POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Interfacce di realtà aumentata a supporto della programmazione di manipolatori robotici

Relatore
prof. Andrea Sanna

Candidato
Valerio Belcamino

Aprile 2021

Indice

Elenco delle figure

Elenco delle tabelle

Abstract

Insieme ai progressi nel campo dell'automazione nell'industria 4.0, l'impiego di robot nel settore industriale ha subito un forte sviluppo portando ad una conseguente diffusione sempre più capillare. Al contempo, le interfacce più comuni per il controllo dei manipolatori come le pulsantiere palmari e i joystick si rivelano poco intuitive e molto time-consuming per l'addestramento di personale. Anche i task più semplici, come per esempio afferrare un oggetto di forma nota e posizionarlo in un altro luogo, richiedono l'intervento di un esperto addetto alla programmazione del robot.

Ecco che quindi, insieme al numero di manipolatori impiegati nel settore industriale, cresce anche il bisogno di interfacce di più facile comprensione più adatte ad essere utilizzate da personale non specializzato in modo da favorire l'interazione tra uomo e robot.

Fin dagli albori della Realtà Aumentata (AR), questa tecnologia ha trovato una delle sue principali aree di applicazione nell'industria semplificando notevolmente le fasi di training dei dipendenti e quelle di manutenzione o riparazione dei macchinari; tuttavia l'adozione di queste tecniche è sempre stata frenata dalla necessità di fornirsi di dispositivi dedicati con un costo proibitivo. La capillare adozione di dispositivi mobili come tablet e smartphone, forniti di sensori e processori adatti allo sviluppo ed alla distribuzione di software AR ha rimosso in fretta questa limitazione, accrescendone le potenzialità.

In particolare, nell'ambito della human-robot collaboration, l'AR permette di incrementare l'affidabilità e la sicurezza rendendo semplice per l'utente capire i movimenti che il manipolatore sta per eseguire e le forze che verranno applicate tramite modelli virtuali animati e posizionati coerentemente nell'ambiente reale. Grande rilevanza è assunta anche dalla visualizzazione in anteprima della traiettoria del robot che fa sì che sia possibile rilevare in anticipo quelli che potrebbero risultare pericolosi o semplicemente inadatti alla conformazione dell'area di lavoro e consente di apportarvi modifiche o interrompere il processo.

Questo lavoro di tesi si pone l'obiettivo di progettare e sviluppare un'applicazione basata su tecnologie di AR per la definizione del path planning di manipolatori industriali mediante una cosiddetta 3DUI (three dimensional user interface). L'interfaccia permette all'utente di inserire, cancella e manipolare i way point che definiscono il cammino del manipolatore.

Essendo la semplicità d'uso uno dei fattori determinanti nella scelta dell'AR rispetto ad altre tecnologie, le funzioni sopra citate sono realizzate in modo da poter essere utilizzate da chiunque abbia dimestichezza con dispositivo dotato di touch screen senza dover necessariamente avere delle competenze nell'ambito della programmazione robotica.

Le limitazioni causate dalla forma bidimensionale dello schermo del dispositivo sono state superate con l'utilizzo di pulsanti e metafore grafiche composte da frecce e gimbal in modo da permettere di posizionare e orientare l'end effector del manipolatore in modo analogo a quanto avviene nei più diffusi game engine e software di modellazione 3D.

Il sistema si compone di un client, realizzato in Unity con l'ausilio della libreria di AR Vuforia e di comunicazione distribuita ROSSharp, e di un server ROS sviluppato in Python che realizza il motore di cinematica inversa e si occupa di trasformare la lista di punti ricevuta nel vero e proprio cammino del manipolatore.

Rispetto ad altri sistemi simili già sviluppati, che permettono di selezionare solo posizioni sul piano di lavoro o su multipli di quantità predefinite, questo progetto ha come obiettivo quello di dare la possibilità di posizionare punti con orientamento e posizione a piacere con 6 gradi di libertà. Al fine di raggiungere tale scopo viene mostrata a runtime la posizione che l'end effector andrà ad assumere in corrispondenza dei riferimenti tracciati dall'utente tramite una sua rappresentazione virtuale.

Altra componente fondamentale dell'applicazione è la funzione di previsualizzazione, già menzionata nell'introduzione, che produce un'animazione del modello del robot virtuale che ricalca il movimento che la sua copia reale eseguirà a seguito della definizione dei way point, in questo modo viene consentita la modifica del movimento del manipolatore prima che esso venga realmente eseguito incrementando così la sicurezza del sistema di programmazione del manipolatore. È inoltre presente un sistema di rilevamento di collisioni con ostacoli in posizioni note che avverte l'utilizzatore marcando il percorso come non valido e impedendone l'esecuzione senza previa modifica.

La precisione dell'interfaccia virtuale è stata successivamente valutata eseguendo dei test in cui i partecipanti avevano il compito di posizionare due punti in corrispondenza di artefatti reali e prendendo la distanza rispetto a un riferimento comune descritto dal centro della base del manipolatore. Tali artefatti sono muniti di indicatori che specificano l'orientamento con il quale si deridera far giungere l'end effector a destinazione e hanno altezze diverse a seconda del livello di complessità. Sono stati pensati cinque task a difficoltà crescente i cui criteri di giudizio consistono nella differenza tra la posizione e l'orientamento proposti con l'oggetto reale e quelli raggiunti con l'interfaccia AR.

Capitolo 1

Introduzione

1.1 Realtà aumentata

La Realtà Aumentata consiste di tutte quelle tecnologie che permettono di fondere elementi reali e virtuali al fine di alterare la percezione visiva di un determinato scenario.

Più precisamente, la Augmented Reality ricade nell'insieme più generale di Mixed Reality, definita nel 1994 da Fumio Kishino e Paul Milgram [1], essa indica le tecniche attraverso cui si ottengono ambienti rappresentabili lungo il Virtuality Conituum: la gamma di sfumature di virtualità i cui estremi sono determinati dal mondo reale e dalla totalmente immersiva Realtà Virtuale.

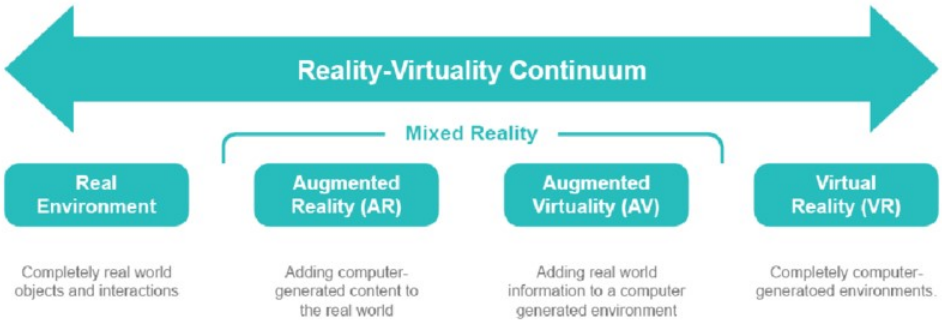

Figura 1.1: Il Continuo della Virtualità [6]

Il posizionarsi tra i due estremi, in particolar modo nella metà sinistra dell'asse, offre diversi vantaggi sia per gli sviluppatori che per gli utenti.

Essendo il reale preponderante all'interno della scena, è necessario sviluppare un numero minore di elementi virtuali, riducendo così sia il costo in termini di tempo e budget durante lo sviluppo, sia il bisogno di capacità computazionali molto elevate.

Dal punto di vista dell'utilizzatore, queste particolarità possono riflettersi in una maggiore accessibilità in quanto non vi è più bisogno di hardware particolarmente potente. Inoltre l'avere sempre ben visibile il mondo circostante elimina le problematiche legate all'immersività totale in un ambiente virtuale che possono insorgere dopo un tempo di utilizzo prolungato o in chi fosse particolarmente predisposto quali motion sickness, nausea e vertigini.

Una definizione più specifica di AR è stata proposta da Ronald Azuma [2], che, nel 1997 ha indicato le tre caratteristiche fondamentali di questa tecnologia:

1. Deve combinare elementi virtuali e reali.
2. Coerenza nella disposizione degli elementi virtuali all'interno del contesto reale.
3. L'applicazione funziona in tempo reale, in tre dimensioni e deve avere delle caratteristiche di interattività.

Il primo di questi punti ribadisce il concetto già espresso nel 1994, mentre i due successivi aggiungono delle nozioni fondamentali alla definizione di AR: ciò che viene espresso al secondo punto significa infatti che gli elementi aumentati devono essere inseriti con criterio all'interno del mondo reale in modo da fondersi il più possibile con lo stesso e creare l'illusione di una realtà unica e coerente mentre l'ultimo punto afferma che l'applicazione deve fornire feedback costanti all'operatore e deve dare la possibilità di interagire con gli asset virtuali.

Affinché il secondo punto possa essere soddisfatto è necessario sapere in ogni momento la posizione degli oggetti inseriti nell'applicazione rispetto al mondo reale, questo obiettivo si raggiunge tramite le tecniche di tracking, che saranno approfondite nei prossimi capitoli e che si avvalgono di componenti hardware quali giroscopi, accelerometri o sensori, che, ad oggi, sono disponibili in quasi tutti i dispositivi portatili che possediamo, come ad esempio gli smartphone.

1.1.1 Framework per la realtà aumentata

I blocchi fondamentali che costituiscono la pipeline di lavoro per un'applicazione AR sono generalmente:

1. Acquisizione del flusso video (tramite camera o altri tipi di sensori).
2. Sistema di tracking del mondo reale.
3. Rendering Engine che genera gli asset e gli applica le corrette informazioni sulla posizione.
4. Sovrapposizione degli elementi virtuali sul flusso video.

Questi passaggi sono necessari al fine di creare un ambiente che sia quanto più possibile coerente e credibile, particolare enfasi è data soprattutto al tracking in quanto il posizionamento erroneo degli asset potrebbe non solo compromettere l'immersività del software, ma renderlo completamente inutilizzabile per l'utente finale.

La parte di acquisizione e quella di rendering giocano invece un ruolo fondamentale per quanto riguarda l'interazione e il feedback e devono essere progettati in modo da minimizzare i tempi di attesa così da creare un'esperienza più fluida e immediata

I framework, anche conosciuti come Software Development Kit (SDK), sono gli strumenti utilizzati forniti agli sviluppatori per implementare le funzionalità della Realtà Aumentata. Questi kit semplificano di molto lo sviluppo delle principali funzionalità richieste in questo tipo di applicazioni tra cui AR recognition, AR tracking e AR content rendering e sono disponibili delle comparazioni tra le varie SDK più utilizzate proposti da [3] e [4].

In [5] è inoltre fornita un'analisi delle funzionalità fornite degli 11 framework più diffusi e delle loro caratteristiche come mostrato in fig1.2.

1.1.2 Hardware

Sebbene la minor presenza di elementi virtuali renda le applicazioni in AR meno complesse computazionalmente rispetto a quelle in VR, le operazioni di tracking e di rendering restano comunque tassanti per l'hardware dei dispositivi sui cui vengono eseguite e possono anche in questo caso creare rallentamente che comprometterebbero l'esperienza dell'utente.

I dispositivi per l'utilizzo delle applicazioni in realtà aumentata si possono dividere in due categorie:

1. Dispositivi hand-held
2. Dispositivi wearable (o indossabili)
3. Dispositivi di proiezione

I primi sono dispositivi che, come suggerisce il nome, vengono tenuti in mano durante l'utilizzo e utilizzati per inquadrare il mondo circostante attraverso la videocamera, in generale corrispondono agli smartphone e ai tablet, che essendo diffusissimi permettono di accedere a queste tecnologie ad un bacino di utenza molto largo.

L'hardware di tali dispositivi è inoltre particolarmente adatto a questo tipo di applicazioni. Essi possiedono infatti tutte le componenti più importanti per

	General information			Target tracking types					Additional features		
	Extension/Platform	Tutorial	Text	Planar Image	3D Object	Multi Targets	Geo-location	Markerless	Online recognition	Offline recognition	AR Editing Platform
ARToolKit (5.3.2)	Windows, Mac OS, Linux, iOS, Android, Unity Package, Smart Glasses	Yes	No	Yes	No	Yes	Yes	No	No	Yes	No
Augment (3.2.1-1)	App (Android, iOS, Windows, Mac)	Yes	No	Yes	No	No	No	No	Yes	No	Yes
Aurasma (3.5.3)	App (Android e iOS)	Yes	No	Yes	No	No	Yes	No	Yes	No	Yes
BlippAR (2.1.1)	App (Android e iOS)	Yes	No	Yes	No	No	No	No	Yes	No	Yes
CraftAR (3.1.3)	App (Android e iOS), Unity Package, Apache Cordova	Yes	No	Yes	No	No	No	No	Yes	Yes	Yes
EasyAR (1.3.1)	Windows, Mac OS, Android, Unity Package	Yes	No	Yes	No	Yes	No	No	No	Yes	No
Kudan (1.5)	Android, iOS, Unity Package	Yes	No	Yes	No	Yes	No	No	No	Yes	No
LayAR (8.5.3)	App (Android, iOS e BlackBerry)	Yes	No	Yes	No	No	Yes	No	Yes	No	Yes
PixLive (5.6.0)	App (Android e iOS), Apache Cordova e Google Glass	Yes	No	Yes	No	No	Yes	No	Yes	No	Yes
Vuforia (6.2)	Windows, iOS, Android, Smart Glasses, Unity Package	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Wikitude (2.1.0-2.1.0)	App (Android e iOS), Unity Package, Cordova, Titanium, Xamarin e Smart Glasses	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Figura 1.2: Funzionalità delle principali SDK [5]

poterle eseguire: una o più videocamere (i più recenti e/o di fascia alta possono anche utilizzare la tecnologia Lidar per determinare la profondità della stanza dal flusso video), un accelerometro, un giroscopio, uno schermo ed ovviamente un processore e una RAM che potrebbero facilmente essere comparati a quelle dei computer di qualche anno fa.

Il secondo tipo di hardware su cui è possibile installare questo tipo di applicazioni sono quelli indossabili, anche conosciuti come HMD (Head Mounted Displays), sono generalmente costituiti da una o due lenti a seconda del dispositivo e riproducono gli asset 3D proiettandoli sul vetro della lente stessa o attraverso display trasparenti posizionati tra l'occhio e l'occhiale. Questa tecnologia permette un'immersività decisamente maggiore in quanto il casco seguirà pedissequamente gli spostamenti del volto senza dover impiegare del tempo a inquadrare la scena con la videocamera.

Figura 1.3: Esempio di applicazione AR con dispositivo handheld

In molti casi gli HMD incorporano una videocamera che può essere utilizzata per il tracking o, su alcuni dispositivi, per catturare il flusso video in modo similare ai dispositivi handheld e poi trasmetterlo sullo schermo in prossimità dell'occhio insieme al rendering degli ologrammi, in questo caso si ottiene una miglior usabilità in contesti particolarmente luminosi [7] che, normalmente, causerebbero riflessi o un'immagine sbiadita, ma si incrementa la latenza rischiando di provocare motion sickness all'utente. Infine, i dispositivi wearable sono decisamente più costosi della loro concorrenza e questo ne costituisce un grosso svantaggio a livello di diffusione tra i consumatori.

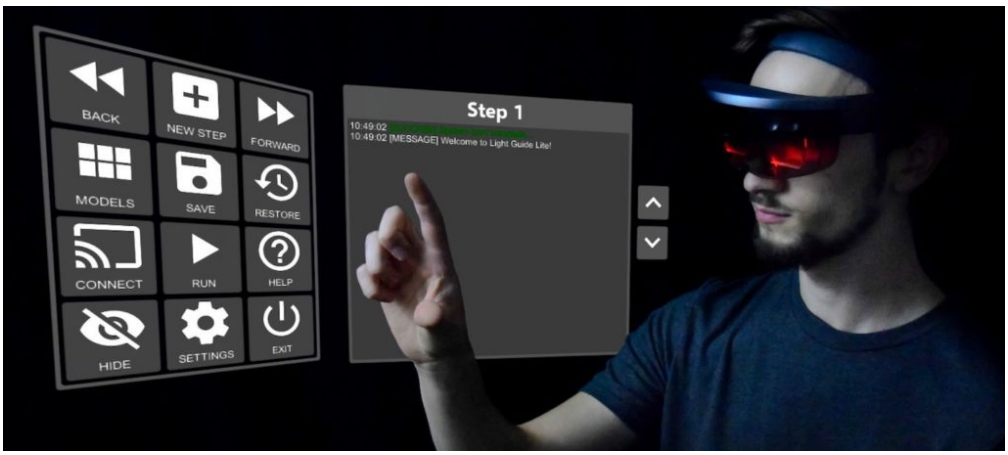

Figura 1.4: Esempio di applicazione AR con dispositivo wearable

L'ultima categoria accorpa tutte le tecniche di projection mapping, o anche projected augmented reality, usata principalmente in ambito industriale e per l'intrattenimento. L'utilizzo di proiettori per la fruizione dell'AR permette di superare quelle che sono le attuali limitazioni degli altri due tipi di dispositivi, in particolare, non essendoci bisogno di tenere fisicamente l'hardware in mano durante l'utilizzo elimina l'ingombro e permette di avere una visione di gruppo delle proiezioni virtuali aumentando la possibilità di collaborazione tra i fruitori. Un altro possibile miglioramento è il fatto che la proiezione non è vincolata a sottostare a problemi di field of view causati dai display head-mounted e non c'è il bisogno di regolare il dispositivo rispetto alla fisionomia di chi lo sta utilizzando in quanto è sufficiente eseguirne il setup solamente la prima volta in base all'ambiente in cui si ha intenzione di utilizzarlo.

