1 mm, viene riportato anche il grafico in *Figura 5.9* che riporta uno "zoom" della visualizzazione precedente.

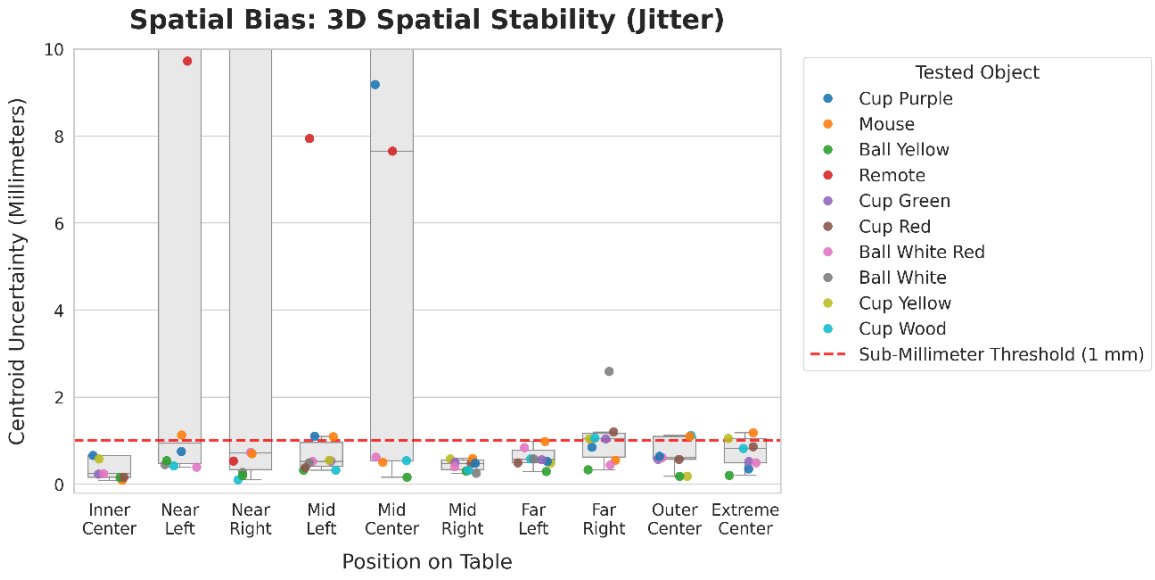


Figura 5.9 Stabilità del baricentro degli oggetti target al variare della loro posizione sul piano di lavoro, con zoom del grafico per mettere in evidenza i casi al di sotto della soglia scelta di 1 mm.

In questa nuova figura possiamo infatti apprezzare meglio la stabilità ottenuta dal sistema del nodo di visione, osservando la corretta localizzazione dei vari *box plot* al di sotto o in prossimità della soglia, che denota le ottime performance ottenute dal sistema implementato.

Al fine di osservare al meglio la stabilità del baricentro rispetto alle varie zone del piano di lavoro, sono state realizzate anche delle mappe bidimensionali, per rappresentare la localizzazione dei baricentri stimati dal sistema rispetto al *ground truth*.

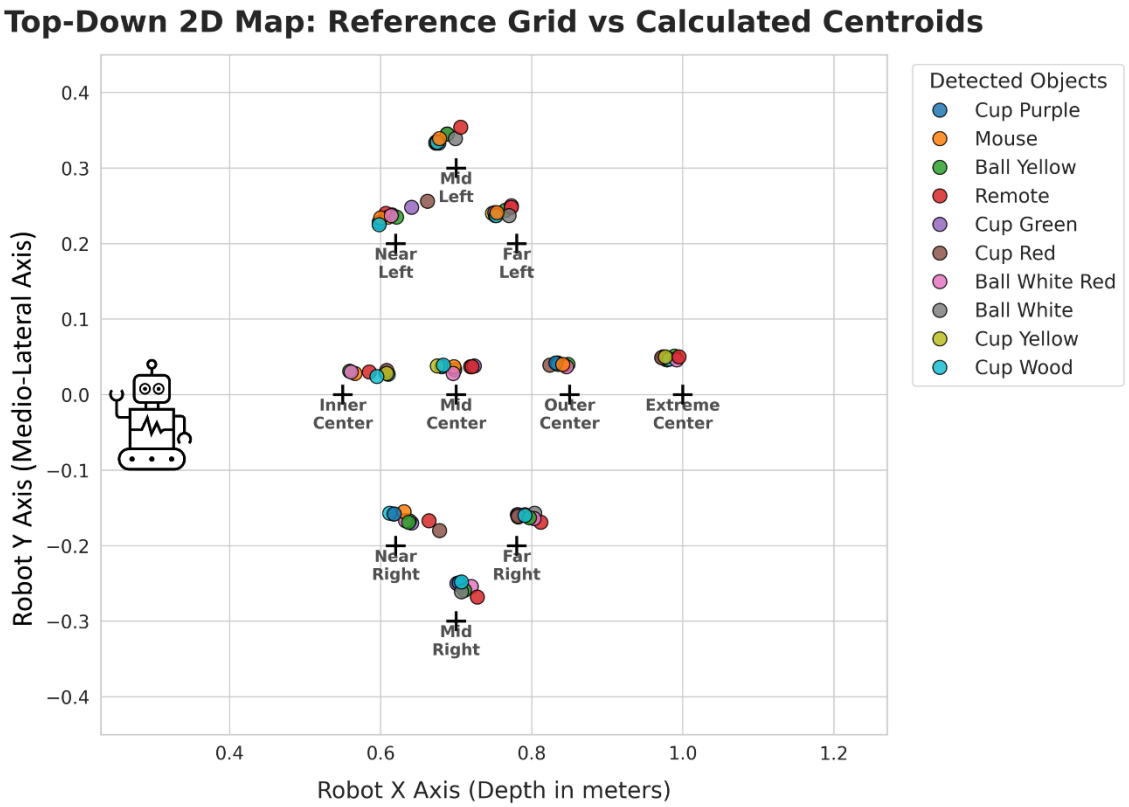


Figura 5.10 Mappa bidimensionale della localizzazione dei baricentri dei vari oggetti target identificati dal sistema di visione (apprezzabili tramite marker colorati), rispetto alle posizioni del ground truth identificate dai simboli "+" presenti nel grafico.

Nella *Figura 5.10* possiamo osservare i vari *marker* colorati corrispondenti ai 10 oggetti *target* e la loro localizzazione nel piano 2D stimata dal sistema di visione, e i simboli "+" identificano le posizioni scelte all'inizio della validazione e delle prove sperimentali, e dunque rappresentative del *ground truth* delle localizzazioni degli oggetti sul piano di lavoro. Si mette in evidenza che *TIAGo* sta sulla sinistra rispetto al grafico con lo sguardo rivolto come in *Figura 5.11:*

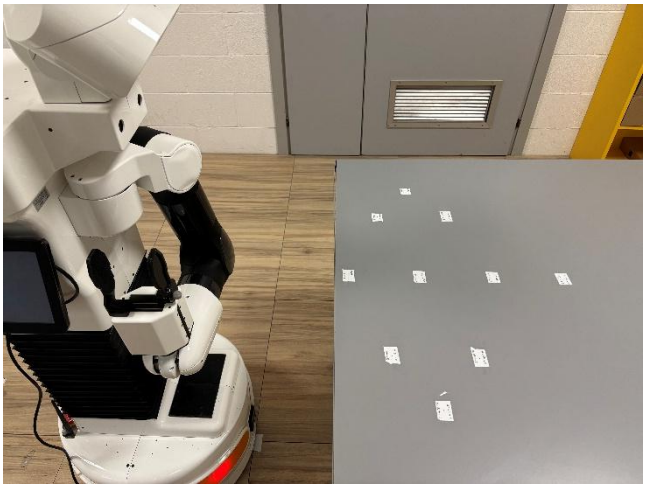


Figura 5.11 Visualizzazione della scena di riferimento con *TIAGo* e il piano di lavoro.

Dal grafico emerge in primo luogo, l'eccellente precisione e indipendenza dall'oggetto; infatti, il dato più incoraggiante è la presenza dei raggruppamenti (*cluster*) dei *marker* colorati che sono indice di un calcolo del baricentro da parte del sistema di visione, corretto e indipendente dall'oggetto. D'altra parte, osserviamo anche la presenza di un *bias* sistematico: mentre la precisione è altissima, poiché i *marker* sono molto vicini e a tratti sovrapposti tra loro, si nota una discrepanza nell'accuratezza; infatti, i *cluster* non sono perfettamente sovrapposti alle croci che identificano il *ground truth*. Questo fenomeno non è da imputare a un fallimento dell'algoritmo di visione, ma bensì ad un impercettibile errore di posizionamento del robot rispetto alle misure effettuate sul piano. Per determinare quale sia l'errore sistematico presente, calcoliamo gli errori di misura come la differenza tra il valore misurato e quello reale (*ground truth*), in modo da ottenere l'errore grezzo per ogni singola misura, compreso di segno:

$$Errore_i = Baricentro_{stimato} - Baricentro_{reale}$$

Successivamente è stata calcolata la media aritmetica, in moto tale da osservare se si tratti di un errore casuale, per cui la media tenderebbe a zero, o di un errore sistematico, per cui la media avrebbe un valore maggiore di zero.

A seguito dell'applicazione della formula:

$$Bias_{sistematico} = \frac{1}{N}\sum_{i=1}^{N} Errore_i$$

notiamo che l'errore in *output* è di natura sistematica, e viene stimato positivo rispetto all'asse Y e pari a circa 4 cm, per ogni oggetto *target*. Viene dunque applicata la seguente

formula, per la correzione dell'errore, andando a eliminare l'errore sistematico dalla misura stimata dal sistema:

$$Baricentro_{Stimato\ Corretto} = \frac{1}{N}\sum_{i=1}^{N} |Baricentro_{Stimato} - Bias|$$

Dopo aver corretto le misure eliminando l'errore sistematico di circa 4 cm, viene ricalcolato l'errore che dunque in questo caso rappresenta l'errore reale compiuto dal sistema:

$$Errore_{Stimato\ Corretto} = \frac{1}{N}\sum_{i=1}^{N} |Baricentro_{Stimato\ corretto} - Baricentro_{reale}|$$

I risultati dei calcoli eseguiti sono riportati nella *Tabella 5.1*:

Asse di riferimento	***Fase iniziale (con bias)***	***Fase finale (con correzione misure)***
Offset sistematico asse X	+ 0.41 mm	+ 0.41 mm
Offset sistematico asse Y	+ 39.12 mm	+ 5.74 mm

Tabella 5.1 Risultati ottenuti da calcolo degli errori di misura nella fase iniziale prima dell'eliminazione del bias sistematico e confronto con la fase finale dopo l'eliminazione del bias e la correzione delle misure.

Di seguito si riporta la mappa bidimensionale della localizzazione dei baricentri stimati rispetto al *ground truth*, non affetta da errori sistematici:

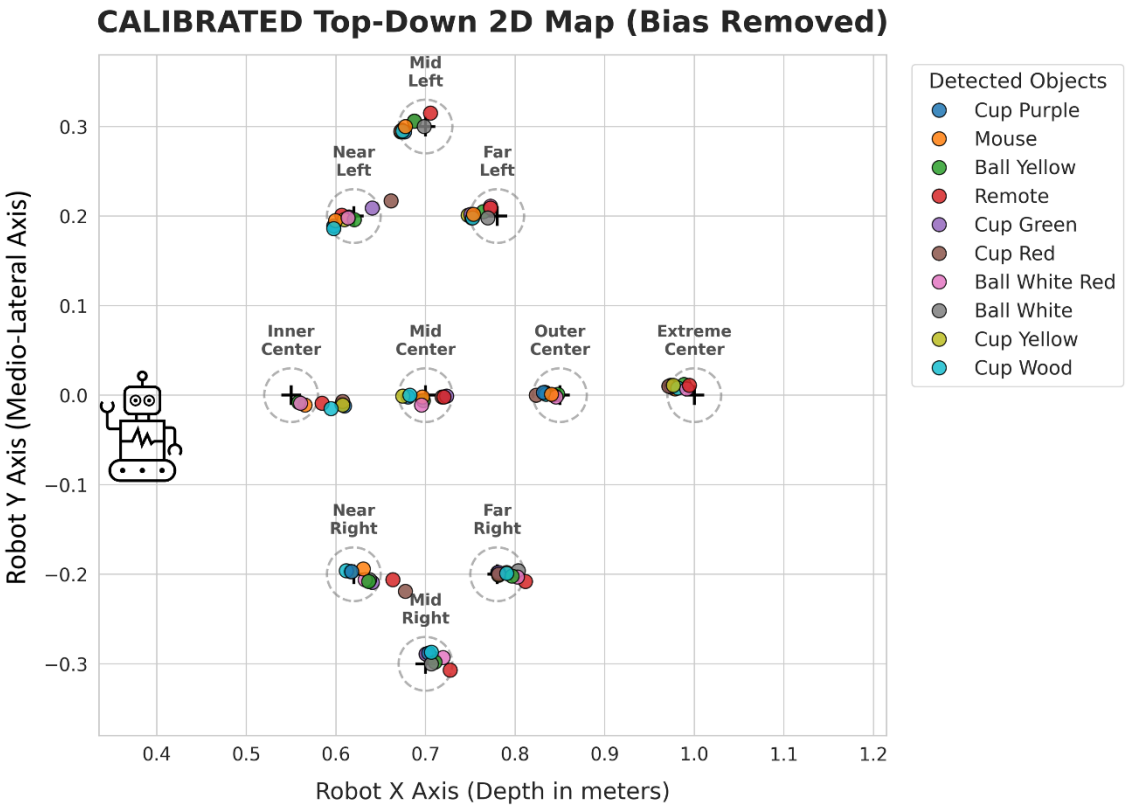


Figura 5.12 Mappa calibrata della localizzazione del baricentro degli oggetti sul piano di lavoro, non affetta da errore sistematico.

Una volta rimosso il *bias* costante, la mappa evidenzia un eccellente allineamento spaziale: le posizioni stimate dalla *pipeline* visiva (i *marker* colorati associati ai vari target) risultano densamente raggruppate attorno al *ground truth* geometrico (rappresentato dalle croci nere). La dispersione spaziale (*jittering*) è minima e le stime ricadono quasi interamente all'interno dei limiti di tolleranza (circonferenze tratteggiate, pari a diametro di 4 cm).

5.1.2.3 Robustezza Della Percezione: Detection Rate

Il *Detection Rate* valuta l'affidabilità semantica e geometrica della *pipeline* di visione ed è calcolato come il rapporto percentuale tra i *frame* in cui l'oggetto è stato correttamente individuato, e di conseguenza segmentato, e il numero totale di frame analizzati dal sensore per raggiungere il *target* di cento acquisizioni utili. Un tasso prossimo al 100% indica un riconoscimento istantaneo e continuo. Al contrario, un calo drastico di questa metrica (ad esempio al di sotto dell'80%) segnala l'incapacità del sistema di mantenere un *tracking* stabile, dovuta alla degradazione della risoluzione spaziale dell'oggetto nell'immagine *RGB*, a riflessi, o alla perdita di dati di profondità ai margini del campo visivo.