La realtà aumentata con proiettori viene utilizzata con successo in diversi ambiti industriali: ne troviamo esempi notevoli nella fase di assemblaggio di veicoli aerospaziali, automobili e schede elettroniche, per i quali vengono applicate tecniche di computer vision e di proiezione per indicare agli addetti dove posizionare i componenti tramite opportuni indicatori luminosi. Anche in medicina viene utilizzata questa tecnica a supporto del chirurgo in sala operatoria per controllare se gli strumenti chirurgici sono puliti e sterilizzati o per mostrare il flusso video delle sonde utilizzate in particolari interventi.

L'intrattenimento è sicuramente un altro campo in cui il projection mapping trova un'importante applicazione ed è infatti spesso adottata per allestire scenografie o veri e propri spettacoli con un vasto pubblico, per il quale le prime due tecniche non troverebbero invece una grande applicazione. In molte città del mondo i proiettori per la realtà aumentata sono stati utilizzati per ricoprire le facciate dei monumenti più celebri Fig.1.5.

1.1.3 Modalità di tracking

Il tracciamento dell'ambiente reale è sicuramente una delle componenti principali nella tecnologia AR e, allo stesso tempo, ne costituisce anche una delle più grandi sfide perché, come detto in precedenza, un buon tracking rende l'esperienza di utilizzo molto fluida, credibile e coerente con le posizioni degli oggetti reali inquadrati.

Tracking con Marker

La soluzione più diffusa è senza dubbio basata su sensori ottici e target fisici quali QRcode o altri riferimenti facilmente riconoscibili.

Nel caso del tracking basato su target, vi possono essere uno o più di questi riferimenti, collocati presso posizioni note ed il tracciamento avviene nel momento

Figura 1.5: Sidney Opera House

in cui vengono inquadrati dalla videocamera e processati dagli algoritmi della SDK. Sicuramente costituisce una soluzione molto pratica ed economica, ma è anche soggetta ad alcune criticità:

1. I target devono essere posizionati precedentemente e se ne deve conoscere la collocazione.
2. Senza i marker non si può utilizzare l'applicazione.
3. La prospettiva dell'utente può determinare una perdita del target se egli si allontana troppo o occlude la visuale della telecamera con degli ostacoli.

Tracking basati su TOF

I sistemi di tracking basati su TOF, o time of flight, sono sistemi che utilizzano un trasmettitore e più ricevitori che ne ricevono i segnali e cercano di determinarne la posizione basandosi sul tempo che il segnale ha impiegato a raggiungerli considerandolo come costante.

Tra le tecnologie impiegate per applicare questo paradigma troviamo:

1. Ultrasuoni
2. Laser

3. GPS
4. Giroscopio ottico

Quando vengono utilizzati gli ultrasuoni troviamo solitamente almeno 3 emettitori sul target da tracciare e altrettanti ricevitori sul sistema di riferimento generalmente costituiti da trasduttori disposti secondo una geometria triangolare. Le posizioni relative tra i ricevitori e tra gli emettitori sono considerate note così come la velocità del segnale, tipicamente 40Khz. I ricevitori e gli emettitori devono essere almeno tre in quanto è il minimo numero di punti necessario a localizzare un piano e con tale piano viene poi determinato l'orientamento del target.

Tra i vantaggi del sistema troviamo il fatto che sia molto leggero e poco ingombrante e che non sia affetto da distorsioni, gli svantaggi sono invece imputabili alla variazione della velocità del suono che è affetta da fattori quali umidità, temperatura e pressione dell'aria e che può comportare imprecisioni nelle misurazioni. Un ultimo svantaggio risiede invece nella perdita di potenza del segnale all'aumentare della distanza.

Il GPS è un sistema di localizzazione diffuso nel mondo dal 1996. Esso si compone di 24 satelliti e 12 stazioni sul terreno. I satelliti hanno un orologio con una deriva di circa 0.1msec all'anno che può comportare errori nella posizione di circa 30km, le stazioni a terra hanno il compito di ricalibrarli ogni 30 secondi. Il sistema può localizzare un trasmettitore target quanto quest'ultimo invia un segnale che raggiunge almeno 3 satelliti basandosi sui rispettivi TOA (Time of Arrivals) dei segnali.

I giroscopi sono generalmente adoperati nelle misurazioni di velocità angolari. I giroscopi ottici utilizzano il principio TOF con raggi laser (RLG) o fibre ottiche (FOG) per determinare la velocità angolare di un target per mezzo di un interferometro. Sono leggeri e consumano poca energia.

Tracking videometrico

Questa tipologia di tracking si basa sull'utilizzo di multiple videocamere piazzate sul target con diversa posizione e orientamento che tentano di riconoscere dei pattern nel ambiente circostante e ne analizzano successivamente la distanza. Generalmente questi sistemi hanno almeno 4 telecamere e vengono utilizzati in spazi preparati appositamente con soffitti o pareti facilmente identificabili.

Tracking con collegamenti meccanici

Il tracking con collegamenti meccanici è solitamente impiegato in sistemi costituiti da un visore di tipo HMD e un braccio meccanico che lo collega a un riferimento generalmente posizionato sul soffitto. Il braccio si muove per seguire i movimenti che l'utilizzatore effettua indossando il caschetto e la posizione è stimata andando ad esaminare le rotazioni dei singoli giunti ed effettuandone la

cinematica diretta. Questa tipologia richiede un grande spazio per essere utilizzata e può causare fastidio nel movimento poichè il braccio meccanico eserciterà sempre una piccola resistenza nel seguire il percorso, il tracking risulta però molto preciso.

Tracking con campo magnetico

Questa tecnologia utilizza una spira nella quale viene fatta passare una corrente sinusoidale che genera un campo magnetico che funge da riferimento e un ricevitore che svolge il ruolo di target. Il campo magnetico generato dalla spira crea delle correnti nei ricevitori la cui intensità dipende dalla distanza dal generatore e dalla rotazione rispetto allo stesso.

Per ottenere una misura accurata della posizione e dell'orientamento di un target occorrono quindi ricevitori composti da tre spire conduttrici perpendicolari l'una all'altra. Il tracking così ottenuto è molto preciso ma non è possibile utilizzarlo in spazi molto ampi in quanto l'intensità del campo magnetico decade con la distanza del target.

Localizzazione e mappatura simultanei

La genesi di questa tecnologia avvenne nel 1986 [8] presso la IEEE Robotics and Automation Conference di San Francisco, dove Smith and Cheesman [10] e Durrant-Whyte [9] proposero un approccio probabilistico per stimare le coordinate di alcuni landmark su una mappa. L'ambiente circostante viene quindi mappato di continuo e ne viene approssimata una ricostruzione in 3D, sul cui riferimento vengono poi applicati gli elementi virtuali dell'applicazione. Viene utilizzato per esempio dalle Hololens di Microsoft.

Infine, questo approccio può anche essere accoppiato al primo in modo da colmarne le lacune, infatti è possibile creare un tracking esteso che è in grado di superare i limiti della perdita del target in quanto quando essa avviene si inizia a stimare le modifiche che stanno venendo applicate alla posizione e rotazione del sistema e viene mantenuto in piedi il sistema AR. Quando poi verrà reinquadrato il target si procederà ad azzerare tutte le eventuali derive indotte dall'approssimazione.

1.1.4 Domini di applicazione

La Realtà Aumentata è sicuramente una tecnologia molto versatile e la sua diffusione sta diventando sempre più capillare anche grazie alla presenza di dispositivi adatti ad ospitarla alla portata di tutti, ad oggi è utilizzata negli ambiti più disparati [11][12] che spaziano da quello medico a quello industriale fino ad arrivare al gioco e all'architettura.

Di seguito una veloce panoramica delle sue principali applicazioni.

Istruzione e addestramento

La possibilità di visualizzare e manipolare i modelli 3D può rappresentare un grande passo in avanti nella didattica di molte materie: si può pensar,e per esempio, a quanto la possibilità di vedere e ruotare a proprio piacimento il modellino di una molecola potrebbe semplificare la comprensione della sua struttura atomica o di quanto potrebbe essere affascinante, nell'ambito della biologia, mostrare una proiezione tridimensionale di un determinato organismo animale o vegetale durante la lezione.

Negli ultimi anni questa possibilità si sta concretizzando e, in molte scuole, delle applicazioni in realtà aumentata sono fornite ai professori come ulteriore strumento di insegnamento e senza dubbio può costituire una utile ed affascinante aggiunta alla didattica tradizionale, specialmente tra gli alunni più giovani, abituati ad averci a che fare già da bambini.

Gli esempi più riusciti e diffusi sono i laboratori virtuali [14][13], in cui si possono svolgere esperimenti per cui non si possiederebbe la strumentazione adatta ed è possibile anche l'interazione tra gli studenti [15] ed i cosiddetti *libri magici* in cui spesso sono presenti, insieme alle tradizionali immagini, anche contenuti in AR.

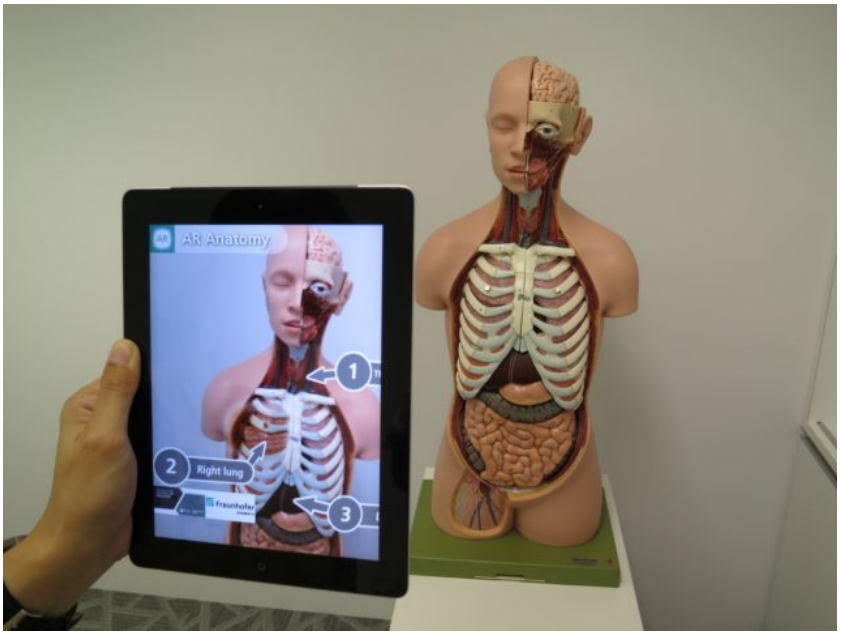

Figura 1.6: AR nello studio dell'anatomia

Particolare diffusione si ha inoltre nell'addestramento di diverse figure professionali in quanto la visualizzazione tramite AR dei vari step delle procedure da svolgere semplifica notevolmente l'apprendimento delle stesse e la velocità con cui si può preparare una persona a svolgere un determinato compito. Tra i compiti

che più beneficiano di tale tecnologia troviamo sicuramente quelli di assemblaggio, le procedure di sicurezza e quelle di manutenzione o riparazione.

Cultura e Intrattenimento

In modo analogo a quanto avviene per i libri con inserti virtuali, sempre più spesso musei e parchi offrono contenuti in Realtà Aumentata per aumentare l'immersività della visita. Questi contenuti possono essere di varia natura: video, descrizioni testuali o ricostruzioni di manufatti, paesaggi o elementi architettonici antichi da integrare alle reali esposizioni per coinvolgere maggiormente i visitatori e ad approfondire il contesto artistico o storico in esame.

Sempre parlando del contesto culturale non è possibile non citare i tour virtuali, che permettono a chi non ha la possibilità di recarsi fisicamente presso la mostra di vivere quest'esperienza da remoto, potendo muoversi liberamente tra le varie aree.

Per quanto riguarda l'intrattenimento invece, la diffusione degli smartphone ha dato la possibilità agli sviluppatori di creare applicazioni in Realtà Virtuale di particolare successo implementando nuove meccaniche di gioco: grazie al posizionamento degli ologrammi nel mondo reale è infatti possibile integrare una meccanica di esplorazione reale che permetta agli utenti di muoversi nel mondo reale e interagire con gli elementi 3D disseminati per la mappa. Le novità apportate con questi giochi si sono, per esempio manifestate nel fenomeno Pokémon GO Fig.1.7 nel 2016 la cui popolarità ha sicuramente fatto da apripista per molte altre future applicazioni.

Figura 1.7: Pokémon GO

Architettura e Costruzioni

Va da sé che la visualizzazione di un ambiente sia molto più comprensibile ed immediata per mezzo di una ricostruzione tridimensionale piuttosto che con i disegni tradizionali e questo è uno dei motivi per cui senza dubbio il settore delle costruzioni ha beneficiato grandemente dell'avvento della Realtà Aumentata.

Figura 1.8: Anteprima di un sito di costruzione

Si pensi a quanto possa essere più appagante poter vedere preventivamente i risultati di una ristrutturazione con dei modelli 3D posizionati coerentemente nella scena reale e sapere quindi quali modifiche sono di nostro gradimento e quali invece preferiremmo ritoccare, o all'eventualità di esaminare una casa senza essere lì veramente così da scegliere più velocemente quelle a cui si è davvero interessati.

Ma le applicazioni in questo campo non sono tutte qua: durante i lavori di costruzione di un edificio è possibile illustrare a schermo la posizione di elementi preesistenti come tubature di acqua e gas o cavi elettrici e costruire tenendo conto di queste informazioni evitando danni e preservando il lavoro già svolto [16]; infine c'è la possibilità di modificare il progetto prima di avere iniziato l'effettiva costruzione qualora ci si accorga dalla ricostruzione che esso non incontra i requisiti che ci si era prefissati in fase di design [17].

Marketing

Anche nel marketing l'AR viene impiegata spesso per integrare o sostituire la fase di progettazione e prototipazione di un determinato prodotto, che, generalmente, sono le più lunghe e costose rendendole più efficienti; è infatti molto più semplice progettare il prodotto lavorando su modelli 3D molto realistici in quanto forniscono una visualizzazione quasi pari ad un prototipo fisico, ma, al contempo, rendono estremamente semplici e economiche le modifiche. Tali modelli, di solito, vengono anche inviati al cliente cosicché possa farsi un'idea del prodotto finale e individuare delle criticità che desidererebbe venissero risolte.

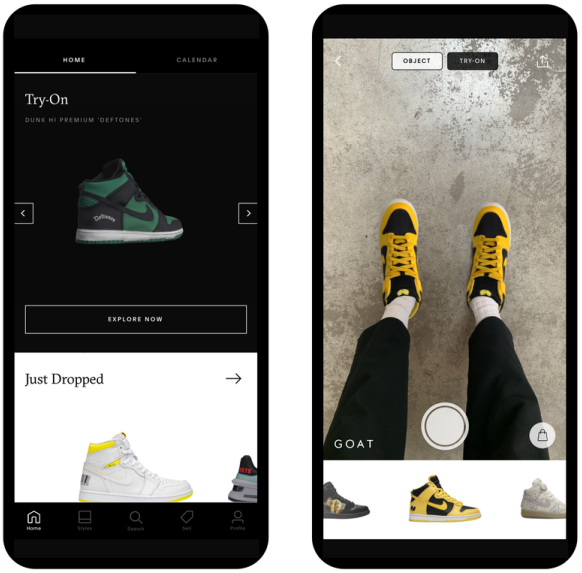

Figura 1.9: Esempio di fitting room virtuale

Ognuno di noi inoltre ha, almeno una volta, utilizzato un catalogo online su qualche sito di e-commerce e per alcune categorie merceologiche vengono fornite non sono le immagini dell'articolo, ma anche la sua ricostruzione tridimensionale con la possibilità di essere ruotata e traslata all'interno della pagina come se ci si trovasse in un negozio vero. Alcuni market online dispongono anche dei camerini virtuali nei quali l'utente, per mezzo della fotocamera e degli algoritmi di riconoscimento del volto o del corpo, possono indossare i loro capi preferiti [18] Fig.1.9.

Infine, ultimamente, anche i negozi fisici e alcuni bar o ristoranti si stanno munendo di listini prezzi e menù digitali da leggere scannerizzando QR code forniti ai clienti così da non doverli stampare e tenerli sempre aggiornati.

Medicina

La Realtà Virtuale ricopre un ruolo importante nelle procedure chirurgiche permettendo ai medici di avere una migliore visione dell'area in cui stanno operando e, di conseguenza, muovere gli strumenti in modo più preciso [19]. Data l'estrema precisione richiesta, tutte le componenti che fanno parte della pipeline della tecnologia AR devono essere impeccabili [20], in particolar modo il tracking, che deve permettere al sistema di raggiungere una precisione dell'ordine del decimo di millimetro. Sono inoltre in fase di test procedure per permettere a specialisti di operare a distanza.

Figura 1.10: Intervento chirurgico con ausilio di un'interfaccia AR

Turismo

L'AR è un formidabile alleato per ogni viaggiatore in quanto permette di estrapolare moltissime informazioni dall'ambiente circostante con l'unico requisito di possedere uno smartphone dotato di fotocamera. Ci si può per esempio orientare in una città che non si conosce o tradurre segnali e cartelli in un'altra lingua come nel caso di Google Lens.

L'esempio più diffuso di AR per il turismo avviene nel settore alberghiero, nel quale spesso le brochure e i volantini vengono arricchiti con elementi digitali da scannerizzare con l'applicazione dell'albergo per mostrare video o ricostruzioni 3D. Queste app sono spesso utilizzate anche per la lettura delle mappe cittadine e delle principali attrazioni turistiche che vengono esposte proprio all'interno dell'hotel per informare i clienti su quali possano essere i migliori itinerari e i servizi consigliati dalle altre persone.

Per incrementare la popolarità di una meta turistica, di frequente si ricorre allo sviluppo di applicazioni specifiche per una località, che, imitando ciò che fa un guida turistica, fornisce ai viaggiatori tutte le informazioni di cui hanno bisogno per muoversi nella zona. Di quest'ultima categoria fa parte, per esempio, Florence Travel Guide, software dedicato ai visitatori del capoluogo toscano.

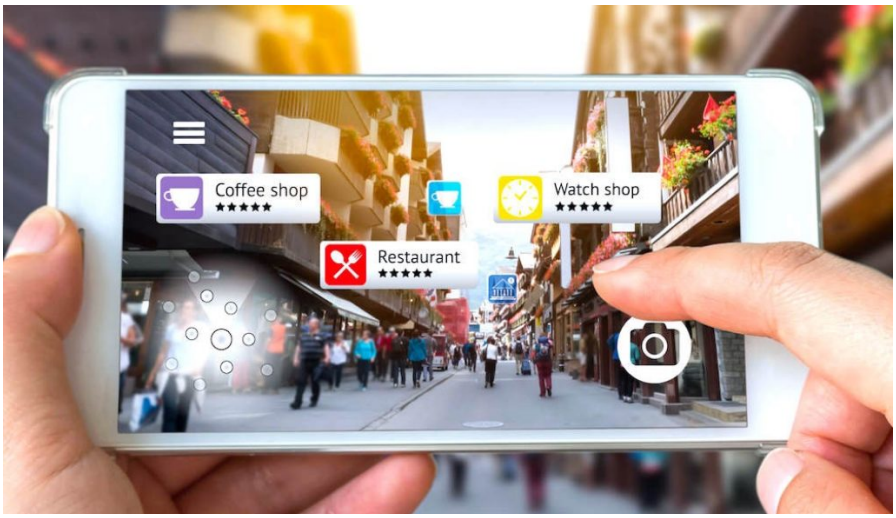

Figura 1.11: Navigazione in Realtà Aumentata

Militare

In campo militare si utilizza la Realtà Aumentata sia per l'addestramento dei soldati sia durante le missioni vere e proprie.

Nel primo caso vengono chiamate BARS (Battlefield Augmented Reality System) e consistono nella simulazione di uno scenario bellico realistico in cui i militari dovranno svolgere degli incarichi. L'addestramento virtuale permette un addestramento più frequente e molto più economico in quanto ricostruire la scena 3D richiede un dispendio di risorse concretamente minore rispetto a mobilitare veri mezzi militari, inoltre può simulare territori diversi da quelli in sta avvenendo il training.

Figura 1.12: Esercitazione militare supportata da AR

Nel secondo caso invece l'impiego di questa tecnologia ha forti criticità dovute alla difficoltà di tracciare correttamente e in fretta l'ambiente circostante e al corretto riconoscimento di alleati, nemici, obiettivi e minacce. Per tali ragioni, i dispositivi che vengono impiegati devono essere molto accurati e performanti. I più impiegati sono gli smart glass in quanto non necessitano di essere manovrati con le mani e seguono in modo fedele i movimenti della testa dell'utilizzatore.

Industriale

Il settore industriale è tra quelli che più di tutti ha sviluppato tecniche all'avanguardia nella Realtà Virtuale e la applica a più livelli nelle diverse fasi della filiera produttiva.

Manutenzione, riparazioni e assemblaggio sono le mansioni che più di tutte hanno ricevuto un vantaggio dall'impiego delle risorse virtuali. Spesso può essere necessario memorizzare procedure molto lunghe e il fallimento di queste ultime può inficiare la corretta produzione di un prodotto ed essere pericolosa per gli addetti che si trovano nelle vicinanze dei macchinari, è proprio per questi motivi che nel traning e nello svolgimento di questo tipo di protocolli vengono impiegate applicazioni AR [21]: possono infatti guidare l'operatore nello svolgimento dei vari step indicando le istruzioni nella giusta sequenza man mano che si procede o mettere in contatto chi sta eseguendo l'incarico con un esperto in remoto condividendo la visuale dell'area operativa [22]. Questo permette a chiunque di svolgere processi di manutenzione e riparazione di guasti aumentando perciò la velocità e la sicurezza con cui si possono eseguire.

Figura 1.13: Esempio di manutenzione e riparazione

1.2 ROS

ROS o Robotic Operating System [23] è un framework che contiene strumenti utili a sviluppare e distribuire software per Robot insieme alle funzionalità tipiche di un sistema operativo come i driver dei dispositivi, i mezzi di testing e le primitive di comunicazione tra processi e calcolatori diversi.

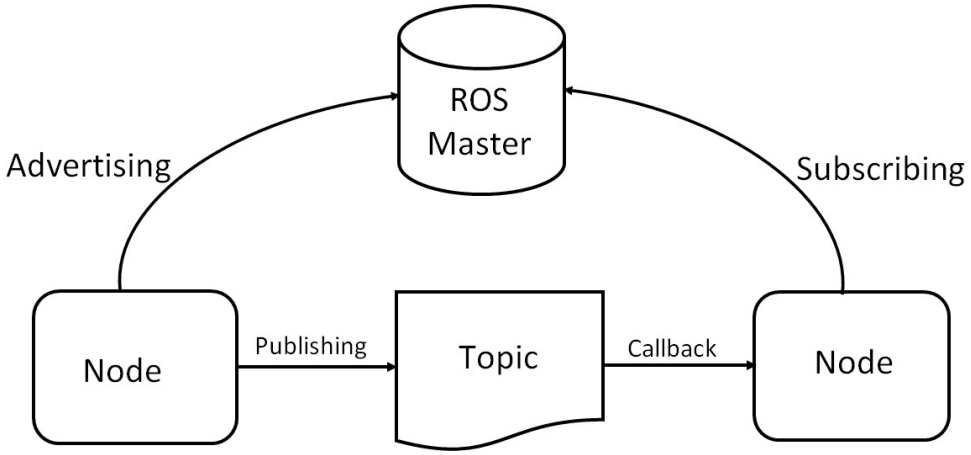

Figura 1.14: Comunicazione in ROS tra master e nodi

Quest'ultima funzionalità è la più importante poiché permette di mettere in rete diverse macchine ROS e di farle comunicare in modo semplice. La comunicazione avviene tra i nodi, pacchetti creati in ROS che contengono programmi e le relative dipendenze e il master, il nodo centrale della rete attraverso il quale gli altri possono comunicare e scambiare messaggi.

Come da Fig.1.14 i nodi dichiarano al master quali tipi di messaggi (*Topic*) sono interessati a ricevere o inviare e quest'ultimo di occupa di fornire le informazioni necessarie a raggiungere gli altri nodi che stanno utilizzando gli stessi topic. Se la comunicazione avviene su macchine separate viene utilizzato il protocollo TCPROS, che è una versione modificata dello standard TCP/IP.

I package di ROS, che a loro volta possono essere raggruppati in stack, contengono nodi, messaggi e servizi insieme a un manifesto in XML che ne descrive nome, versione, dipendenze e metadati.

Sebbene ROS sia disponibile solo su Unix permette di scrivere i propri nodi in diversi linguaggi e di farli comunicare indipendentemente da come essi siano stati scritti Fig.1.15; in particolare sono disponibili le librerie *rospy* per Python e *roscpp* per C++, ma ci sono anche librerie che fanno da ponte con Java e JavaScript, rispettivamente *rosjava* e *roslibjs* (o *rosnodejs*)

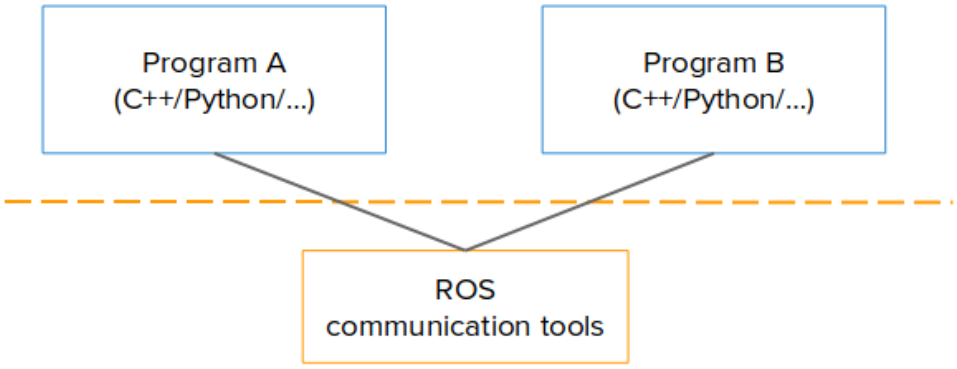

Figura 1.15: ROS con diversi linguaggi di programmazione

Le primitive di comunicazione fornite in ROS sono:

1. Topic
2. Servizi
3. Azioni

Il primo permette di mandare messaggi tra i nodi in flussi diversi a seconda dell'argomento, il secondo crea un paradigma client/server sincrono tra i nodi ed il terzo fornisce la possibilità di formare un architettura client/server asincrona. Ci possono inoltre essere multipli nodi sottoscritti allo stesso Topic.

1.3 Manipolatori

I manipolatori sono robot che permettono a un sistema di interagire con l'ambiente circostante autonomamente. La loro alta precisione e ripetibilità, unitamente a un minor costo hanno portato ad un loro impiego sempre maggiore nell'ambito industriale in luogo degli operatori umani.

La struttura di un robot è composta da una base, saldamente ancorata al suolo in una posizione predefinita attraverso cui si può risalire precisamente alla sua posizione ed orientamento nello spazio e diversi link, che sono le parti rigide che lo costituiscono. Questi elementi sono poi collegati dai giunti che possono essere di vario tipo, principalmente rotatori o prismatici a seconda del movimento relativo che permettono.

1.3.1 Manipolatori a polso sferico

Tra i manipolatori più utilizzati in ambito industriale ci sono quelli a polso sferico Fig.1.16, il cui nome deriva dalla somiglianza con la struttura ossea del braccio umano. Essi sono formati da sei giunti di cui, contando dalla base, i primi due vengono chiamati giunti di spalla, il terzo di gomito ed il quarto, quinto e sesto formano il polso. Quest'ultimo può essere a sua volta definito come monocentrico qualora gli assi di rotazione dei tre giunti si incrocino in un punto solo.

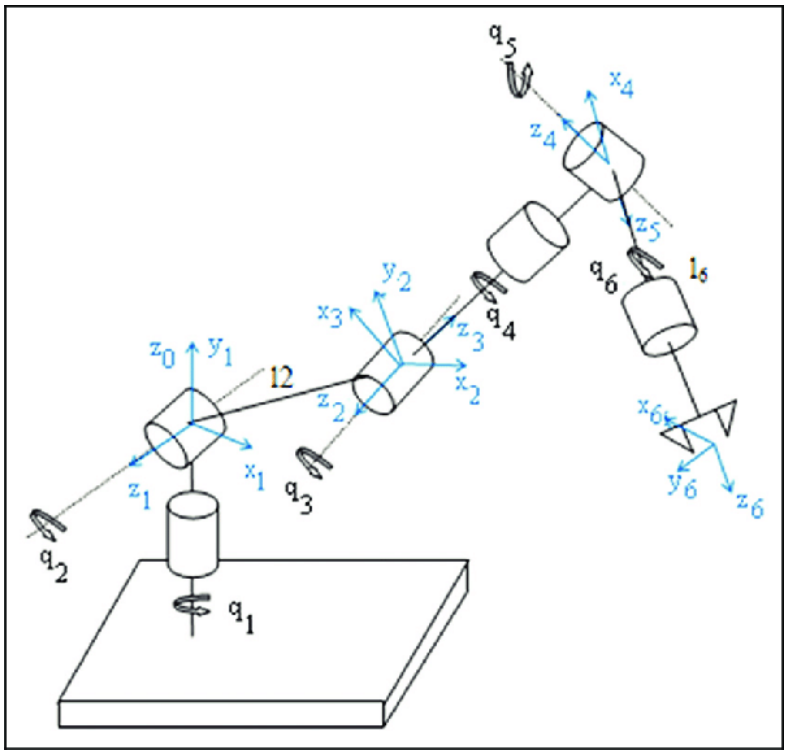

Figura 1.16: Manipolatore antropomorfo a polso sferico. Immagine tratta da [24]

1.3.2 e.DO

e.DO è un robot antropomorfo a polso sferico prodotto da Comau S.p.a la cui logica di controllo è gestita da un Raspberry Pi installato nel link di base. È un manipolatore modulare in quanto è possibile agganciare diversi strumenti all'end effector, come ad esempio una pinza, e fargli svolgere task diversi.

La scheda Raspberry Pi utilizza Raspbian Jessie OS e la versione di ROS installata è ROS Kinetic; la comunicazione con altre macchine, quali tablet o calcolatori, avviene tramite interfacce Ethernet o Wi-Fi, mentre quella con i giunti che compongono il braccio tramite USB.

Figura 1.17: e.DO. Immagine tratta da [25]

Specifiche	Valore
Numero di assi	6
Carico massimo	1 kg
Massima estensione	478 mm
Asse 1	+/- 180 deg 22.8 deg/s
Asse 2	+/- 99 deg 22.8 deg/s
Asse 3	+/- 99 deg 22.8 deg/s
Asse 4	+/- 180 deg 33.6 deg/s
Asse 5	+/- 104 deg 33.6 deg/s
Asse 6	+/- 180 deg 33.6 deg/s
Peso totale	11.1 kg
Peso del braccio	5.4 kg
Materiale della struttura	Ixef 1022

Tabella 1.1: Tabella delle specifiche da [25]

Ogni giunto è dotato di una propria scheda, motore e sensore e può ruotare in entrambe le direzioni lungo il proprio asse entro i limiti della struttura del giunto. Le specifiche tecniche sono fornite da Comau e riportate in Tabella 1.1.

e.DO Gripper

L'end effector installato sul polso del manipolatore è una pinza con due dita,

Fig.1.18. La presenza di un giunto prismatico e di un motore permette alla pinza di aprirsi e chiudersi come si può vedere nelle Figure 1.19 e 1.20 con le relative misure in mm.

Figura 1.18: Pinza e.DO

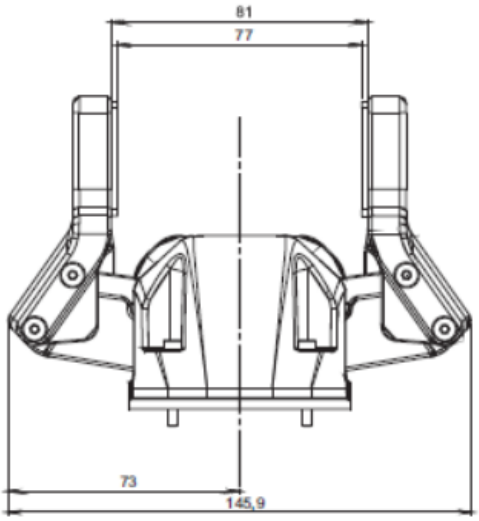

Figura 1.19: Pinza aperta (mm)

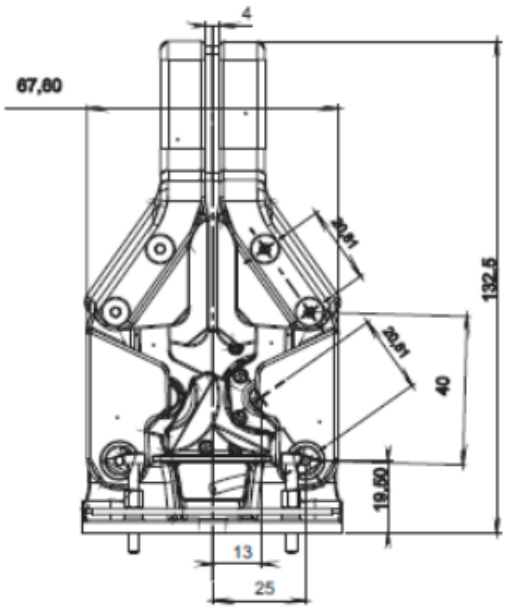

Figura 1.20: Pinza chiusa (mm)

Descrizione	Caratteristica
Type	Gripper
Weight	200 g
Stroke	With neoprene fingertip cover 77 mm
Payload	400 g
Opening/closing maximum speed	80 mm/s
Clamping force	¡65 N
On-board sensors	Encoder
Comau Part No.	CR82442100

Tabella 1.2: Tabella delle specifiche fornita da Comau

1.3.3 Moti rigidi

In questa sezione saranno illustrati alcuni degli aspetti di base delle trasformazioni nello spazio.

Posa

Conoscere la posizione e l'orientamento dell'end effector è fondamentale per raggiungere delle posizioni nello spazio. Dato un sistema di riferimento, si possono descrivere le coordinate di un corpo rigido tramite la sua posizione $\mathbf{x}$(t)e orientamento $\mathbf{a}$(t):

$$p(t) = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{a}(t) \end{bmatrix} \tag{1.1}$$

Traslazione

Il termine moto rigido indica lo spostamento di un corpo rigido e può essere descritto come la somma di una traslazione lungo una linea e di una rotazione su un'asse parallelo ad essa.