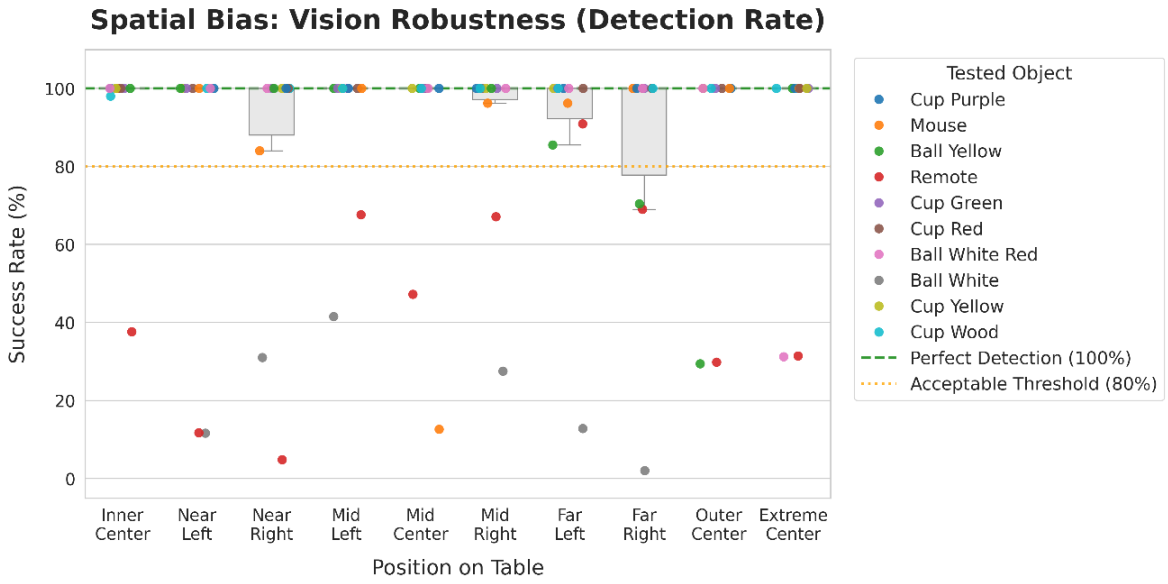


Figura 5.13. Grafico che mostra la percentuale di successo di identificazione dell'oggetto da parte della rete YOLO (detection rate), rispetto alle varie posizioni dell'oggetto stesso sul piano di lavoro.

Il dato visivamente predominante è la massiccia concentrazione di misurazioni esattamente sulla soglia del 100%. Per una larghissima maggioranza degli oggetti (in particolar modo le *cup* cilindriche) e nella maggior parte delle posizioni, la rete neurale *YOLO* non "perde" l'oggetto dalla sua "vista" tra i vari *frame*. Questo dimostra che, al netto di riflessi che inficiano il *Jitter* tridimensionale (come analizzato in precedenza), la segmentazione 2D di *YOLO* è precisa e costante, perciò, il sistema riesce a mantenere un'attenzione ininterrotta sui *target* solidi. In netta controtendenza rispetto alla stabilità del sistema, si nota un fallimento cronico legato unicamente all'oggetto *Ball White* (*marker* grigio). Per questo specifico target, il *box plot* spesso non è visibile, e il *Detection Rate* sprofonda a valori critici: 40% nel *Mid Left*, 25% nel *Mid Right*, fino a crollare al 10% in *Far Left* e sfiorare lo zero in *Far Right*. Inoltre, la totale assenza del pallino in posizioni centrali chiave (*Inner, Outer* ed *Extreme Center*) certifica l'incapacità assoluta di generare rilevamenti validi. Questo ribadisce il limite ottico intrinseco già documentato: l'assenza di discontinuità cromatiche tra la sfera bianca e l'ambiente circostante impedisce a *YOLO* di estrarre *feature* valide, indipendentemente dalla distanza; perciò, il degrado è di natura materico cromatica e non spaziale. L'ultimo comportamento anomalo riguarda il *Remote* (telecomando segnato dal *marker* rosso). A differenza della pallina bianca, il telecomando raggiunge il 100% in posizioni vicine e centrali (es. *Inner Center, Mid Left*), ma subisce fortissimi cali prestazionali in posizioni laterali o molto distanti (*Near Left* al 10%, *Near Right* quasi allo 0%, *Far Right* sotto il 40%). In questo caso, il calo non è legato al colore (il telecomando nero è contrastato bene sul tavolo), ma all'angolo di incidenza della telecamera. Essendo un oggetto molto

piatto (*low-profile*), quando si trova lontano o ai margini del *Field of View*, i pixel che lo rappresentano si assottigliano fino ad apparire come una sottilissima linea scura. *YOLO* non dispone di sufficienti informazioni volumetriche in 2D per attivare i nodi di riconoscimento, subendo un collasso drastico dell'affidabilità semantica.

5.2 Validazione Del Nodo Di Grasping In Living Lab

Al fine di validare l'intera pipeline di manipolazione, ho condotto dei test di *pick and place* posizionando i *target* in tre punti strategici del piano di lavoro: una posizione centrale, una sulla destra e una sulla sinistra del piano. Per valutare la robustezza del sistema al variare della geometria, sono stati selezionati tre oggetti: un bicchiere colorato (*cup*), una pallina bianca e rossa (*sports ball*) e un telecomando (*remote*). Il *setup* prevede quindi un totale di 9 acquisizioni (3 oggetti per 3 posizioni), e per ogni iterazione, verranno registrati i tassi di successo delle singole fasi operative del robot:

- *Planning success:* Risoluzione del problema della cinematica inversa (*IK*) al fine di determinare correttamente la posa di pre-presa (*pre-grasp*).
- *Approach Success:* Generazione ed esecuzione del moto lineare di discesa verticale verso l'oggetto per prepararsi al grasping.
- *Grasp Success*: Esecuzione della presa e dunque bloccaggio dell'oggetto tra le dita del *gripper*.
- *Lift Success*: Sollevamento dell'oggetto di qualche centimetro lungo l'asse Y.
- *Place Success*: Riposizionamento del *target* nella medesima posa di partenza in condizioni di stabilità statica.

Al fine di quantificare la precisione di esecuzione del *task* da parte di *TIAGo*, è stato ricavato l'errore di posizionamento, calcolato come differenza tra le coordinate iniziali dell'oggetto prima dell'avvio del nodo di *grasping* e le sue coordinate finali post rilascio.

5.2.1 Analisi Sperimentale E Discussione Dei Risultati Ottenuti

La validazione sperimentale ha previsto l'utilizzo di un piano di lavoro le cui dimensioni, altezza, larghezza e profondità, sono pari rispettivamente a 80 cm, 90 cm e 130 cm. Lo stesso tavolo è stato impiegato per le prove condotte con oggetti di forma cilindrica, mentre per le prove riguardanti la sfera e l'oggetto di forma parallelepipeda, si è reso necessario adottare un piano di lavoro più basso: tali geometrie, infatti, richiedono un approccio cinematico differente rispetto a quello adottato per i cilindri (*Figura 5.14*), e l'utilizzo del tavolo originario (troppo alto) comportava il fallimento della cinematica stessa.

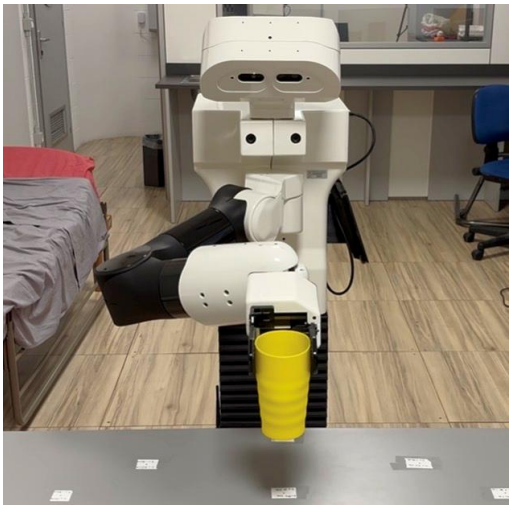


Figura 5.14 Frame estrapolato da video di esecuzione del task di pick and place con oggetto cilindrico.

Per questo motivo, in sostituzione del tavolo, è stata utilizzata una scatola di dimensioni più contenute (*Figura 5.15* e *Figura 5.16*), pari a 60 cm di altezza, 48 cm di larghezza e 36 cm di profondità, che ha permesso di superare tale criticità.



Figura 5.15 Frame estrapolato da video di esecuzione del task di pick and place con oggetto di forma parallelepipeda.

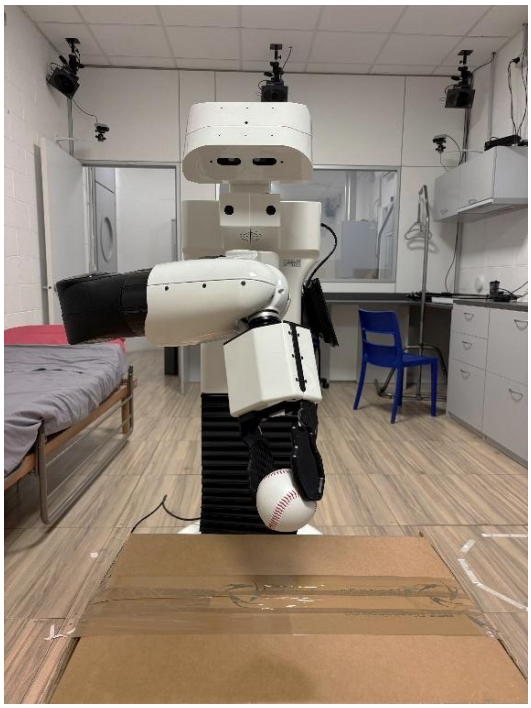


Figura 5.16 Frame estrapolato da video di esecuzione del task di pick and place con oggetto di forma sferica.