Dato un vettore $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$, definiamo la traslazione lungo una direzione data dal vettore $\mathbf{t} = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$ come $Trasl(\mathbf{v}, \mathbf{t})$:

$$Trasl(\mathbf{v}, \mathbf{t}) = \begin{bmatrix} v_1 + t_1 \\ v_2 + t_2 \\ v_3 + t_3 \end{bmatrix} = \mathbf{v} + \mathbf{t} \tag{1.2}$$

Rotazione

Data una matrice quadrata $\mathbf{R}$ e un vettore generico $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$ l'operazione di rotazione è descritta come $Rot(\mathbf{v}, \mathbf{R})$:

$$Rot(\mathbf{v}, \mathbf{R}) = \mathbf{R}\mathbf{v} \tag{1.3}$$

Per effettuare multiple rotazioni, ognuna rappresentata da una matrice $Rot(\mathbf{u_i}, \theta_\mathbf{i}) = \mathbf{R}_i^{i-1}$ che si riferisce al sistema di riferimento ottenuto dopo aver effettuato la rotazione precedente, possiamo definire:

$$Rot(\mathbf{u_1}, \theta_\mathbf{1}) Rot(\mathbf{u_2}, \theta_\mathbf{2}) \dots Rot(\mathbf{u_N}, \theta_\mathbf{N}) = Rot(\mathbf{u}, \theta) \tag{1.4}$$

E la matrice che rappresenterà la rotazione complessiva sarà

$$Rot(\mathbf{u}, \theta) = \mathbf{R_N^0} = \mathbf{R_1^0 R_2^1} \dots \mathbf{R_N^{N-1}} \tag{1.5}$$

1.3.4 Cinematica

La cinematica permette di rappresentare posizioni velocità e accelerazioni del manipolatore. Una serie di giunti e bracci forma una catena cinematica, nel caso ideale, priva di massa, attrito ed elasticità. A seconda di quanti giunti la compongono questa catena avrà un certo numero di gradi di libertà (DOF) e gradi di movimento (DOM).

Giunti

Un giunto permette un grado di libertà tra due elementi collegati e può avere due diverse strutture:

1. Rotoidale
2. Prismatico

L'end effector è l'ultimo componente della catena cinematica ed è quello per mezzo del quale il manipolatore effettua i suoi compiti. Il centro dell'end effector, anche chiamato Tool Centre Point (TCP), è il punto che il software muove nello spazio. Se c'è un solo percorso dalla base al Tool Centre Point la catena cinematica viene definita aperta, se ce ne sono molteplici viene chiamata chiusa.

Working space

Lo spazio di lavoro può essere il Task space, cioè l'insieme di posizioni che il Tool Centre Point può raggiungere nello spazio, o il Joint space, rappresentato dalle variabili dei giunti q_i.

A seconda dei gradi di libertà del TCP e del task da svolgere, rispettivamente indicati con n' e n è possibile definire tre casi:

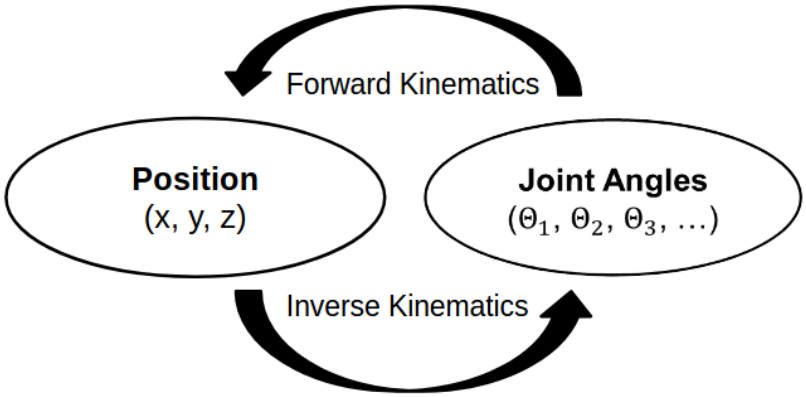

Figura 1.21: Trasformazione da un sistema di riferimento all'altro

1. $n' = n \rightarrow$ Il robot ha tanti DOF quanti sono richiesti dal task $\rightarrow$ Catena non ridondante
2. $n' > n \rightarrow$ Il robot ha più DOF di quelli richiesti $\rightarrow$ Catena ridondante
3. $n' < n \rightarrow$Il robot ha meno DOF di quelli richiesti $\rightarrow$ Catena inadatta al compito

Funzioni di cinematica

Ci sono quattro funzioni chiamate funzioni cinematiche che trasformano le variabili dei giunti in posizioni cartesiane:

1. Posizione diretta: dallo spazio dei giunti a quello del task
2. Posizione inversa: dallo spazio del task a quello dei giunti
3. Velocità diretta: dalle velocità dei giunti a quelle nel sistema del task
4. Velocità inversa: dalle velocità nel sistema del task a quelle dei giunti

Il primo passo per applicare queste funzioni è decidere un sistema di riferimento per ogni giunto del robot, dopodiché è possibile applicarle a qualunque livello nella catena cinematica.

Convenzione di Denavit-Hartenberg

Introdotta per la prima volta nel 1955 da Jacques Denavit e Richard S. Hartenberg, questa convenzione permette di definire quattro gradi di libertà tra due

sistemi di riferimento invece dei sei che sarebbero generalmente necessari. I primi due dei quattro che rimangono da definire riguardano le traslazioni, gli altri sono legati alle rotazioni. Tra questi, a loro volta, tre dipendono solamenta dalla struttura geometrica del robot e non variano con il tempo né con il movimento mentre il terzo dipende dal movimento relativo dei due sistemi di riferimento.

Assumendo che $q_i(t)$ sia l'i-esima variabile giunto, b_i l'i-esimo link e $g_i(t)$ l'i-esimo giunto per definire i sistemi di riferimento valgono quindi le seguenti regole:

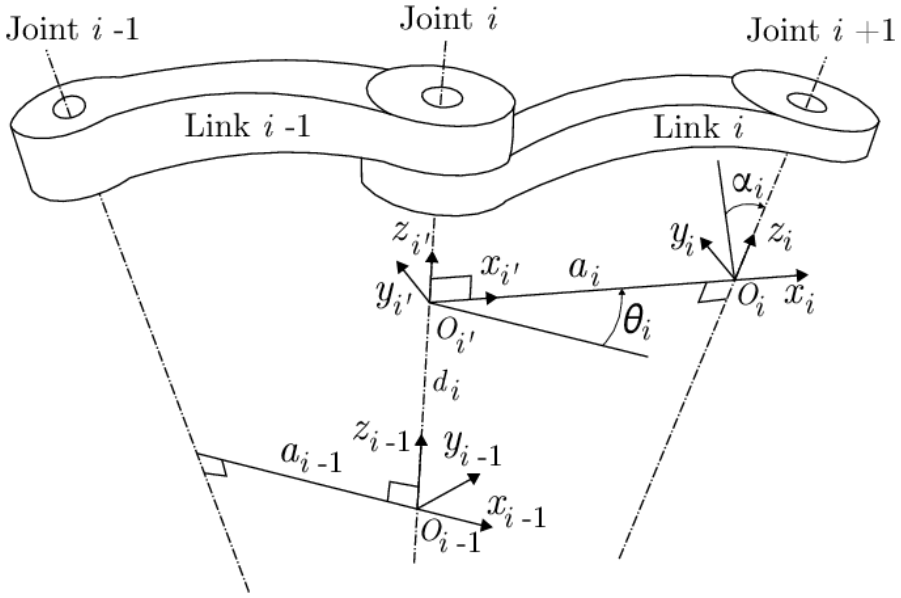

Figura 1.22: Convenzione di Denavit-Hartenberg

1. Il sistema di riferimento dell'i-esimo link si trova sull'asse del giunto g_{i+1}. Nel caso in cui questo asse e quello del giunto precedente si intersechino allora il sistema sarà posizionato in quel punto, altrimenti può essere fissato a piacere (generalmente sul link)
2. Il versore z_i è allineato con l'asse del giunto g_{i+1} con lo stesso verso
3. Il versore x_i è scelto ortogonale a z_{i-1} ed essendo anche ortogonale a z_i costituirà la normale rispetto al piano definito da questi due versori
4. Il versore y_i completa la terna seguendo la regola della mano destra

La trasformazione dal sistema di riferimento della base a quello della fine della catena cinematica è descritto tramite i seguenti parametri:

1. d_i: la distanza dell'asse z_{i-1} dalla normale comune

2. θ_i la rotazione intorno a z_{i-1} per allineare x_{i-1} a x_i
3. a_i la distanza minima tra x_{i-1} e x_i
4. α_i la rotazione intorno a x_i per allineare z_{i-1} a z_i

La convenzione di Denavit-Hartenberg definisce quindi la trasformazione da R_{i-1} d R_i dipendentemente dal tipo di giunto come:

$$Prismatic \rightarrow \begin{cases} q_i(t) = d_i(t) \\ \theta_i, a_i, \alpha_i \text{ sono fissati} \end{cases} \tag{1.6}$$

$$Rotoidale \rightarrow \begin{cases} q_i(t) = \theta_i(t) \\ d_i, a_i, \alpha_i \text{ sono fissati} \end{cases} \tag{1.7}$$

Si può dunque dimostrare che la roto-traslazione è descritta con una matrice omogenea contenente sia i termini rotazionali che quelli relativi alla traslazione:

$$T_i^{i-1} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\cos\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \tag{1.8}$$

Capitolo 2

Stato dell'arte

2.1 Metafore di interazione con l'ambiente virtuale

L'interazione con gli asset virtuali posizionati nelle scena scena di un'applicazione AR costituisce un punto cardine nella credibilità e usabilità del software stesso. A seconda dello scopo per cui è stato sviluppato il programma i comandi con cui l'utente deve approcciarvisi assumono diversa importanza: nel caso in cui gli unici elementi interagibili siano i pulsanti dell'interfaccia grafica, l'interazione risulterà semplice sia da implementare che da utilizzare e non richiederà un particolare design in fase di progettazione, viceversa, qualora ci sia bisogno di spostare e ruotare gli ologrammi con più gradi di libertà essa acquisirà un ruolo decisivo influendo pesantemente sull'esperienza di uso [26].

A seconda della piattaforma a cui è destinata l'applicazione, il modo in cui l'utente interagisce con la stessa può variare fortemente; si può infatti passare dal touchscreen di un tablet al tracciamento dei gesti delle mani come nel caso delle Hololens di Microsoft e in altri dispositivi wearable simili [27].

Figura 2.1: Riconoscimento delle gesture della mano. Immagine tratta da [29]

2.1.1 Mappare spazio 3D sulla superficie tablet

Nel caso in cui si stia sviluppando un'applicazione AR per tablet occorre scegliere con cautela in che modo l'utente potrà interagire con i modelli 3D in quanto, sebbene non insorgano particolari difficoltà nel muovere gli oggetti lungo un piano parallelo allo schermo del tablet, potrebbe diventare particolarmente insidioso il posizionamento di un punto con 6 gradi di libertà, ossia di cui vogliamo cambiare con precisione sia la posizione sia la rotazione lungo qualsiasi asse nello spazio 3D.

Qualora il tablet non possedesse sistemi più avanzati come Lidar [30] o non fosse dotato di multiple videocamere, mancherà una nozione di profondità ed rimarremo ancorati alla bidimensionalità dello schermo. Incorreremo allora sia in un problema di visualizzazione poiché avremo bisogno di un riferimento per determinare quanto vicino o lontano sia situato l'oggetto o la sua posizione relativa ad altri elementi virtuali, sia di un problema di mappatura dei comandi sulle varie gesture o pulsanti che avremo a disposizione.

Nel posizionamento su un singolo piano possono essere sufficienti le operazioni di trascinamento con il dito nella direzione designata, l'utilizzo di gesture con più dita per la rotazione [28][29] e il pinch per lo scalamento, ma se l'obiettivo è scegliere una qualsiasi posizione nello spazio sorge un evidente problema di usabilità dell'interfaccia.

2.1.2 Interazione tramite gesture

Come accennato il precedenza è possibile superare le limitazioni dello schermo in 2D applicando gesti con più dita, in [31] viene proposto un modello chiamato 3DTouch basato solo su gesture con pollice e indice che mira a migliorare altri modelli precedenti quali Sticky-tools [32], Screen-Space [34] e DS3 [33] che impiegavano invece più dita creando confusione nei comandi e una più invasiva occlusione dello schermo causata dalla mano dell'operatore.

In [35] si confronta a 3DTouch un ulteriore sistema chiamato HOMER-S, mutuato da HOMER [36], un metodo di manipolazione di oggetti virtuali in Realtà virtuale tramite raytracing ed adattato per funzionare in un contesto di Realtà aumentata su tablet. Con Homer-S si selezionano gli oggetti con un raggio che viene tracciato perpendicolarmente al tablet nel punto di collisione con il dito e successivamente si può decidere quale tipo di trasformazione si vuole usare cliccando uno degli appositi pulsanti: rotazione, traslazione e scalamento.

2.1.3 Interazione tramite gizmos

Nel progetto dell'applicazione per gestire i percorsi da far eserguire ad un manipolatore è stato scelto un sistema similare a Homer-S in cui si può selezionare un

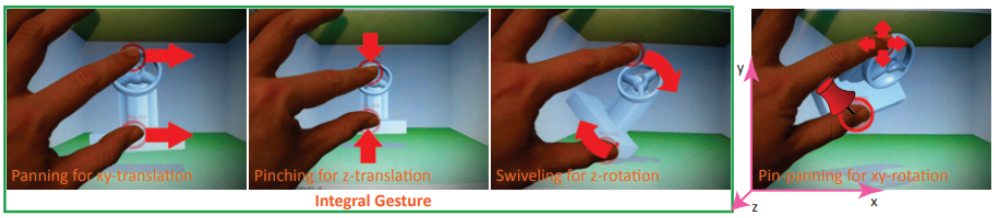

Figura 2.2: Esempio controllo con gesture. Immagine tratta da [31]

ologramma tramite raycasting ed è poi possibile scegliere il tipo di trasformazione con i pulsanti situati nell'angolo dello schermo.

Una volta effettuati questi due primi passaggi vengono mostrate a schermo gli assi su cui effettuare tale trasformazione con delle gizmos simili a quelle presenti nell'editor di Unity, nel caso della traslazione e dello scalamento appariranno infatti gli assi X, Y e Z nel sistema di riferimento del target e nel caso della rotazione ci saranno delle gimbals riferite sempre a tali assi. A questo punto per applicare l'operazione desiderata bisogna trascinare la maniglia corrispondente.

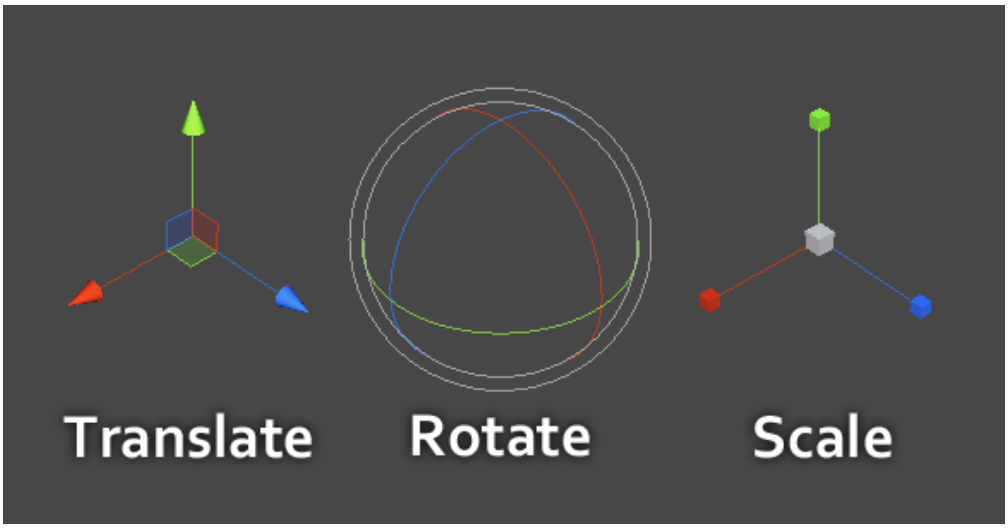

Figura 2.3: Esempio di gizmos o maniglie analoghe a quelle utilizzate

2.2 Interfacce AR nel controllo di manipolatori

Un'applicazione in Realtà Aumentata può contribuire positivamente all'interazione uomo-robot. È infatti molto più intuitivo scegliere posizione e orientamento di un punto avendone una rappresentazione istantanea nello spazio di lavoro reale rispetto a pianificarli via codice.

Proprio per questo l'impiego dell'AR in ambito robotico si sta diffondendo sempre di più sia per il training degli operatori sia per la pianificazione e manutenzione

dei manipolatori industriali [37], facilitando altresì le operazioni da remoto tramite videocamera.

Tra le principali sfide vi sono senza dubbio la precisione del sistema di tracking, che in una reale situazione in fabbrica deve risultare impeccabile per permettere ai manipolatori di svolgere le operazioni di assemblaggio, la mobilità dell'operatore, che deve poter scegliere le traiettorie e i punti da far eseguire al robot senza doversi avvicinare troppo e trovarsi in una situazione di pericolo e il posizionamento dei riferimenti da tracciare: è infatti plausibile che durante il suo percorso il robot possa occluderne la visuale alla videocamera del sistema AR e causare un errore nel posizionamento delle strutture virtuali.

Possono inoltre sorgere altri problemi legati allo specifico task, per esempio, nelle operazioni di pick and place potrebbe non esser nota a priori la forma del prodotto da afferrare o la sua consistenza [38], rendendo difficile la scelta di quanto aprire la pinza o di quanta forza esercitare nella presa.

Il riconoscimento degli oggetti e delle loro dimensioni potrebbe semplificare non solo la fase di raccolta degli stessi, ma sarebbe altrettanto utile per inserire a runtime delle riproduzioni virtuali da intersecare con i punti che compongono il percorso che il robot dovrà eseguire in modo da prevenire eventuali collisioni [39] fermando il processo. Questo varrebbe anche in un ambiente condiviso da più robot in cui potrebbe essere plausibile controllare preventivamente i percorsi e modificarli qualora si trovassero a collidere.

Infine è particolarmente utile inserire la Realtà Aumentata nel controllo dei robot mobili [41] e dei droni [42] in quanto durante lo svolgimento delle loro attività potrebbero uscire dal campo visivo degli operatori; in questo caso risulta particolarmente utile manovrarli da remoto e controllare che cosa sta succedendo.

2.2.1 Soluzioni più recenti

Di seguito riporto alcune delle più recenti soluzioni nell'ambito della programmazione di manipolatori con l'ausilio di interfacce in Realtà Aumentata.

Chacko e Kapila in [38] propongono un approccio marker based con un dispositivo handheld sul quale viene installata l'applicazione la quale permette di svolgere semplici compiti di pick and place segnalando la posizione degli oggetti rispetto alla base del braccio robotico. Il tracking avviene tramite un singolo target costituito da un QRCode posizionato in prossimità dell'ancoraggio del robot e sullo schermo del dispositivo che sta riproducendo il programma viene immediatamente mostrata una zona a mezzaluna corrispondente alle posizioni raggiungibili per completare tale task. Viene inoltre mostrato un mirino al centro dello schermo che deve venire allineato con gli oggetti reali in modo da piazzare delle ancore rappresentate da modelli 3D nella posizione desiderata. Tale posizione deve necessariamente essere situata sul piano di lavoro in quanto il metodo di posizionamento consiste in un semplice raycast che passa perpendicolarmente al dispositivo nel centro del mirino

e va a determinare un punto di collisione con un piano virtuale che combacia con il tavolo reale. È inoltre disponibile un secondo pulsante che permette di annullare il posizionamento. L'applicazione permette di svolgere il task in due maniere differenti:

1. Singolo oggetto
2. Oggetti multipli

Nella prima modalità si possono scegliere solo due posizioni, una rappresentata da un simbolino blu e una con un simbolo rosso, le quali indicheranno rispettivamente la posizione iniziale in cui andare a prendere l'oggetto e la posizione finale ossia quella in cui andrà poi appoggiato.

Nella seconda invece saranno disponibili un numero a piacere di segnalini blu da posizionare e andranno posizionati in corrispondenza degli oggetti reali che si intende ricollocare. Una volta finita la fase di selezione, il manipolatore inizierà a raggiungere uno per volta gli articoli e li porterà, sempre uno per volta in una posizione particolare determinata a priori.

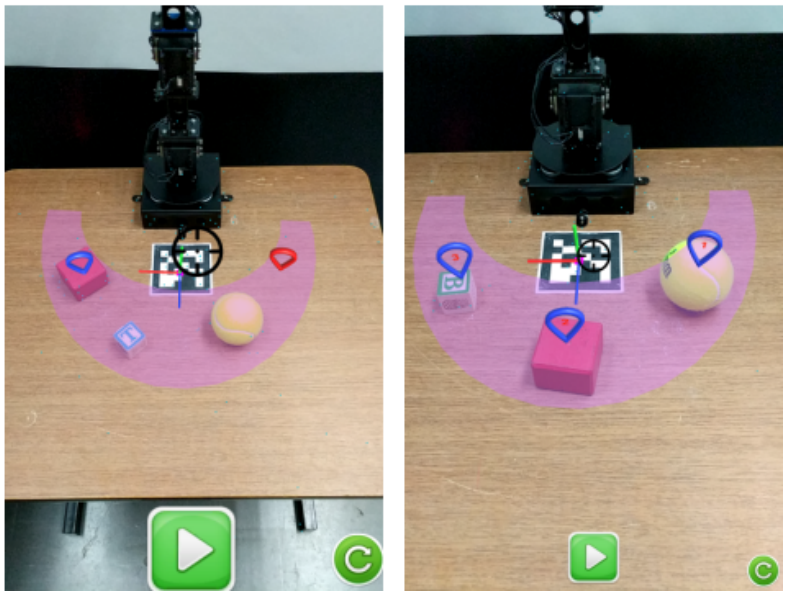

Figura 2.4: Immagini tratte da [38] che illustra le due modalità di utilizzo.

Successivamente nel 2019, in [43], gli stessi autori migliorano il sistema precedente adattandolo al funzionamento per oggetti di forma geometrica ma dimensione ignota aggiungendo un riferimento visivo e due slider nell'interfaccia grafica in grado di scalarlo per andare a indicare le effettive dimensioni di una delle facce dell'elemento da afferrare. Questo permetterà poi al manipolatore di scegliere, se possibile il lato più corto in modo da avere una presa migliore. Questo approccio

è stato poi testato su oggetti di forme diverse quali palline da tennis, lattine e confezioni di creme ed ha ottenuto ottimi risultati.

Nel 2017 invece, Frank, Moorhead e Kapila hanno pubblicato una soluzione simile [44], sempre basata su dispositivi handheld e Realtà Virtuale, ma che differisce nella tipologia sia del robot sia della tecnica di tracking. Nel loro lavoro infatti sono presenti due bracci meccanici e non sono ancorati al tavolo su cui vengono posizionati gli oggetti ma su una struttura centrale ruotati di 90 gradi in modo analogo alle braccia di un essere umano.

Il task da svolgere è molto semplice e consiste nello spostare degli oggetti di forma cubica nota nelle scatole del colore corrispondente e viene conseguito tracciando sia la superficie del tavolo sia i cubetti con dei marker QRCode e tecniche di tresholding e computer vision. Come si può evincere da ciò i marker sono multipli, fino a un totale di 15 per avere un tracking più accurato e sopperire alle eventuali occlusioni causate da altri operatori o dagli stessi bracci e l'impalcatura di metallo che li sorregge. È inoltre presente una griglia trasparente che mostra all'utente le posizioni raggiungibili dai manipolatori e delle versioni virtuali colorate degli oggetti da spostare.

Quest'ultima applicazione rappresenta un'evoluzione rispetto a quella riportata in [45] sempre dagli stessi autori nel 2016 nella quale vi era un singolo manipolatore antropomorfo ed un unico marker per il tracking dell'ambiente piazzato in verticale sulla base del robot. Sempre in [45] era già stata implementata la griglia di riferimento e le destinazioni in cui piazzare gli oggetti erano invece indicate da immagini 2D in realtà aumentata in alcune posizioni raggiungibili.

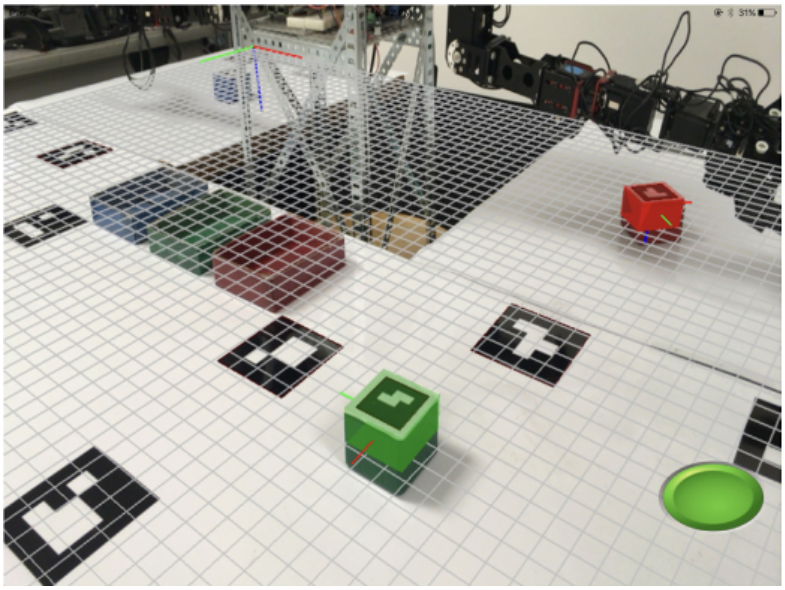

Figura 2.5: Immagine dell'interfaccia AR tratta da [44].

In entrambi i casi, comunque, è possibile muovere gli oggetti solamente sul

piano o su altri cubetti e cioè in posizioni verticali prefissate multiple della loro altezza, questo fa sì che il numero totale di gradi di libertà nel posizionamento dei punti sia inferiore e permette di spostare gli ologrammi AR semplicemente trascinando il dito sullo schermo non essendoci alcuna ambiguità dovuta alla terza dimensione.

In [46] è invece descritto un approccio del 2018 che utilizza la Realtà Aumentata per mezzo di un dispositivo wearable, più in particolare, i Microsoft Hololens. Il robot utilizzato in questo caso è un Barrett Whole-Arm Manipulator (WAM) a 7 gradi di libertà e viene inoltre impiegato un bracciale MYO [47] per migliorare le feature di riconoscimento dei gesti.

Questo sistema è articolato in 6 moduli distinti che costituiscono i passaggi da svolgere per programmare il percorso del manipolatore:

1. Interazione
2. Registrazione
3. Scelta del path
4. Anteprima e modifica
5. Esecuzione
6. Controllo

La prima fase, l'interazione, è fondata sull'utilizzo delle SDK di Microsoft che forniscono all'utente funzioni di riconoscimento vocale e dei gesti delle mani proprio attraverso cui è possibile interagire con i punti in AR e praticare lo spostamento tramite pinch o cambiare la fase in cui ci si trova con gli appositi comandi vocali. È inoltre continuamente mostrato a schermo un cursore che segue il movimento del capo dell'operatore e si può posizionare sopra agli ologrammi.

La seconda consiste nell'allineare il mondo virtuale a quello reale tramite tracking e può essere conseguito per mezzo di marker o allineando manualmente un artefatto 3D con la sua copia reale utilizzando le capacità di scanning dell'Hololens.

Nella terza avviene la creazione del percorso secondo due modalità distinte: percorso sul piano e percorso nello spazio.

Per creare un percorso sul piano viene utilizzato il cursore descritto nel primo punto che posiziona un segnalino sulle coordinate in cui avviene l'intersezione tra un raggio passante per tale cursore e il piano scelto per il task in modo similare a quanto avveniva in [38].

Creare un percorso a piacere nello spazio è un compito decisamente più complicato in quanto aumentano i gradi di libertà con cui si lavora, in questo caso

vengono scelti gli artefatti AR che rappresentano le posizioni in cui si desidera far passare l'end effector del manipolatore ed il sistema elabora un percorso tramite B-spline [48].

La fase di anteprima può essere avviata tramite comando vocale, crea un'animazione del modello virtuale del robot che segua i segnalini creati dall'utente e la visualizza sull'Hololens. Durante questo procedimento è anche permessa la modifica del percorso pinzando gli indicatori che costituiscono il percorso e trascinandoli dove si desidera.

L'esecuzione avviene in modo similare all'anteprima e cioè viene avviata con la voce da chi sta utilizzando l'applicazione con la differenza che questa volta il percorso sarà eseguito dal manipolatore reale che avrà la possibilità di completarlo interamente punto per punto fino alla fine oppure potrà essere gestito nel progresso con l'ausilio del bracciale MYO che, in questo caso, viene applicato per determinare la percentuale di completamento del percorso determinando quindi uno spostamento maggiore o minore della punta del manipolatore.

L'ultima fase è quella di controllo e può avvenire in modalità controllata in posizione [50], più accurata nel seguire il percorso definito ma che rischia di creare forze di collisione troppo grandi, o controllata in impedenza [49][51], più adatta per task che richiedono un'adesione alla superficie in cui la forza di contatto deve rimanere contenuta.

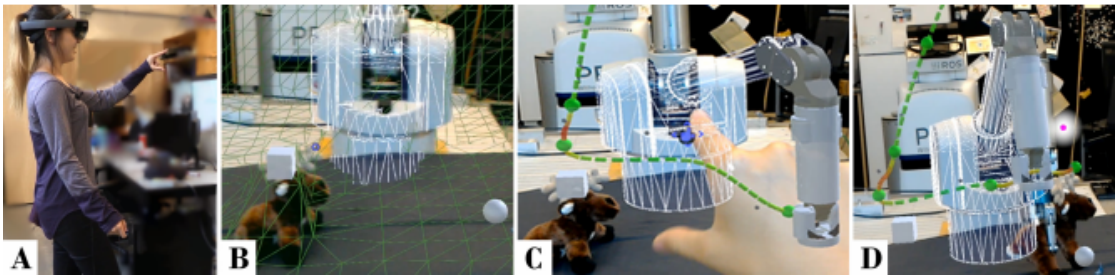

Figura 2.6: Esempio di svolgimento di un compito di pick and place.

In Fig.2.6 è mostrato un esempio dello svolgimento di un compito di pick and place. In (A) l'operatore indossa l'Hololens e utilizza le gesture. In (B) è illustrata la visualizzazione in AR del robot virtuale, della griglia che rappresenta il piano e delle metafore degli oggetti da controllare, in (C) la modifica del percorso tramite gesti. Infine,(D) rappresenta l'esecuzione del percorso da parte di entrambi i robot. Immagine tratta da [49].

Sempre nell'ambito dell'interazione uomo-robot, [52] propone un'interfaccia basata su nodi, ciascuno dei quali è contraddistinto da una determinata operazione da eseguire.

Tali nodi hanno offrono diverse opzioni a seconda di che operazione vi sia stata assegnata, per esempio, il nodo "pick from table" e quello "place" permettono

all'utente di scegliere le posizioni in cui si intende afferrare e trasferire un oggetto facendone apparire una versione in Realtà Aumentata con la conseguente possibilità di spostarlo attraverso due joystick situati nell'interfaccia grafica del touchscreen.

Una volta posizionati i nodi che occorrono per svolgere un task si procede a collegarli, essi infatti dispongono di una o più maniglie, a seconda del proprio utilizzo, da utilizzare per creare una logica nelle istruzioni in modo molto simile a un diagramma a blocchi. Con questo sistema è pertanto possibile, ad esempio, prelevare un elemento da una catena di montaggio, esaminarlo, e sulla base del risultato del punto precedente decidere la posizione in cui esso dovrà essere lasciato.

Infine è permesso anche un certo livello di parallelizzazione nelle singole operazioni, infatti se due operazioni non devono manipolare fisicamente l'oggetto che si sta utilizzando allora possono essere eseguite in contemporanea come nel caso di Fig.2.7 in cui vengono eseguite una raccolta e la stampa di un'etichetta nello stesso momento.

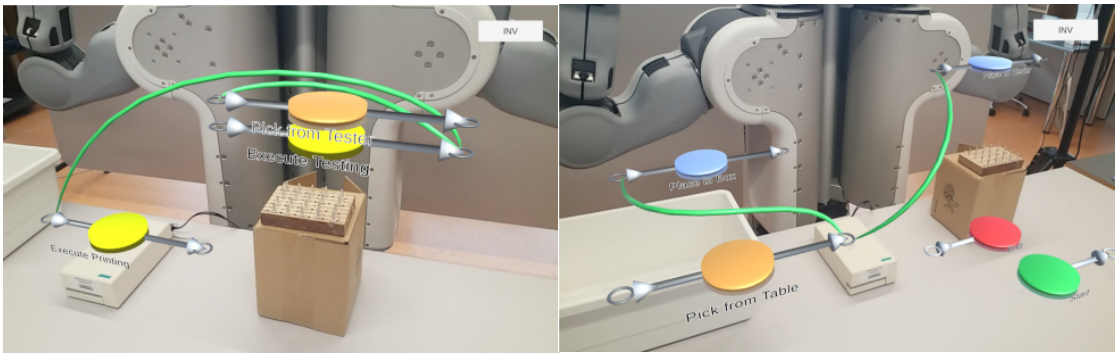

Figura 2.7: A sinistra un esempio di esecuzione parallela, a destra, uno di esecuzione condizionale. Immagini tratte da [52]

Finalità

Osservando i lavori già svolti nell'ambito delle interfacce grafiche per la programmazione di manipolatori si nota che pur permettendo di svolgere task diversi nessuno di essi dà all'utente la possibilità di scegliere posizioni e rotazioni a piacere con 6 gradi di libertà, molte si limitano infatti al solo approccio planare e non contemplano la possibilità che i waypoints da raggiungere siano situati a diverse altezze, quelli che invece considerano questa possibilità tendono a utilizzare altezze fissate e multiple della lunghezza degli oggetti che impiegano nel compito di pick and place in modo tale che l'unico possibile spostamento in altezza corrisponda ad avere tali artefatti impilati uno sopra l'altro.

Per quanto riguarda l'orientamento con cui l'end effector raggiunge le posizioni prefissate, nessuno di questi metodi sembra occuparsene, ecco quindi che questo

lavoro di tesi si tenta di colmare tale lacuna concedendo di scegliere liberamente tutte le possibili combinazioni per ogni singolo punto, sarà poi compito del motore di cinematica inversa andare a segnalare quelle non valide in modo che l'utente sia conscio dell'effettiva area di lavoro del robot.