Test	*Target*	*Posa Iniziale Oggetto*	*Posa Finale Oggetto*	*Errore Riposizionamento (cm)*
1	Cilindro (centro del piano)	X = 0.858 Y = 0.064 Z = 0.943	X = 0.884 Y = 0.083 Z = 0.944	3.22
2	Cilindro (destra del piano)	X = 0.776 Y = - 0.281 Z = 0.952	X = 0.771 Y = - 0.270 Z = 0.953	1.21
3	Cilindro (sinistra del piano)	X = 0.710 Y = 0.356 Z = 0.943	X = 0.718 Y = 0.374 Z = 0.941	1.98
4	Sfera (centro del piano)	X = 0.818 Y = 0.091 Z = 0.727	X = 0.729 Y = 0.091 Z = 0.729	8.9
5	Sfera (destra del piano)	X = 0.839 Y = - 0.068 Z = 0.734	X = 0.839 Y = - 0.068 Z = 0.734	0
6	Sfera (sinistra del piano)	X = 0.817 Y = 0.257 Z = 0.719	X = 0.816 Y = 0.255 Z = 0.719	0.2
7	Parallelepipedo (centro del piano)	X = 0.764 Y = 0.018 Z = 0.689	X = 0.765 Y = 0.019 Z = 0.688	0.2
8	Parallelepipedo (destra del piano)	X = 0.823 Y = - 0.150 Z = 0.689	X = 0.823 Y = - 0.150 Z = 0.689	0
9	Parallelepipedo (sinistra del piano)	X = 0.760 Y = 0.201 Z = 0.676	X = 0.783 Y = 0.211 Z = 0.647	3.8

Tabella 5.2 Test di validazione del nodo di grasping: prove condotte su tre oggetti target in posizione centrale, destra e sinistra rispetto al piano di lavoro. Dove x corrisponde alla misura di profondità (distanza oggetto-base_footprint), y corrisponde all'asse orizzontale (escursione laterale) e z identifica l'asse verticale (altezza). Il sistema di riferimento risiede in base_footprint.

Le prove di validazione condotte hanno mostrato un tasso di successo del 100% nell'intera pipeline di *pick and place* (acquisizione, pianificazione, presa, sollevamento e rilascio) per tutti gli oggetti testati, confermando l'elevata affidabilità del nodo ROS 2 e del solutore MoveIt. I test hanno evidenziato prestazioni ottimali per tutte le geometrie. Per il cilindro, l'operazione ha richiesto in media 31,28 secondi con un errore di posizionamento di soli 2,14 cm, a conferma dell'efficacia della logica di approccio e *pre-grasp*. La sfera ha registrato l'esecuzione più rapida (28,63 s): la presa diagonale ha evitato con successo urti sul piano e il lieve scostamento di 3,04 cm è dovuto esclusivamente al naturale rotolamento dell'oggetto durante il rilascio. Infine, il

parallelepipedo ha ottenuto la massima precisione spaziale (errore di 1,33 cm) grazie all'approccio *top-down*, a fronte di un tempo leggermente superiore (33,80 s) legato alla maggiore complessità della pianificazione cinematica.

5.3 Conclusioni Della Validazione Sperimentale della Pipeline di Pick And Place

La campagna di validazione sperimentale conferma l'efficacia e la robustezza del nodo *ROS 2 tiago_pick_place* nell'effettuare delle manovre di manipolazione mobile. L'ottenimento di un tasso di successo del 100% su tutte le geometrie analizzate (cilindro, sfera e parallelepipedo) e in diverse posizioni del piano di lavoro attesta la validità dell'architettura di controllo sviluppata. L'impiego di strategie di *grasping* differenziate in base alla forma, abbinato al calcolo dinamico dello *yaw* di approccio, all'applicazione degli *offset* spaziali dedicati e alla verifica della quota minima di sicurezza ha consentito di superare i limiti di accuratezza della stima visiva *RGB-D*, prevenendo collisioni fisiche e singolarità cinematiche. A conclusione della validazione sperimentale, è stata implementata un'ulteriore funzionalità del nodo per rimarcare lo scopo ultimo di questo lavoro di tesi: definire una pipeline modulare di *pick and place* che non si limiti alla sola manipolazione, ma sia strutturata e finalizzata alla riabilitazione robotica post ictus. Per questa ragione, è stato integrato un modulo dedicato che guida il robot *TIAGo*, dopo la fase di presa e sollevamento, nel trasporto dell'oggetto verso il lato destro del tavolo. Nella disposizione d'interazione uomo-robot, questa area corrisponde infatti all'emispazio peripersonale sinistro del paziente. In questo contesto, il *task* di *pick and place* diviene uno strumento terapeutico attivo: lo spostamento dell'oggetto nell'area negletta forza il paziente a svolgere compiti di tracciamento visivo e riesplorazione spaziale, stimolando l'attenzione verso l'emispazio trascurato. Proprio per mostrare la versatilità della *pipeline* e dimostrarne la prontezza operativa in uno scenario clinico reale, sono state condotte alcune prove preliminari di fattibilità, di cui riporto un esempio: partendo da una posa iniziale dell'oggetto pari a $[0.844, 0.039, 0.953]$ m, *TIAGo* ha eseguito il *task*, sollevando e riposizionando l'oggetto nella coordinata finale pari a $[0.866, -0.145, 0.954]$ m (*Figura 5.17*).

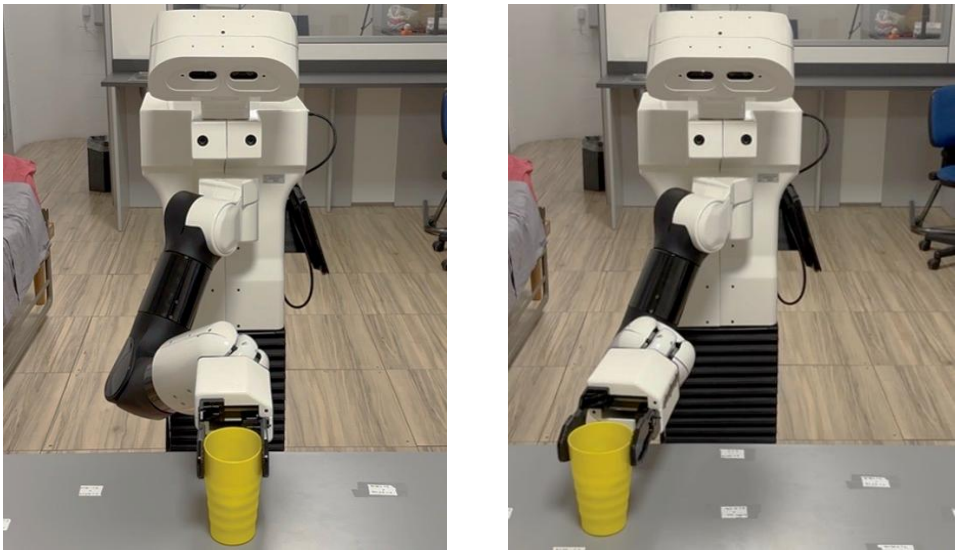


Figura 5.17 Esecuzione del task di pick and place, con spostamento dell'oggetto da una posizione di partenza centrale (immagine a sinistra) ad una nuova posizione sul lato destro del piano d'appoggio (immagine a destra).

L'analisi differenziale delle coordinate conferma che lo spostamento è avvenuto quasi esclusivamente lungo l'asse Y (traslazione laterale), mantenendo pressoché inalterati i valori di X e Z a riprova della precisione del controllo del braccio. Pur trattandosi di riscontri preliminari non ancora integrati in modo sistematico nella sperimentazione clinica, questi dati attestano la robustezza del sistema e pongono le basi per i futuri sviluppi applicativi dedicati alla realizzazione di task di pick and place in sessioni riabilitative post ictus.