2.2.2 Casi d'uso

Tra le principali applicazioni della Realtà Aumentata in ambito robotico ricadono come già detto nel campo dell'interazione tra uomo e manipolatore, pertanto si potrebbe scegliere di integrare questa tecnologia per tutte le operazioni in cui degli operatore devono interfacciarsi al robot: dal settore dell'istruzione, in cui l'approccio visivo alla programmazione del robot potrebbe garantire una più facile comprensione della materia trattata, al training degli addetti alla manutenzione, fino ad arrivare alla vera e propria pianificazione del movimento sia sul posto sia da remoto.

Ma l'AR è impiegata con successo anche in medicina con la Robot-Assisted Surgery (RAS), lo sviluppo di protesi e la riabilitazione, nel controllo di Multi-Agent systems negli stessi ambienti di lavoro, nelle teleoperazioni, nelle interfacce aptiche e nel gaming [40].

Il sistema complessivo è stato poi testato nello svolgimento di due task distinti: uno su un percorso planare costituito dalla cancellazione di un disegno su una lavagna bianca e uno su un percorso nello spazio che prevedeva lo spostamento di un pupazzo di pezza in una scatola. Quest'ultimo task prevedeva inoltre la presenza di un ostacolo che impediva la possibilità di creare un semplice percorso rettilineo tra scatola e oggetto da prendere richiedendo quindi una manipolazione più complessa e accurata del path.

Capitolo 3

Tecnologie

In questo capitolo sarà analizzata nel dettaglio l'applicazione AR sviluppata nell'ambito di questa tesi, in particolare verranno descritti i requisiti, le librerie utilizzate per sviluppare il progetto e la relativa architettura.

3.1 Requisiti

La tesi si propone di implementare un'interfaccia in Realtà Aumentata per la pianificazione del percorso di un manipolatore. Il software progettato è destinato a dispositivi handheld quali tablet e smartphone e permette di posizionare i punti di un percorso con 6 gradi di libertà tramite le opportune metafore di interazione, tale software comunicherà inoltre con una macchina ROS con un'architettura client server per calcolare e ricevere la pianificazione completa ottenuta dall'algoritmo di cinematica inversa.

Solo al termine dell'intera comunicazione tra client e server vengono inviate le posizioni calcolate al vero manipolatore permettendo sia di previsualizzare in Realtà Aumentata i punti intermedi che l'end effector andrà ad occupare una volta inviato il comando di movimento sia di rendere l'applicazione indipendente dal robot che si sta utilizzando: il calcolo della cinematica inversa è infatti affidato al plugin di Rviz Moveit che, basandosi sulla descrizione del manipolatore tramite file URDF ne elabora la traiettoria con uno degli algoritmi già implementati al suo interno. Qualora si desideri utilizzare l'applicazione con un altro robot sarà sufficiente inserirne la descrizione sul server ROS e seguire la procedura guidata per il setup di Moveit.

3.2 Vuforia

Vuforia [55] è un SDK di la Realtà Aumentata per dispositivi mobili che permette di sviluppare applicazioni che possono tracciare e riconoscere sia immagini bidimensionali sia oggetti fisici in tempo reale. Supporta sia lo sviluppo nativo per iOS che per Android, ma rende altresì molto semplice creare progetti in Unity che sono facilmente importabili in entrambi gli ambienti. Dispone infine di API per C++, Java e i linguaggi .NET.

Le principali funzionalità fornite sono:

1. Image Targeting: Vuforia permette di identificare e tracciare qualsiasi immagine sufficientemente dettagliata, inoltre fornisce uno strumento di creazione di un database di immagini in PNG e JPG per inserire i propri marker. Ad ognuno degli elementi del database viene anche assegnata una votazione su una scala da 1 a 5 che ne determina la qualità rispetto ai requisiti del sistema di riconoscimento.

2. Tracking esteso: la capacità di mantenere stabile il mondo virtuale qualora l'applicazione non sia in grado di riconoscere il marker. Se il target non si trova più nel flusso video della camera a causa di altri oggetti che ne ostruiscono la visuale o semplicemente perché si sta inquadrando un oggetto troppo lontano, allora Vuforia procede a calcolare le coordinate degli ologrammi utilizzando come riferimento una mappa dell'ambiente circostante precedentemente elaborata in fase di tracking.

   Le derive accumulate in questa fase verranno poi compensate non appena il target sarà nuovamente riconosciuto. Questa feature è particolarmente importante in quanto permette di costruire ambienti virtuali più grandi e di rendere il sistema più robusto alle interferenze causate da elementi che bloccano la visuale.

3. Riconoscimento di oggetti: Vuforia è in grado di riconoscere e tracciare anche oggetti tridimensionali reali ed il suo funzionamento è tanto più accurato quanto più l'oggetto è di forma geometrica, solido ed opaco. Grande importanza è anche data all'illuminazione dell'ambiente che deve essere il più possibile uniforme per evitare riflessi.

Le prime due funzionalità sono state utilizzate per sviluppare questo software e permettono di posizionare correttamente una versione simulata del robot e gli altri elementi virtuali utilizzati per posizionamento e manipolazione dei punti nello spazio.

Data la dimensione del robot, che misura circa un 1,20m in altezza, si è rivelata di fondamentale importanza la funzionalità del tracking esteso, senza la quale non

sarebbe stato possibile inquadrare correttamente tutto l'ambiente senza causare grosse derive nel tracciamento.

Il manipolatore è posizionato su un piano sulla cui superficie è stata stampata una tavola Fig.3.1 con le indicazioni di diversi punti disposti lungo cerchi concentrici a distanze fisse e multiple di 10cm dal centro del manipolatore e.DO. La posizione del manipolatore è determinata dal riferimento esagonale presente sulla tavola che è della stessa forma e dimensione della base di e.DO mentre invece l'immagine a colori è il target dell'applicazione AR.

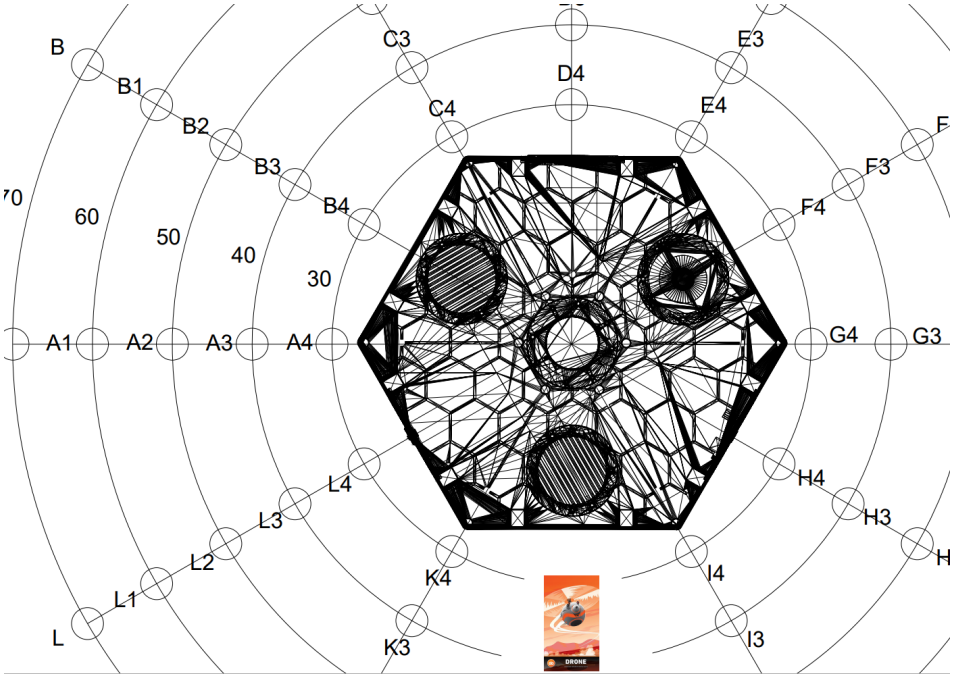

Figura 3.1: Riferimento stampato sul piano del manipolatore

Al fine di migliorare il tracking è stato inoltre incrementata la dimensione dell'immagine di riferimento seguendo quelle che sono le best practice elencate sul sito ufficiale di Vuforia secondo cui la minima dimensione che dovrebbe avere il tracker si deve misurare in rapporto alla distanza con cui la videocamera lo andrà ad inquadrare e dovrebbe sempre essere maggiore di 10. Per esempio inquadrando il target da un metro e mezzo, esso dovrebbe misurare almeno 15cm di larghezza.

Per quanto riguarda invece la scelta dell'immagine è stata utilizzata una di quelle di default fornite dalla SDK che viene valutata dall'algoritmo presente sul sito con la votazione massima, che dovrebbe riflettersi nella massima efficienza in fase di tracking.

3.3 Unity 3D

Unity 3D [56] è l'ambiente di sviluppo scelto per l'implementazione dell'applicazione. Questo Software permette infatti di costruire con grande semplicità gli ambienti 3D grazie al suo editor e alla sua interfaccia grafica e di importare modelli da altri software di modellazione 3D come per esempio Blender convertendoli in FBX. Altro grande vantaggio portato da Unity è quello di rendere possibile la distribuzione su più piattaforme diverse quali Android, iOS e Windows.

Inoltre la grande popolarità di Unity, sia in ambito accademico che in quelli industriale e videoludico insieme al fatto che sia opensource fa si che siano disponibili una grande quantità di librerie e packages da utilizzare al suo interno.

3.3.1 ROS Sharp

ROS Sharp [57] è un set di librerie e strumenti opensource in C# per far comunicare applicazioni .NET, in particolar modo applicazioni Unity, con macchine ROS.

Il pacchetto per Unity 3D contiene tutti gli asset necessari a implementare un Rosbridge Client con cui interfacciarsi a un server ROS e l'URDF importer che rende facile la condivisione delle descrizioni testuali dei robot ed il loro inserimento nell'ambiente virtuale lato Unity.

In particolare vengono forniti dei componenti da aggiungere ai GameObject che forniscono le primitive di comunicazione necessarie a contattare il Rosbridge Server.

ROS Connector

Il ROS Connector Fig.3.2 è uno script C# che permette di selezionare un serializzatore, un protocollo di comunicazione e un indirizzo IP con la porta su cui si trova il server oltre che un tempo di timeout in secondi al termine del quale rinuncerà a contattare ROS qualora ci siano problemi di connessione.

La connessione avviene nella fase Awake di Unity, che avviene non appena la scena è stata caricata ed è chiamata una sola volta per ogni GameObject nel suo ciclo di vita, e lancia un nuovo thread che attiva un socket per connettersi all'IP desiderato. È necessario che questo componente si trovi su ogni GameObject che desideri avere al suo interno un publisher o un listener di topic ROS.

Per quanto riguarda il protocollo di rete da utilizzare è necessario selezionare Web Socket Sharp, mentre per il serializzatore serve selezionare Microsoft per pubblicare e Newtonsoft_JSON per ricevere.

Proprio quest'ultima necessità ha comportato il bisogno di creare due diversi ROS Connector all'interno della scena per comunicare con la workstation del server in quanto non è possibile cambiare il serializzatore a runtime né utilizzare due diversi collegati allo stesso Connector, pertanto, il tablet verrà considerato

dal Rosbridge Server come un doppio client e attiverà due connessioni, una per pubblicare e una per sottoscrivere i topic ROS.

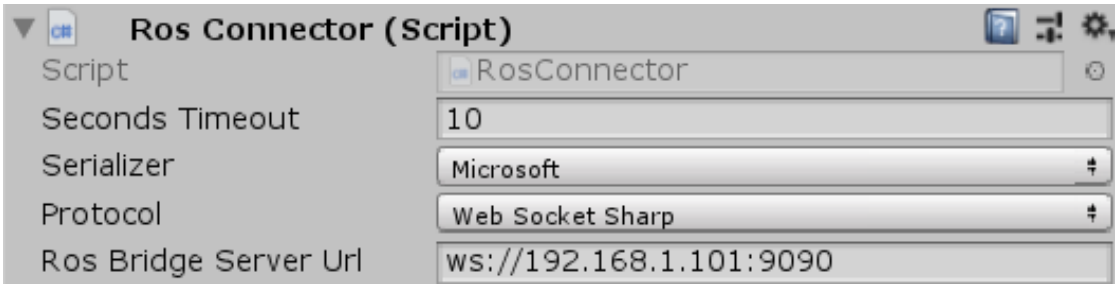

Figura 3.2: ROS Connector in Unity

Unity Publisher

Lo Unity Publisher è uno script contenuto in ROS Sharp che fornisce un'interfaccia per pubblicare un certo tipo di messaggio su un topic.

Si possono scrivere script che sfruttano i concetti legati all'ereditarietà delle classi per implementare le funzioni del publisher e si può definire via codice o dall'editor di Unity il che topic vogliamo sottoscrivere. Nel caso di messaggi custom sarà anche necessario creare una classe che contiene tutti i campi che sono necessari a ROS per capire cosa contiene e accettarlo.

Unity Subscriber

Analogamente al publisher, ROS Sharp mette a disposizione uno script per creare dei listener, anche in questo caso si può scrivere classi custom che ereditano dallo Unity Subscriber e si mettono in ascolto sull'IP e sulla porta selezionati nel componente del connettore aspettando di ricevere messaggi pubblicati con il topic selezionato. Così come per il publisher è richiesto che sul GameObject che ospita questa classe sia posizionato anche il Connector in quanto verranno utilizzati sia i campi di IP e Protocol sia alcuni metodi come Advertise e Publish.

A differenza del publisher però il subscriber verrà utilizzato su un nuovo thread, creato in fase di Start di Unity, che si accerterà, dopo un congruo tempo di timeout, che la connessione al server abbia dato esito favorevole, dopodiché si metterà in ascolto con la funzione Subscribe.

Siccome a tale funzione viene passata anche la funzione astratta ReceiveMessage come parametro sarà necessario eseguirne l'override per implementare la logica più opportuna al nostro tipo di messaggio.

Messaggio *"geometry_msgs/Pose"*

I messaggi inviati dall'apllicazione Unity a ROS sono di tipo *geometry_msgs/Pose*, un tipo già implementato nella libreria ROS Sharp che contiene un Vector3 con

le coordinate x,y e z e un quaternione con le informazioni sulla rotazione. Tali messaggi vengono mandati dal publisher sul topic /geometry_msgs/Pose e vengono utilizzati per mandare, uno per volta, i punti che devono entrare a far parte della pianificazione.

Messaggio *"moveit_msgs/DisplayTrajectory"*

I messaggi di tipo DisplayTrajectory invece sono custom di Moveit e non sono pertanto implementati di base dalla libreria, pertanto è stato necessario crearne una versione in C#.

La classe ottenuta seguendo il modello dalla wiki di Moveit contiene diversi campi il più importante dei quali è senza dubbio il vettore di RobotTrajectory al cui interno c'è un'altra classe chiamata JointTrajectory che a sua volta contiene un vettore di punti per ognuno dei quali sono note le rotazioni da applicare ai giunti del manipolatore per ottenere un punto nello spazio.

Questo tipo di messaggio viene inviato dal server al client nel momento in cui viene finito di calcolare il plan del nostro percorso e contiene le posizioni ai giunti non solo per i punti che abbiamo fornito, ma anche per le posizioni intermedie calcolate dalla cinematica inversa, il cui numero può variare in base alla distanza da percorrere.

Messaggio *"edo_core_msgs/MovementCommand"*

MovementCommand è un messaggio custom utilizzato per comunicare con il robot reale e.DO, analogamente ai messaggi DisplayTrajectory contengono le informazioni di cinematica diretta per applicare le rotazioni ai giunti, ma contiene altri campi come l'intero ovr che determina la velocità con cui verranno movimentati i giunti tramite un valore da 0 a 100, l'intero MovementCommand e MoveType che vengono utilizzati per scegliere la modalità di movimento del robot e JointMaskFlag che corrisponde a quale tipo di end effector è montato sul manipolatore.

MovementCommand viene inviato sul topic */bridge_move* dal client a e.DO, il quale dovrà aver attivato il Rosbridge Server in fase di avvio e inizierà ad eseguire le operazioni di movimento non appena li riceverà.

Capitolo 4

Architettura

4.1 Architettura Client Server

L'architettura del sistema Fig.4.1 di basa su un modello client server in cui il client è costituito dall'applicazione sul tablet e il server è la macchina ROS che avvia il Rosbridge Server per mettersi in ascolto.

In questo caso, essendo presente anche il manipolatore e.DO, ci sono due server con i quali il tablet deve comunicare, non essendo infatti possibile accedere al risolutore della cinematica inversa proprietario del manipolatore è stato necessario adottare una soluzione in cui i percorsi vengono calcolati sulla macchina ROS e successivamente inviati tramite cinematica diretta al robot vero e proprio.

Non è infatti possibile fornire dei punti da calcolare al manipolatore senza che quest'ultimo inizi a percorrerli consecutivamente ed essendo la previsualizzazione del path uno degli elementi costitutivi di questa applicazione si è dovuta per forza adottare un'altra strada.

Bisogna anche considerare che, come detto nella sezione riguardante ROS Sharp, il tablet incorpora due diversi Connector per poter usufruire sia in ricezione sia in invio delle funzionalità di comunicazione venendo pertanto considerato come due client distinti dal server e la versione di ROS installata su e.DO non consente multiple connessioni al Rosbridge per motivi di sicurezza.

La comunicazione avviene quindi in due fasi ben distinte di cui la prima avviene tra tablet e server ROS e consiste nella scelta più appropriata di punti per comporre il percorso, la relativa pianficazione e previsualizzazione tramite risolutore IK di Moveit e la successiva modifica nel caso in cui l'utente non fosse soddisfatto del risultato finale e la seconda, tra tablet ed e.DO, che implica il solo invio della lista di posizioni intermedie al manipolatore da applicare ai giunti in cinematica diretta.

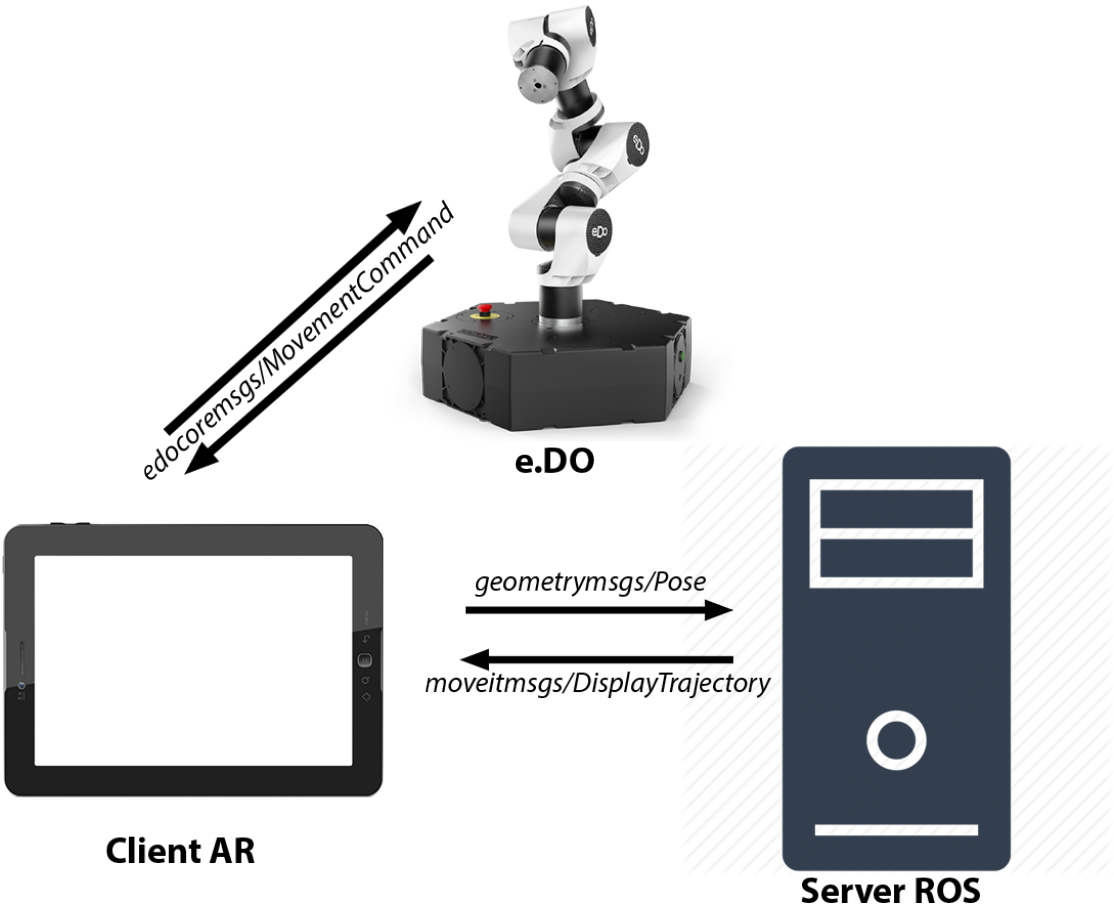

Figura 4.1: Architettura del sistema

4.2 Server

Il server è una macchina Linux su cui sono installati Ubuntu versione 16.04.7 LTS (Xenial Xerus) e la versione Kinetic Kame di ROS insieme agli interpreti di Python.

All'avvio il server deve attivare il Rosbbridge Server tramite il comando roslaunch:

```
$ roslaunch rosbridge_server rosbridge_websocket.launch
```

Che mette in ascolto il server sull'IP e la porta desiderati. Di base i valori di questi campi sono 0.0.0.0/9090 ma è possibile cambiarli in qualsiasi momento andando a modificare il file rosbridge_websocket.launch nella cartella del package rosbridge facilmente raggiungibile con:

```
$ roscd rosbridge_server
```

Oltre al websocket è necessario eseguire anche il motore di cinematica inversa tramite il plugin di Rviz Moveit. La versione di Moveit installata è quella per ROS Kinetic e il comando per lanciare il programma è:

```
$ roslaunch [custom_robot_package] demo.launch
```

Laddove [*custom_robot_package*] sta ad indicare il package di ROS creato seguendo la procedura MoveIt! Setup Assistant che richiede di fornire il file URDF del proprio robot, generare la matrice delle self-collision, creare i giunti virtuali e i gruppi di planning e infine di salvare delle pose tipiche come quella di base con tutti i giunti nella posizione di partenza.

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

Infine una volta attivati i due programmi precedenti è necessario lanciare lo script di Python che è stato creato per gestire la logica nella ricezione, elaborazione e risposta dei messaggi ROS ricevi dall'applicazione tablet.

```
$ rosrun [script_package] ik_input.py
```

È importante notare che è necessario aver fatto partire per primi Moveit e Rosbridge in quanto nei file launch è nascosta la chiamata a Roscore, una collezione di nodi necessari al corretto funzionamento di ogni altro elemento ROS.

A questo punto il server è inizializzato ed è possibile far partire l'applicazione del client per cominciare a scambiare i messaggi.

4.2.1 Logica del server

Il comportamento del server è gestito dallo script uno script di Python che implementa una classe chiamata MoveGroupInterface al cui interno sono istanziate le variabili necessarie al controllo del robot simulato in Moveit:

```
moveit_commander.roscpp_initialize(sys.argv)
rospy.init_node('ik_input', anonymous=True)

robot = moveit_commander.RobotCommander()
scene = moveit_commander.PlanningSceneInterface()
group_name = "edo"
move_group = moveit_commander.MoveGroupCommander(group_name)
```

Sostanzialmente le prime due linee servono a inizializzare il nodo Rospy e il Moveit Commander che successivamente servirà per simulare il manipolatore, la terza istanzia la variabile robot che contiene tutte le informazioni sullo stato corrente del manipolatore ed il suo modello cinematico, la quarta crea la scena e permette di modificare la vision che il robot simulato ha dell'ambiente virtuale e l'ultima prepara il MoveGroupCommander, che è al centro della logica del programma in quanto proprio attraverso questa classe è possibile andare a impartire i comandi di planning ed esecuzione.

Il nome "edo" assegnato al group_name e passato al costruttore del Commander è lo stesso che è stato scelto in fase di setup di Moveit ed è anche contenuto nell'URDF del robot stesso, nel caso in cui si utilizzasse un manipolatore differente esso dovrebbe essere ovviamente modificato in modo di conseguenza.

```
display_trajectory_publisher = rospy.Publisher(
                                        '/move_group/display_planned_path',
                                        moveit_msgs.msg.DisplayTrajectory,
                                        queue_size=20)


plan_wrong_publisher = rospy.Publisher(
                                      '/move_group/plan_went_wrong',
                                      moveit_msgs.msg.DisplayTrajectory,
                                      queue_size=20)

display_execution = rospy.Publisher(
                                    '/move_group/display_execution',
                                     moveit_msgs.msg.DisplayTrajectory,
                                     queue_size=20)
```

Queste tre righe instanziano i ROS Publisher assegnando a ciascuno di essi un topic specifico, un tipo di messaggio e la dimensione massima della coda di messaggi che è possibile conservare.

Al termine dell'inizializzazione delle variabili della classe viene chiamata la funzione listener():

```
def listener(self):

     rospy.Subscriber("/geometry_msgs/Pose", Pose,
        self.callback)
     rospy.Subscriber("/geometry_msgs/FinalPose", Pose,
        self.callback2)
     rospy.spin()
```

Questa funzione attiva i due ROS Subscriber e li sottoscrive al topic specificato per un certo tipo di messaggio che in questo caso è comune ad entrambi ed è quello di Pose contenuto nel package geometry_msgs, viene inoltre fornito un puntatore a una funzione per elaborare ciò che viene ricevuto ed è possibile creare funzioni diverse per gestire in modo differente le informazioni legate ad ogni argomento.

La funzione rospy.spin() crea un loop infinito mantenendo il server in ascolto con i suoi Subscriber fintanto che non riceve un segnale di shutdown.

La prima callback si occupa di processare i messaggi del topic "/geometry_msgs/Pose" che corrispondono ai punti inviati da tablet per i quali è necessario eseguire il planning attraverso la relativa funzione plan_cartesian_path():

```
def plan_cartesian_path(self, pose):

  rospy.loginfo("plan_cartesian_path")

  position = pose.position
  quat = pose.orientation

  move_group = self.move_group

  waypoints = []
  wpose = pose

  move_group.set_pose_target(wpose, end_effector_link="link_6")

  plan = move_group.plan()

  if (len(plan.joint_trajectory.points) == 0):
      plan_wrong_publisher = self.plan_wrong_publisher
      wrong_trajectory = moveit_msgs.msg.DisplayTrajectory()
      plan_wrong_publisher.publish(wrong_trajectory)

      return plan, -1


  waypoints.append(copy.deepcopy(wpose))

  stato = RobotState()
  join = JointState()

  join.header = Header()
  join.header.stamp = rospy.Time.now()
  join.name = ["joint_1", "joint_2", "joint_3",
            "joint_4", "joint_5", "joint_6"]
```

```
join.header.frame_id = "/base_link"

for numero in plan.joint_trajectory.points[-1].positions:
    join.position.append(numero)

stato.joint_state = join
move_group.set_start_state(stato)
fraction = 1

return plan, fraction
```

A plan_cartesian_path() viene passato come argomento la posa da raggiungere composta dalle coordinate cartesiane e un quaternione ad indicarne l'orientamento nello spazio 3D.

Viene poi chiamata la funzione set_pose_target() della MoveCommander alla quale sono passati il nome dell'ultimo link del braccio sul quale andrebbe montato l'end effector e la posa da raggiungere, successivamente si utilizza la funzione plan() sempre della classe MoveCommander per ottenere le posizioni intermedie dal motore della cinematica inversa e lo si conserva nella variabile plan.

Una volta ottenuta la pianificazione del percorso si procede con un controllo sulla sua effettiva dimensione e, qualora esso risulti vuoto, implicando il fallimento dell'algoritmo di IK viene inizializzato un messaggio di errore e mandato sul topic "/move_group/plan_went_wrong" per informare il client che il processo non è andato a buon fine. La funzione fa ritorno alla callback con il plan, che in questo caso sarà vuoto e con un numero intero chiamato fraction che avrà valore -1 e indicherà l'insuccesso.

Se la fase di planning va a buon fine si inizializza un nuovo messaggio di tipo RobotState nel quale vengono inserite le coordinate e le informazioni di rotazione dell'ultimo punto raggiunto nel percorso e lo si utilizza per chiamare la funzione del MoveCommander set_start_state() che applica una nuova configurazione di partenza al manipolatore simulato in modo che alla prossima chiamata della funzione plan esso cominci a calcolare il path a partire dall'ultima posizione trovata e non dallo stato di riposo del robot.

In base al valore di ritorno di plan_cartesian_path() la callback inserirà o meno il plan ottenuto nella lista globale che contiene tutti i plan calcolati.

È importante sottolineare che, contrariamente a quanto avviene in caso di fallimento, il messaggio contenente il plan ottenuto con successo non deve essere inviato manualmente via codice ma viene già pubblicato dalla funzione plan() con un messaggio di tipo DisplayTrajectory del pacchetto moveit_msgs sul topic "/move_group/display_planned_path".

La seconda callback contenuta nello script gestisce i messaggi riguardanti il topic "/geometry_msgs/FinalPose" che possono essere inviati dall'applicazione client per 4 diversi scopi a seconda del valore contenuto:

1. Cancellare la lista di plan ottenuta finora
2. Riportare il robot virtuale nella posizione di riposo
3. Riportare il robot virtuale nella posizione di riposo
4. Eseguire la lista di plan sul manipolatore reale

Nel caso in cui si debbano eseguire le pianificazioni effettuate in precedenza sul robot reale viene chiamata la funzione sendPts() che prende ogni pianificazione contenuta nella lista e procede ad inviarla sul topic "/move_group/display_execution":

```
def sendPts(self):

   rospy.loginfo("sendPts")
   robot = self.robot
   display_execution = self.display_execution

   for p in myPlans:
      display_exe = moveit_msgs.msg.DisplayTrajectory()
      display_exe.trajectory.append(p)
      display_execution.publish(display_exe)
```

Oltre a inviare uno ad uno i plan già ottenuti si svuota la lista e si aggiorna l'ultima posizione valida raggiunta dal robot in modo che se uno dei successivi percorsi non dovesse andare a buon fine sarebbe possibile ritornarvi.

Invece per ritornare alla posizione di partenza è sufficiente creare un nuovo messaggio di tipo RobotState ed applicarlo al robot virtuale tramite la funzione set_start_position() della classe MoveCommander già vista in precedenza, solo che in questo caso verranno inseriti come posizioni all'interno di JointState un vettore di 6 elementi float uguali a 0 ad indicare le rotazioni nulle per tutti i giunti.

```
stato = RobotState()
joint = JointState()

joint.header = Header()
joint.header.stamp = rospy.Time.now()
joint.name = ["joint_1", "joint_2", "joint_3",
```

```
                "joint_4", "joint_5", "joint_6"]

    joint.header.frame_id = "/base_link"
    joint.position = [0.0,0.0,0.0,0.0,0.0,0.0]

    stato.joint_state = joint
    move_group.set_start_state(stato)
```

Tra i metodi forniti dalla classe MoveCommander compaiono anche set_goal_tolerance() che determina l'accuratezza con cui il motore IK cercherà di raggiungere i punti forniti e set_planning_time() che stabilisce il tempo massimo che è possibile concedere al risolutore per trovare un percorso valido, scaduto il quale viene restituito un percorso vuoto.

Il client è un'applicazione Android sviluppata con la versione 2018 di Unity compatibile con le librerie ROS Sharp, Vuforia e ARCore utilizzate per lo svolgimento del progetto. Quest'ultima libreria, sebbene non sia utilizzata per le sue feature di Realtà Aumentata è necessaria per abilitare le funzionalità di tracking esteso fornite da Vuforia nell'apposita scheda dell'inspector.

Per installare l'applicazione è richiesta, come minimo, la versione di Android 7.0 Nougat (api 24) o una più recente.

4.3 Interfaccia Grafica

Figura 4.2: Interfaccia grafica dell'applicazione.

L'applicazione fornisce diversi pulsanti all'utente per svolgere il proprio task di pianificazione del percorso per il manipolatore, nell'angolo il basso a destra è

presente un bottone per passare dalla modalità di traslazione a quella di rotazione e viceversa che sostituisce le frecce del sistema di riferimento che permettono la traslazione di un oggetto con le gimbal per la modifica dell'orientamento.

Alla sua sinistra è invece situato il pulsante per l'eliminazione degli oggetti che rimuove l'elemento dalla lista di posizioni e successivamente chiama il metodo Destroy() di Unity che distrugge il GameObject dalla scena corrente; per chiamare la funzione di cancellazione tramite il pulsante è necessario avere un oggetto attivo selezionato, altrimenti il bottone sarà semitrasparente e non interagibile.

In basso sul lato sinistro vi sono invece i tre pulsanti dedicati alla comunicazione con il server ROS: il pulsante blu serve per inviare il percorso creato a Moveit e calcolarne il path risultante con il motore di cinematica inversa. Qualora tale path risultasse corretto il pulsante blu verrebbe sostituito da un simbolo di "play" verde con la funzione di richiedere al server ROS tutte le pianificazioni finora ottenute ed inviarle al manipolatore reale.

Il pulsante giallo permette la rimozione multipla dei punti di supporto gialli nel caso ve ne siano, altrimenti cancella tutti i punti posizionati dall'utente, quello di "Home" serve invece per riportare sia il robot reale sia quello simulato nella posizione con tutti i giunti ruotati di 0 gradi.

In alto si trovano i pulsanti per la gestione della visibilità degli elementi di supporto quali griglie, ostacoli e le rappresentazioni degli end effector sui punti. Nel caso degli ostacoli e della griglia viene disattivato sia il Mesh Renderer sia il Collider in modo che non intercettino i tocchi dell'utente sullo schermo nel momento in cui interagisce con una delle posizioni dove si trovavano compromettendo i comandi dell'app.

Per quanto riguarda le rappresentazioni degli ostacoli e le anteprime dell'end effector situati sulle posizioni selezionate dall'utente è necessario che esse siano attivate nel momento in cui si va ad inviare il percorso al server o al manipolatore in quanto vengono utilizzati per calcolare le collisioni che il reale end effector andrebbe ad effettuare nel suo tragitto. Se per caso fossero disattivati l'applicazione provvederebbe da sé ad avvertire l'utente tramite Android Toast e ad attivarli in autonomia.

Sempre riguardo alle metafore che descrivono i punti del percorso, è disponibile uno slider sul lato sinistro dello schermo che ne può variare la dimensione così da evitare occlusioni e sovrapposizioni di oggetti, da notare che tale funzionalità non ha alcun effetto sulle riproduzioni dell'end effector in quanto esse servono proprio ad indicarne la posizione nello spazio ed è necessario che mantengano sempre le proporzioni reali.

Infine, gli ultimi due pulsanti, compaiono solamente quando vi è un punto selezionato e servono per permettere l'inserimento di punti in posizioni intermedie del percorso. Per esempio avendo un percorso costituito da tre punti sarà possibile creare un punto intermedio tra il primo e il secondo con il pulsante "Insert Before" e uno tra il secondo e il terzo con il pulsante "Insert After".