6 Capitolo Sei. Conclusioni E Sviluppi Futuri

In questa sezione è presentato l'ultimo capitolo della tesi, in cui vengono discussi i risultati ottenuti e gli eventuali sviluppi futuri del progetto.

6.1 Conclusioni

Il presente lavoro di tesi ha condotto allo sviluppo di una pipeline completa dedicata all'esecuzione di *task* di *pick and place* da parte del robot *TIAGo*. Il sistema integra tecnologie avanzate come algoritmi di *Object Detection* (*YOLO*), tecniche di ricostruzione tridimensionale e logiche di programmazione predittiva, allo scopo di ottimizzare l'efficienza dei movimenti del manipolatore e determinare la strategia di presa più adeguata alle caratteristiche di ciascun oggetto *target*. L'errore registrato sulla stima della posizione dei centroidi è risultato minimo, pari a 5.74 mm, e questo dato ha permesso di ottenere un tasso di successo nelle prove di *grasping* pari al 100%. Il sistema sviluppato ha permesso di ottenere una ricostruzione tridimensionale affidabile, garantendo un'elevata accuratezza nel calcolo delle dimensioni degli oggetti e nella stima del loro baricentro. Il robot ha inoltre dimostrato un'ottima precisione nell'esecuzione del *task* di *pick and place*, con errori di riposizionamento nell'ordine di pochi cm, rispetto alla posizione di partenza.

6.2 Sviluppi E Prospettive Future

Un primo potenziale sviluppo della pipeline riguarda il perfezionamento dell'algoritmo della stima del baricentro. Allo stato attuale, il sistema fornisce risultati precisi e affidabili per oggetti dotati di simmetria radiale, come sfere e cilindri. Al contrario, per solidi come i parallelepipedi, l'approccio adottato costituisce una semplificazione. Il metodo attuale, infatti, prevede il calcolo del centroide sul piano bidimensionale, seguito da una traslazione in profondità pari al raggio o alla metà dello spessore dell'oggetto. Nel caso di geometrie come i parallelepipedi, questo metodo introduce un'approssimazione: non possedendo simmetria radiale, l'estensione in profondità del solido varia in funzione del suo orientamento rispetto alla telecamera (quindi a seconda del fatto che esposto verso la telecamera ci sia il lato corto o il lato lungo dell'oggetto) rendendo la traslazione di un valore fisso a metà spessore non sempre precisa. Nel nostro caso, gli oggetti sono stati appoggiati sulla superficie di supporto con lo spessore lungo l'asse verticale e il lato corto, parallelo all'asse Y, verso *TIAGo* e la telecamera *RGB-D;* perciò la stima ha portato all'ottenimento di un baricentro che ricadeva correttamente sempre all'interno dell'oggetto. Grazie a questa impostazione specifica, durante le prove sperimentali e i *task* di *pick and pace* è stato registrato il 100% di successo. Uno sviluppo futuro, dunque, potrebbe essere quello di sviluppare una funzione dedicata all'estrazione del baricentro 3D di questa tipologia di oggetti, individuando dapprima le tre dimensioni principali

dell'oggetto (lunghezza, larghezza e profondità o altezza) e successivamente ricavando il punto medio di ciascuna dimensione. Durante il progetto di tesi, non è stata implementata questa metodologia a causa della mancanza di oggetti della categoria parallelepipedo che potessero essere analizzati e segmentati con precisione dal modello YOLO. L'obiettivo futuro dell'architettura *software* sviluppata in questo lavoro di tesi potrebbe essere l'evoluzione del robot *TIAGo* da mero esecutore di comandi a *partner* terapeutico intelligente, capace di supportare attivamente i pazienti *post ictus* affetti da eminegligenza spaziale. Per raggiungere questo traguardo, gli sviluppi futuri si potrebbero articolare su due temi principali: l'ottimizzazione tecnica della *pipeline* robotica e la validazione in ambito clinico. Per quanto riguarda l'evoluzione della pipeline, il sistema può essere migliorato aggiungendo un sistema di rotazione automatica della testa per l'inseguimento dell'oggetto durante la movimentazione. Attualmente, l'orientamento della testa di *TIAGo* è fisso e necessita di un intervento manuale per inquadrare l'oggetto al centro della scena. Le capacità di manipolazione di *TIAGo* (tra cui la possibilità di eseguire *task* di *pick and place*) potrebbero inoltre essere integrate in un ecosistema di *Serious Games*, in cui il robot non si limiterà a eseguire movimenti ripetitivi, ma analizzerà i dati visivi in tempo reale per valutare le prestazioni del paziente e attribuire loro un punteggio legato ad un livello di difficoltà nel "gioco". *TIAGo,* in particolare, potrà essere in grado di modulare dinamicamente la difficoltà del compito (ad esempio, posizionando l'oggetto sempre più in profondità nello spazio negletto del paziente) in base ai successi ottenuti durante gli esercizi proposti, evitando così frustrazione o noia, due fattori particolarmente importanti per una riabilitazione ottimale. Il sistema visivo di *TIAGo* potrebbe essere espanso per calcolare e registrare metriche cliniche oggettive, tra cui i tempi di esecuzione del *task*, tracciando il movimento, l'accuratezza del rilascio dell'oggetto e l'ampiezza dell'esplorazione motoria nello spazio controlaterale alla lesione. Il culmine degli sviluppi futuri dovrebbe focalizzarsi nel trasferimento della tecnologia dall'ambiente di laboratorio alla reale pratica ospedaliera, tramite uno studio pilota in collaborazione con strutture cliniche specializzate su un campione di pazienti *post ictus*. Questa fase permetterebbe di validare non solo la robustezza del *software* e l'usabilità del sistema, ma soprattutto l'efficacia terapeutica del robot nel promuovere la neuroplasticità, il recupero funzionale e la mitigazione dei sintomi di eminegligenza spaziale.

Bibliografia

[1] D. S. J. Ting, A. Pollock, F. N. Doubal, D. S. W. Ting, M. Thompson, and B. Dhillon, “Visual Neglect Following Stroke: Current Concepts and Future Focus,” *Surv. Ophthalmol.*, vol. 56, no. 2, pp. 114–134, Mar. 2011, doi: 10.1016/j.survophthal.2010.08.001.

[2] P. Azouvi, S. Jacquin-Courtois, and J. Luauté, “Rehabilitation of unilateral neglect: Evidence-based medicine,” *Ann. Phys. Rehabil. Med.*, vol. 60, no. 3, pp. 191–197, Jun. 2017, doi: 10.1016/j.rehab.2016.10.006.

[3] M. J. Moore, E. Milosevich, J. B. Mattingley, and N. Demeyere, “The neuroanatomy of visuospatial neglect: A systematic review and analysis of lesion-mapping methodology,” *Neuropsychologia*, vol. 180, p. 108470, Feb. 2023, doi: 10.1016/j.neuropsychologia.2023.108470.

[4] M. N. Toba, R. Migliaccio, A. Potet, P. Pradat-Diehl, and P. Bartolomeo, “Right-side spatial neglect and white matter disconnection after left-hemisphere strokes,” *Brain Struct. Funct.*, vol. 227, no. 9, pp. 2991–3000, Aug. 2022, doi: 10.1007/s00429-022-02541-7.

[5] G. Kerkhoff and T. Schenk, “Rehabilitation of neglect: An update,” *Neuropsychologia*, vol. 50, no. 6, pp. 1072–1079, May 2012, doi: 10.1016/j.neuropsychologia.2012.01.024.

[6] A. Gouret, A. Delaux, S. Le Bars, and S. Chokron, “Rewiring Attention: Virtual Reality and Brain–Computer Interfaces in the Rehabilitation of Unilateral Spatial Neglect,” *J. Clin. Med.*, vol. 15, no. 3, p. 1036, Jan. 2026, doi: 10.3390/jcm15031036.

[7] A. Z. Durfee and A. E. Hillis, “Unilateral Spatial Neglect Recovery Poststroke,” *Stroke*, vol. 54, no. 1, pp. 10–19, Jan. 2023, doi: 10.1161/STROKEAHA.122.041710.

[8] Y. S. Choi, K. W. Lee, J. H. Lee, S. B. Kim, G. T. Park, and S. J. Lee, “The effect of an upper limb rehabilitation robot on hemispatial neglect in stroke patients,” *Ann. Rehabil. Med.*, vol. 40, no. 4, pp. 611–619, 2016, doi: 10.5535/arm.2016.40.4.611.

[9] K. Li and P. A. Malhotra, “Spatial neglect,” *Pract. Neurol.*, vol. 15, no. 5, pp. 333–339, Oct. 2015, doi: 10.1136/practneurol-2015-001115.

[10] P. Langhorne, J. Bernhardt, and G. Kwakkel, “Stroke rehabilitation,” *The Lancet*, vol. 377, no. 9778, pp. 1693–1702, May 2011, doi: 10.1016/S0140-6736(11)60325-5.

[11] X. Li, Y. He, D. Wang, and M. J. Rezaei, “Stroke rehabilitation: from diagnosis to therapy,” *Front. Neurol.*, vol. 15, p. 1402729, Aug. 2024, doi: 10.3389/fneur.2024.1402729.

[12] Y. Rossetti, G. Rode, L. Pisella, A. Farné, L. Li, D. Boisson, and M. T. Perenin, “Prism adaptation to a rightward optical deviation rehabilitates left hemispatial neglect,” *Nature*, vol. 395, no. 6698, pp. 166–169, Sep. 1998, doi: 10.1038/25988.

[13] R. Sturt and T. D. Punt, “Caloric vestibular stimulation and postural control in patients with spatial neglect following stroke,” *Neuropsychol. Rehabil.*, vol. 23, no. 2, pp. 299–316, 2013, doi: 10.1080/09602011.2012.755831.

[14] W. Fu, Y. Bai, and X. Wang, “Galvanic vestibular stimulation for postural rehabilitation in neurological disorders: a systematic review,” *Front. Neurosci.*, vol. 19, p. 1580078, Apr. 2025, doi: 10.3389/fnins.2025.1580078.

[15] A. P. Yelnik, A. Kassouha, I. V. Bonan, M. C. Leman, C. Jacq, E. Vicaut, and F. M. Colle, “Postural visual dependence after recent stroke: Assessment by optokinetic stimulation,” *Gait Posture*, vol. 24, no. 3, pp. 262–269, Nov. 2006, doi: 10.1016/j.gaitpost.2005.09.007.

[16] K. Jamal, S. Leplaideur, C. Rousseau, S. Cordillet, A. M. Raillon, S. Butet, A. Cretual, and I. Bonan, “The effects of repetitive neck-muscle vibration on postural disturbances after a chronic stroke,” *Neurophysiol. Clin.*, vol. 50, no. 4, pp. 269–278, Sep. 2020, doi: 10.1016/j.neucli.2020.01.005.

[17] M. J. Matarić, J. Eriksson, D. J. Feil-Seifer, and C. J. Winstein, “Socially assistive robotics for post-stroke rehabilitation,” *J. Neuroeng. Rehabil.*, vol. 4, no. 1, p. 5, Feb. 2007, doi: 10.1186/1743-0003-4-5.

[18] M. Chita-Tegmark and M. Scheutz, “Assistive Robots for the Social Management of Health: A Framework for Robot Design and Human–Robot Interaction Research,” *Int. J. Soc. Robot.*, vol. 13, no. 2, pp. 197–217, Mar. 2020, doi: 10.1007/s12369-020-00634-z.

[19] A. Carnevale, A. Raso, C. Antonacci, L. Mancini, A. Corradini, A. Ceccaroli, C. Casciaro, V. Candela, A. de Sire, P. D’Hooghe, and U. G. Longo, “Exploring the Impact of Socially Assistive Robots in Rehabilitation Scenarios,” *Bioengineering*, vol. 12, no. 2, p. 204, Feb. 2025, doi: 10.3390/bioengineering12020204.

[20] K. Swift-Spong, E. Short, E. Wade, and M. J. Mataric, “Effects of comparative feedback from a Socially Assistive Robot on self-efficacy in post-stroke rehabilitation,” in *Proc. IEEE Int. Conf. Rehabil. Robot.*, Sep. 2015, pp. 764–769, doi: 10.1109/ICORR.2015.7281294.

[21] S. Karner, H. Stenner, M. Spate, J. Behrens, and K. Krakow, “Effects of a robot intervention on visuospatial hemineglect in postacute stroke patients: a randomized controlled trial,” *Clin. Rehabil.*, vol. 33, no. 12, pp. 1940–1948, Dec. 2019, doi: 10.1177/0269215519865993.

[22] J. Yang, Y. Zhu, H. Li, K. Wang, D. Li, and Q. Qi, “Effect of robotic exoskeleton training on lower limb function, activity and participation in stroke patients: a systematic review and meta-analysis of randomized controlled trials,” *Front. Neurol.*, vol. 15, p. 1453781, Aug. 2024, doi: 10.3389/fneur.2024.1453781.

[23] A. Demeco, L. Zola, A. Frizziero, C. Martini, A. Palumbo, R. Foresti, G. Buccino, and C. Costantino, “Immersive Virtual Reality in Post-Stroke Rehabilitation: A Systematic Review,” *Sensors*, vol. 23, no. 3, p. 1712, Feb. 2023, doi: 10.3390/s23031712.

[24] R. Bazan, B. H. de S. Fonseca, J. M. de A. Miranda, H. R. de C. Nunes, S. G. Z. Bazan, and G. J. Luvizutto, “Effect of Robot-Assisted Training on Unilateral Spatial Neglect After Stroke: Systematic Review and Meta-Analysis of Randomized Controlled Trials,” *Neurorehabil. Neural Repair*, vol. 36, no. 8, pp. 545–556, Aug. 2022, doi: 10.1177/15459683221110894.

[25] PAL Robotics, “Documentation - PAL Robotics.” Accessed: Sep. 03, 2026. [Online]. Available: https://pal-robotics.com/documentation/

[26] S. Macenski, A. Soragna, M. Carroll, and Z. Ge, “Impact of ROS 2 Node Composition in Robotic Systems,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 7, pp. 3996–4003, Jul. 2023, doi: 10.1109/LRA.2023.3279614.

[27] W.-L. Chuang, M.-H. Yeh, and Y.-L. Yeh, “Develop Real-Time Robot Control Architecture Using Robot Operating System and EtherCAT,” *Actuators*, vol. 10, no. 7, p. 141, Jun. 2021, doi: 10.3390/act10070141.

[28] I. Malavolta, G. Lewis, B. Schmerl, P. Lago, and D. Garlan, “How do you architect your robots? State of the practice and guidelines for ros-based systems,” in *Proc. Int. Conf. Softw. Eng.*, Jun. 2020, pp. 31–40, doi: 10.1145/3377813.3381358.

[29] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, “3-D Mapping with an RGB-D camera,” *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, 2014, doi: 10.1109/TRO.2013.2279412.

[30] J. G. Da Silva Neto, P. J. Da Lima Silva, F. Figueredo, J. M. X. N. Teixeira, and V. Teichrieb, “Comparison of RGB-D sensors for 3D reconstruction,” in *Proc. 22nd Symp. Virtual Augmented Reality (SVR)*, Nov. 2020, pp. 252–261, doi: 10.1109/SVR51698.2020.00046.

[31] R. Herrera, D. Pozo-Espin, and M. Herrera, “3D face reconstruction with RGB-D camera: A practical approach,” in *Proc. 2nd Int. Conf. Inf. Syst. Softw. Technol. (ICI2ST)*, Mar. 2021, pp. 25–30, doi: 10.1109/ICI2ST51859.2021.00012.

[32] Orbbec, “Astra Series: Structured Light Cameras | Orbbec.” Accessed: Sep. 08, 2026. [Online]. Available: https://www.orbbec.com/products/structured-light-camera/astra-series/

[33] J. Shen and S.-C. S. Cheung, “Layer Depth Denoising and Completion for Structured-Light RGB-D Cameras,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 1187-1194, doi: 10.1109/CVPR.2013.157.

[34] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[35] T. Diwan, G. Anirudh, and J. V. Tembhurne, “Object detection using YOLO: challenges, architectural successors, datasets and applications,” *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, Aug. 2022, doi: 10.1007/s11042-022-13644-Y.

[36] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, “Deep Learning for Generic Object Detection: A Survey,” *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Oct. 2019, doi: 10.1007/s11263-019-01247-4.

[37] A. Munteanu, H. Wang, P. Hu, W. Darwish, S. E. Ghazouali, Y. Mhirit, A. Oukhrid, U. Michelucci, and H. Nouira, “FusionVision: A Comprehensive Approach of 3D Object Reconstruction and Segmentation from RGB-D Cameras Using YOLO and Fast Segment Anything,” *Sensors*, vol. 24, no. 9, p. 2889, Apr. 2024, doi: 10.3390/s24092889.

[38] U. Budak, L. Ilola, L. Kondrad, P. R. Alface, P. Fasogbon, and S. Schwarz, “Essential depth map processing algorithms for real-time applications: an overview,” *IET Conf. Proc.*, vol. 2024, no. 10, pp. 138–145, 2024, doi: 10.1049/ICP.2024.3296.

[39] Y. Zhang and T. Funkhouser, “Deep Depth Completion of a Single RGB-D Image,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2018, pp. 175–185, doi: 10.1109/CVPR.2018.00026.

[40] M. Bertalmío, A. L. Bertozzi, and G. Sapiro, “Navier-Stokes, fluid dynamics, and image and video inpainting,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, 2001, doi: 10.1109/CVPR.2001.990497.

[41] L. Li, F. Yang, H. Zhu, D. Li, Y. Li, and L. Tang, “An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells,” *Remote Sens.*, vol. 9, no. 5, p. 433, 2017, doi: 10.3390/rs9050433.

[42] Z. Jin, T. Tillo, and F. Cheng, “Depth-map driven planar surfaces detection,” in *Proc. IEEE Vis. Commun. Image Process. Conf. (VCIP)*, Feb. 2015, pp. 514–517, doi: 10.1109/VCIP.2014.7051619.

[43] L. Yang, Y. Li, X. Li, Z. Meng, and H. Luo, “Efficient plane extraction using normal estimation and RANSAC from 3D point cloud,” *Comput. Stand. Interfaces*, vol. 82, p. 103608, Aug. 2022, doi: 10.1016/j.csi.2021.103608.

[44] N. Tukhtamanov, E. Chebotareva, T. Tsoy, R. Lavrenov, M. Svinin, and E. Magid, “Open Source Library of Human Models for Gazebo Simulator,” in *Proc. Int. Siberian Conf. Control Commun. (SIBCON)*, 2022, doi: 10.1109/SIBCON56144.2022.10002978.

[45] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, “Simulation environment for mobile robots testing using ROS and Gazebo,” in *Proc. 20th Int. Conf. Syst. Theory Control Comput. (ICSTCC)*, Dec. 2016, pp. 96–101, doi: 10.1109/ICSTCC.2016.7790647.

[46] N. Koenig and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, 2004, pp. 2149–2154, doi: 10.1109/IROS.2004.1389727.

[47] T. Linner, A. Shrikathiresan, M. Vetrenko, B. Ellmann, and T. Bock, “Modeling and operating robotic environments using Gazebo/ROS,” in *Proc. 31st Int. Symp. Autom. Robot. Constr. Min. (ISARC)*, Sydney, Australia, 2014, pp. 493-500, doi: 10.22260/ISARC2014/0065.

[48] A. Valverde and P. Tsiotras, “Spacecraft robot kinematics using dual quaternions,” *Robotics*, vol. 7, no. 4, Oct. 2018, doi: 10.3390/robotics7040064.

[49] A. Bajrami, M. C. Palpacelli, L. Carbonari, and D. Costa, “Posture Optimization of the TIAGo Highly-Redundant Robot for Grasping Operation,” *Robotics*, vol. 13, no. 4, p. 56, Mar. 2024, doi: 10.3390/robotics13040056.

[50] G. S. Kashyap, D. Mahajan, O. C. Phukan, A. Kumar, A. E. I. Brownlee, and J. Gao, “From simulations to reality: enhancing multi-robot exploration for urban search and rescue,” *Int. J. Inf. Technol.*, vol. 17, no. 7, pp. 4261–4272, Apr. 2025, doi: 10.1007/s41870-025-02513-0.

[51] E. Aljalbout, J. Xing, A. Romero, I. Akinola, C. R. Garrett, E. Heiden, A. Gupta, T. Hermans, Y. Narang, D. Fox, D. Scaramuzza, and F. Ramos, “The Reality Gap in Robotics: Challenges, Solutions, and Best Practices,” *Annu. Rev. Control Robot. Auton. Syst.*, vol. 9, no. 1, pp. 403–432, May 2026, doi: 10.1146/annurev-control-031924-100130.

Sommario delle Figure

Figura 1.1 Confronto tra un soggetto sano e un paziente con *NSU* nel test di bisezione di linee. Al paziente viene chiesto di tracciare una linea al cento del segmento, indice della sua corretta percezione degli spazi destro e sinistro. Lo spostamento del segno verso destra da parte del paziente colpito da neglect evidenzia l'incapacità del paziente di elaborare correttamente gli stimoli provenienti dall'emispazio sinistro [1]. 4

Figura 1.2 Test di copia: lo scopo è disegnare una copia del simbolo che si osserva sulla parte sinistra del foglio. I pazienti affetti da neglect tendono ad omettere gli elementi situati sul lato sinistro di ciascun oggetto [9]. 6

Figura 1.3 Test di cancellazione: lo scopo è trovare un simbolo target tra altri simboli distrattori, in questo caso le figure a forma di stella più piccole. I pazienti affetti da neglect individuano bersagli esclusivamente sul lato ipsilesionale [8]. 6

Figura 1.4 Confronto tra paziente trattato con adattamento prismatico (immagine a sinistra) e paziente che ha eseguito la riabilitazione con occhiali neutri (immagine a destra). 9

Figura 2.1 Componenti hardware principali del robot di servizio TIAGo utilizzato per il progetto di tesi [26]. 15

Figura 2.2 Componenti, cinematica e ingombri della parte superiore di TIAGo. Sulla sinistra vediamo i sensori visivi e uditivi (telecamera RGB-D, microfoni stereo e autoparlante) e i punti di ancoraggio sulla sommità. Il pannello centrale mette in evidenza la configurazione dei due giunti rotoidali (pan e tilt) che orientano la testa. Sulla destra, infine, viene mostrato lo spazio utile a disposizione (33 cm x 27,5 cm) per l'eventuale alloggiamento di un pc portatile o di hardware aggiuntivo [26]. 16

Figura 2.3 Modello cinematico e limiti articolari del robot di servizio TIAGo. Sulla sinistra troviamo la rappresentazione dei sistemi di riferimento sui giunti; a destra, i limiti meccanici di escursione per il torso, la testa e il braccio robotico a sette gradi di libertà [26]. 17

Figura 2.4 Parallel gripper in dotazione sul modello di TIAGo utilizzato per il lavoro di tesi [26]. 17

Figura 2.5 Vista frontale (figura di sinistra) e vista posteriore (figura di destra) della base mobile di TIAGo. L'immagine evidenzia la disposizione dei principali sensori di navigazione (telemetro laser e sonar posteriori) e delle interfacce di alimentazione e servizio [26]. 18

Figura 2.6 Visualizzazione della scena nel software RViz, dove possiamo osservare il modello cinematico del robot TIAGo sulla destra e il braccio manipolatore con il gripper aperto intento ad eseguire un task di pick and place. Notiamo poi la planning scene, tramite un blocco marrone che identifica l'ingombro del piano di lavoro e un cilindro verde sopra il piano che rappresenta il /grasping_markers corrispondente all'oggetto target. 20

Figura 2.7 Telecamera Orbbec Astra in configurazione stand alone [33]. 21

Figura 3.1 Object detection. Nell'ordine, da sinistra verso destra, osserviamo: classificazione, localizzazione (tramite bounding box) e segmentazione della stessa immagine di riferimento [35]. 25

Figura 3.2 Esempio di ricostruzione 3D basata sulla mappa di profondità: l'immagine evidenzia la presenza di pixel fluttuanti (flying pixels) lungo i margini dell'oggetto e l'instabilità dinamica nel calcolo del baricentro, fenomeno visibile come continue fluttuazioni nei video di ricostruzione. 28

Figura 3.3 Confronto tra la segmentazione 2D ideale (a sinistra) e l'immagine di debug (a destra), dove i pixel bianchi mancanti, che rendono la figura incompleta, evidenziano la perdita di segnale di profondità sulla superficie curva dell'oggetto. 29

Figura 3.4 Esempio di ricostruzione di un oggetto cilindrico per focus sul problema dei dati mancanti nella ricostruzione basata su depth. 29

Figura 3.5 Risultato del fitting geometrico basato sul modello volumetrico. 33

Figura 3.6 Box plot dell'errore relativo dimensionale dei modelli YOLO: vista di dettaglio con limite di scala al 20% per agevolare il confronto della tendenza centrale..... 42

Figura 3.7 Box plot degli errori dimensionali nella condizione di acquisizione "condizioni ottimali": posizione dell'oggetto rispetto alla telecamera pari a 95 cm di distanza. 43

Figura 3.8 Box plot degli errori dimensionali della condizione di acquisizione "condizioni ottimali": posizione dell'oggetto rispetto alla telecamera pari a 95 cm di distanza. Con zoom al 15%. I trattini orizzontali isolati nei grafici indicano campioni a varianza nulla, mentre le scatole mancanti segnalano veri fallimenti di rilevamento. 44

Figura 4.1 Esempio di scenario realizzabile attraverso l'ambiente simulato Gazebo. ... 49

Figura 4.2 Visualizzazione su Gazebo della scena di simulazione controllata, popolata manualmente selezionando gli oggetti disponibili del sistema. 51

Figura 4.3 Software RViz. 54

Figura 4.4 Test di validazione del nodo di visione in simulazione nella piattaforma Gazebo. A sinistra (in Gazebo) possiamo osservare la scena di simulazione con l'oggetto target posto sul piano di lavoro e il robot TIAGo posizionato davanti al tavolo. A destra troviamo invece la piattaforma RViz in cui possiamo osservare l'oggetto sottoforma di marker rosso sovrapposto alla vista RGB di TIAGo. 55

Figura 4.5 Test di validazione del nodo di visione con oggetto target sferico. A sinistra osserviamo la scena del software Gazebo, mentre a destra abbiamo la rappresentazione in RViz. 57

Figura 4.6 Fase di grasping all'interno dell'ambiente simulato. Il gripper di TIAGo esegue la chiusura adattiva attorno all'oggetto target cilindrico fino al rilevamento del contatto fisico. A destra (in RViz) è visibile la corretta pianificazione spaziale, mentre a sinistra (in Gazebo) avviene l'effettiva interazione fisica. 61

Figura 5.1 Visualizzazione delle posizioni scelte nel piano di lavoro contrassegnate da marker per la validazione del nodo di visione. 65

Figura 5.2 Massima escursione possibile del braccio manipolatore di TIAGo. 65

Figura 5.3 Esempio di esecuzione del test per il nodo di visione con inquadratura sulla sinistra: visualizzazione dell'output su RViz, in cui si evince il corretto posizionamento del marker dell'oggetto nello spazio operativo. 66

Figura 5.4 Errore percentuale dimensionale del diametro delle sfere rispetto alle varie posizioni sul piano di lavoro. 67

Figura 5.5 Confronto tra pallina bianca (Ball White), sulla sinistra, e pallina gialla (Ball Yellow), sulla destra: possiamo osservare come, rispetto alla stessa posizione sul piano e la medesima illuminazione ambientale, YOLO identifichi correttamente la pallina gialla (grazie al suo contrasto rispetto al piano) e classifichi correttamente l'oggetto target (come sports ball), rispetto alla pallina bianca che, a causa dello scarso contrasto, non viene percepita dalla rete YOLO. 68

Figura 5.6 Errore percentuale dimensionale dell'altezza degli oggetti target (cilindri e parallelepipedi) rispetto alle varie posizioni sul piano di lavoro. 69

Figura 5.7 Errore percentuale dimensionale della larghezza degli oggetti target (cilindri e parallelepipedi) rispetto alle varie posizioni sul piano di lavoro. 70

Figura 5.8 Stabilità del baricentro degli oggetti target al variare della loro posizione sul piano di lavoro. 71

Figura 5.9 Stabilità del baricentro degli oggetti target al variare della loro posizione sul piano di lavoro, con zoom del grafico per mettere in evidenza i casi al di sotto della soglia scelta di 1 mm. 72

Figura 5.10 Mappa bidimensionale della localizzazione dei baricentri dei vari oggetti target identificati dal sistema di visione (apprezzabili tramite marker colorati), rispetto alle posizioni del ground truth identificate dai simboli "+" presenti nel grafico. 73

Figura 5.11 Visualizzazione della scena di riferimento con *TIAGo* e il piano di lavoro. 74

Figura 5.12 Mappa calibrata della localizzazione del baricentro degli oggetti sul piano di lavoro, non affetta da errore sistematico. 76

Figura 5.13. Grafico che mostra la percentuale di successo di identificazione dell'oggetto da parte della rete YOLO (detection rate), rispetto alle varie posizioni dell'oggetto stesso sul piano di lavoro. 77

Figura 5.14 Frame estrapolato da video di esecuzione del task di pick and place con oggetto cilindrico. 79

Figura 5.15 Frame estrapolato da video di esecuzione del task di pick and place con oggetto di forma parallelepipeda. 79

Figura 5.16 Frame estrapolato da video di esecuzione del task di pick and place con oggetto di forma sferica. 80

Figura 5.17 Esecuzione del task di pick and place, con spostamento dell'oggetto da una posizione di partenza centrale (immagine a sinistra) ad una nuova posizione sul lato destro del piano d'appoggio (immagine a destra). 83

Sommario delle Tabelle

Tabella 3.1 Valutazione comparativa delle pipeline di visione (Depth vs Volume): stime dimensionali, ground truth delle dimensioni e stima del baricentro. Dove x corrisponde all'asse orizzontale, y all'asse verticale e z identifica la profondità, quindi la distanza oggetto-telecamera. Il sistema di riferimento risiede nel centro dell'obiettivo della Orbbec Astra. 35

Tabella 3.2 Valutazione comparativa delle pipeline di visione (Depth vs Volume): oscillazione tra i vari frame delle stime dimensionali (diametro, larghezza e altezza) e jitter del baricentro (vedi equazione paragrafo 3.5.2 Protocollo Di Test: Performance A Confronto). 37

Tabella 3.3 Analisi dell'instabilità spaziale (Jittering): medie e deviazioni standard lungo gli assi X, Y e Z registrate dai diversi modelli YOLO negli scenari ambientali di test Condizioni Ottimali, Base Piano Scura e Luce Scarsa. 46

Tabella 3.4 Analisi dell'instabilità spaziale (Jittering): medie e deviazioni standard lungo gli assi X, Y e Z registrate dai diversi modelli YOLO negli scenari ambientali di test Maggiore Distanza su Piano con Base Chiara e Scura. 47

Tabella 4.1 Risultati ottenuti dalle prove di validazione del nodo di visione condotte sui due oggetti target disponibili nel sistema (cilindro e sfera). Dove x corrisponde alla misura di profondità (distanza oggetto-base_footprint), y corrisponde all'asse orizzontale (escursione laterale) e z identifica l'asse verticale (altezza). Il sistema di riferimento risiede in base_footprint. 56

Tabella 5.1 Risultati ottenuti da calcolo degli errori di misura nella fase iniziale prima dell'eliminazione del bias sistematico e confronto con la fase finale dopo l'eliminazione del bias e la correzione delle misure. 75

Tabella 5.2 Test di validazione del nodo di grasping: prove condotte su tre oggetti target in posizione centrale, destra e sinistra rispetto al piano di lavoro. Dove x corrisponde alla misura di profondità (distanza oggetto-base_footprint), y corrisponde all'asse orizzontale (escursione laterale) e z identifica l'asse verticale (altezza). Il sistema di riferimento risiede in base_footprint. 81