4.4 Manipolazione dei punti

Posizionamento

Nella scena virtuale sono presenti alcuni oggetti deputati alla creazione di nuovi punti, quali un piano orizzontale semitrasparente che ricopre l'intera superficie della tavola 3.1 e una griglia verticale con linee ogni 10cm, tale funzione può essere estesa ad altri oggetti, come ad esempio gli ostacoli, applicandovi il tag "floor" nell'editor di Unity.

Il vero e proprio posizionamento dei punti avviene tramite doppio tap sul touchscreen del tablet in corrispondenza del quale viene fatto passare un Raycast in direzione perpendicolare allo schermo, se questo raggio collide con uno degli elementi appena descritti allora sarà salvata la posizione della collisione inserendo nella scena una sferetta blu con una piccola freccia verde ad indicare la direzione del vettore right dell'oggetto nel suo sistema di riferimento locale all'interno di Unity, questo vettore è fondamentale in quanto l'algoritmo di cinematica inversa farà arrivare la punta del manipolatore proprio in corrispondenza di esso nell'orientamento corrispondente.

La freccia verde è un GameObject a sé stante, assegnato come "child" alla sfera in modo da copiarne le trasformazioni di traslazione, rotazione e scalamento in fase di modifica, inoltre, di base, il sistema farà comparire queste frecce sempre in direzione perpendicolare alla superficie scelta dell'utente.

Anteprima del percorso

Ognuno degli oggetti creati in questo modo viene inserito all'interno di una lista di GameObject che viene utilizzata sia per le funzionalità di calcolo sul server sia per visualizzare un percorso provvisorio nell'ambiente virtuale. Questo percorso provvisorio viene elaborato in tempo reale da un algoritmo di Catmull-Rom [53] e mostrato all'utente tramite un componente di Unity chiamato LineRenderer, che accetta una lista di posizioni spaziali e le interpola con una linea.

La linea che rappresenta il percorso è una semplice approssimazione fintanto che il percorso non è stato effettivamente calcolato dal server, successivamente, utilizzando le posizioni intermedie ricevute, esso può essere considerato uno schema alquanto preciso del cammino finale della punta dell'end effector.

Nel caso in cui ci fossero problemi con il percorso, come ad esempio il posizionamento di un punto troppo vicino ad un ostacolo, la linea, che di base è gialla, diventerebbe rossa in modo da avvertire l'utente dell'errore. Il planning di un percorso raggiungibile applicherà invece un colore verde alla linea.

Modifica dei punti già presenti

I punti presenti nella lista di GameObject possono essere selezionati dall'utente tramite singolo tap ed attivano così alcune funzioni che prima non erano disponibili:

1. Cancellazione singola
2. Inserimento in posizioni intermedie
3. Metafore di interazione

La prima non fa altro che rimuovere l'oggetto dalla lista e distruggerlo nella scena, mentre la seconda permette di inserire punti tra altri già presenti, in particolare, ciò che viene creato in questo modo sarà sempre posizionato al centro del vettore che unisce i due punti già presenti tranne nel caso in cui non ci sia uno dei due punti, in quel caso verrà utilizzata la posizione dell'unico punto presente più un piccolo offset.

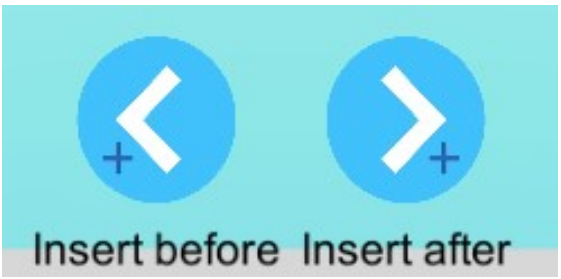

Figura 4.3: Pulsanti per l'inserimento

Tale procedimento si può ripetere fintanto che la distanza tra le due sfere presenti permette di inserirne un'altra senza causare collisioni, in caso negativo l'utente è avvertito tramite Android Toast e l'inserimento non è permesso.

L'ultima funzione è la più importante in quanto permette di far comparire sull'oggetto desiderato le maniglie che controllano le funzionalità di movimento. A seconda del tipo di trasformazione scelta compariranno le frecce o le gimbals da utilizzare tramite trascinamento sul touch screen.

Solo un oggetto per volta può essere attivo e pertanto è possibile anche spostarne uno solo.

Tra le altre operazioni attuabili sugli oggetti vi sono la cancellazione multipla, eseguibile anche senza una sfera selezionata, e la possibilità di abilitare la visione

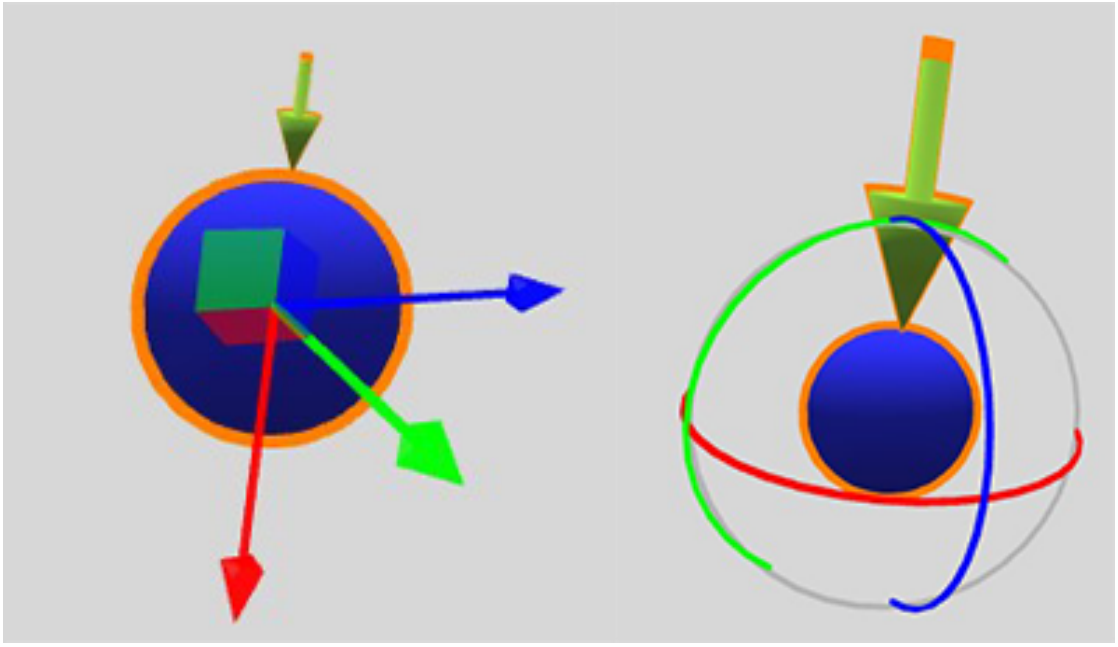

Figura 4.4: Interfaccia per le trasformazioni dei punti AR

degli end effector tramite metafore semitrasparenti in concomitanza della posizione desiderata Fig.4.5.

Quando queste rappresentazioni sono attive, avendo esse un collider che impedirebbe all'utente di selezionare la sfera che desidera, è stato fatto in modo che la selezione del modello dell'end effector vada invece a selezionare la sfera su cui è situato.

Il modello utilizzato per descrivere l'end effector di e.DO è quello della pinza già analizzata in precedenza in Fig.1.18.

Gestione delle collisioni

All'interno della scena di Unity è stato predisposto un particolare asset chiamato ObstacleHandler a cui è possibile attribuire altri GameObject legati da un vincolo parent-child. A runtime si provvederà al calcolo delle intersezioni tra gli elementi figli dell'ObstacleHandler e la linea che attraversa le sfere posizionate dall'utente in modo da individuare e prevenire le eventuali collisioni.

Al conteggio delle collisioni non intervengono solo gli elementi già citati ma anche le versioni virtuali dell'end effector in modo da avere una precisione maggiore nell'individuare le posizioni scorrette.

Gli oggetti devono essere preparati nella scena prima del build dell'applicazione e, per ora, vengono utilizzati per mappare nell'ambiente reale degli ostacoli di cui

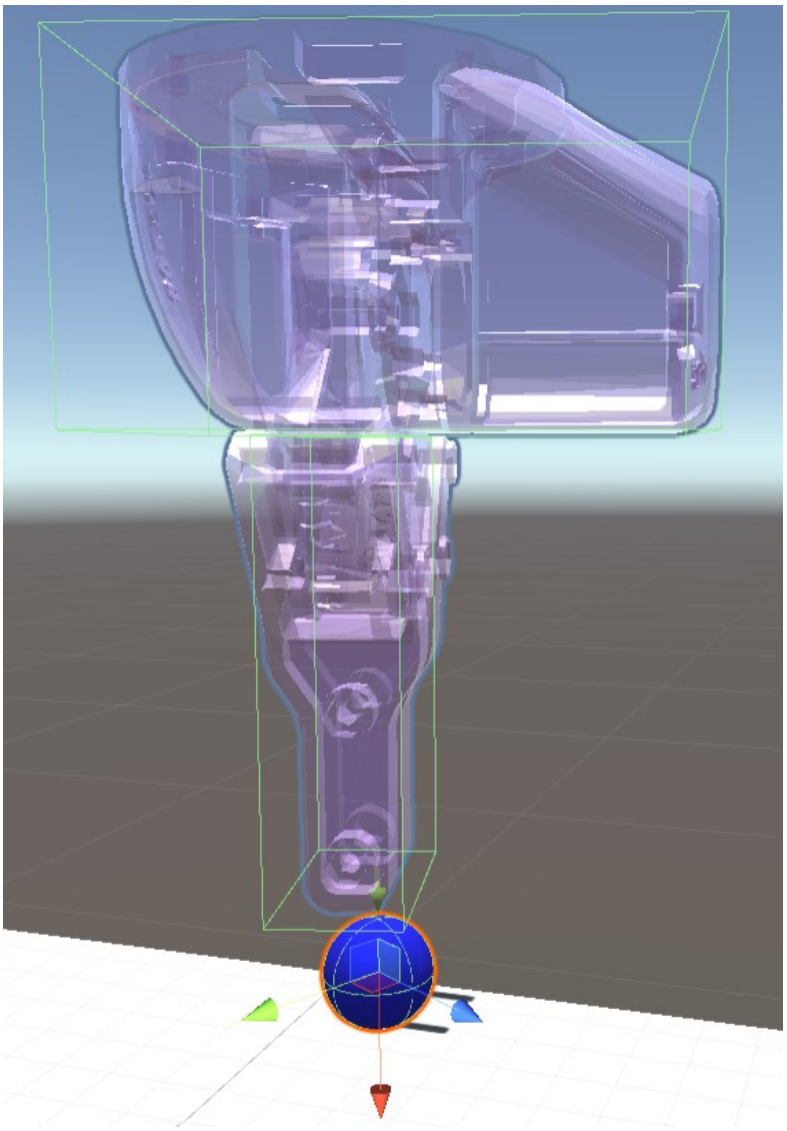

Figura 4.5: Rappresentazione grafica della posizione dell'end effector

è già nota la forma e la posizione, inoltre è obbligatorio che essi abbiano assegnato un componente Collider in modo da svolgere la loro funzione di ostacoli.

Previsualizzazione del movimento del manipolatore

Una volta che l'utente è soddisfatto del percorso creato ha la possibilità di inviare l'elenco di posizioni al server che risponderà con la traiettoria completa elaborata dal motore IK.

Alla ricezione di questa traiettoria, l'applicazione predispone il movimento della versione virtuale del manipolatore per dare all'utente un'ulteriore prova visiva del reale percorso che sarà poi occupato dai giunti meccanici.

Durante il suo movimento il braccio procederà al posizionamento di ulteriori

GameObject nella scena virtuale sotto forma di sfere, analoghe a quelle che si possono creare con il doppio tap, ma di colore giallo e dimensione leggermente ridotta.

I punti così tracciati corrispondono alle posizioni intermedie trovate dal server anche se in numero inferiore in quanto poiché, in caso contrario, risulterebbero in un eccessivo numero di elementi visibili a schermo pregiudicando l'usabilità del sistema e concorrono, insieme a quelli blu al calcolo delle collisioni con gli altri oggetti virtuali ed al disegno della linea con Catmull-Rom.

È infatti possibile che un percorso apparentemente corretto si trasformi in uno inutilizzabile a causa delle posizioni intermedie fornite dalla macchina ROS che, non avendo cognizione dell'ambiente circostante potrebbe dare delle soluzioni che intersecano oggetti presenti sul piano di lavoro.

In tal caso la linea che rappresenta il path diviene di colore rosso e impedisce all'utente di attuare il cammino sul manipolatore reale. Per uscire da questa situazione occorre riposizionare le sfere che causano il fallimento e rinviare i punti a Moveit.

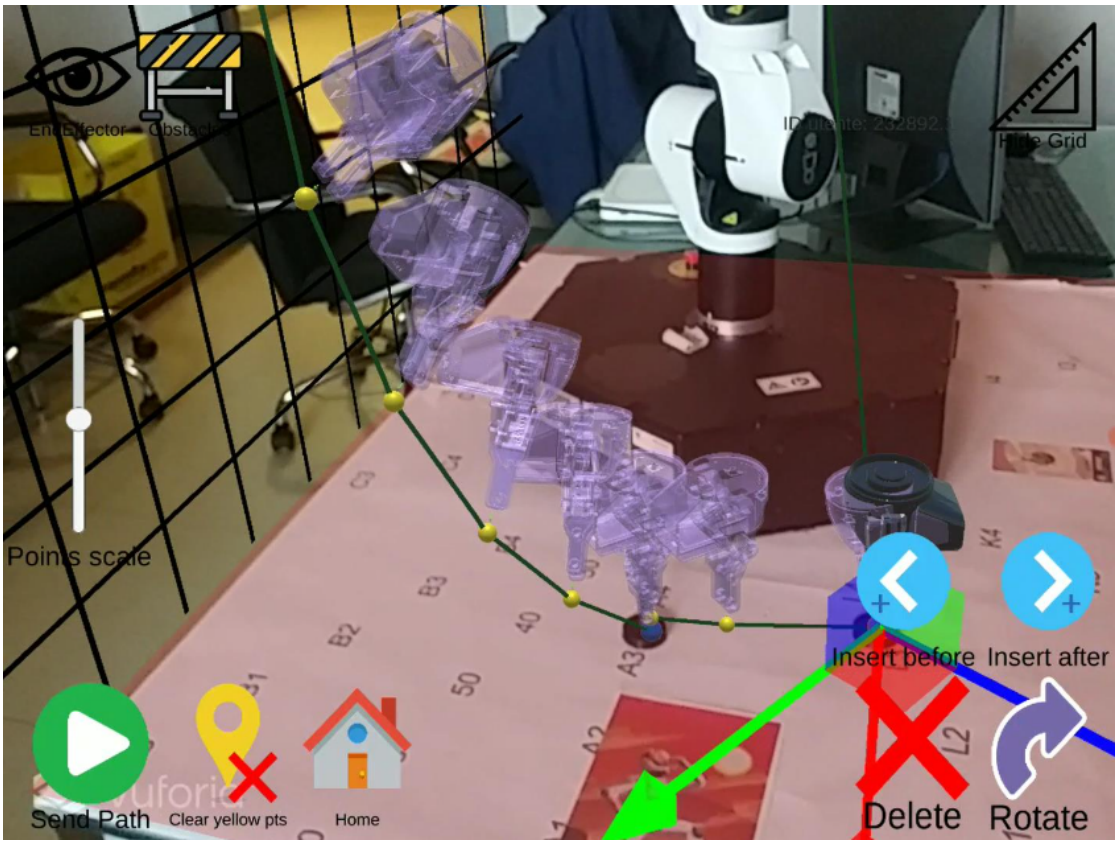

Figura 4.6: Previsualizzazione tramite robot virtuale

Per assistere l'utente in questa operazione, l'applicazione permette di applicare

le stesse trasformazioni che può applicare ai punti da lui piazzati anche alle sfere gialle posizionate in modo automatico, inoltre queste ultime vengono promosse a blu nel momento in cui vengono modificate in modo che nel successivo ricalcolo del percorso, che è preceduto dalla cancellazione dei punti di controllo gialli, restino attive e nella stessa posizione.

Capitolo 5

Risultati e usabilità

5.1 Test

Per verificare le performance del sistema sono stati organizzati dei test presso il Politecnico di Torino e vi hanno preso parte dodici persone tra tesisti e dottorandi di età compresa tra i venti e i trentadue anni.

I test sono stati pensati in modo che ciascuno di essi comprendesse cinque compiti in cui fossero necessarie diverse delle funzionalità fornite dall'applicazione. In questo modo, a seconda di quante e quali funzioni fosse necessario svolgere, i diversi task sono stati ordinati per livello di difficoltà crescente.

Gli utenti hanno svolto questi test guidati da alcuni artefatti stampati in 3D e posizionati sul piano di lavoro del robot in corrispondenza di alcune posizioni prefissate, tali artefatti sono stati creati in modo da fornire un'indicazione sia per quanto riguarda la posizione e l'orientamento finali dell'end-effector.

Il fine ultimo dell'esperienza è stato quindi quello di posizionare i punti con l'interfaccia in Realtà Aumentata nel modo più fedele possibile rispetto a quanto proposto nell'ambiente reale.

I compiti sono stati quindi divisi come segue a seconda delle operazioni che li contraddistinguono:

1. Traslazione 2D
2. Traslazione 3D
3. Rotazione
4. Rotazione e Traslazione 2D
5. Rotazione e Traslazione 3D

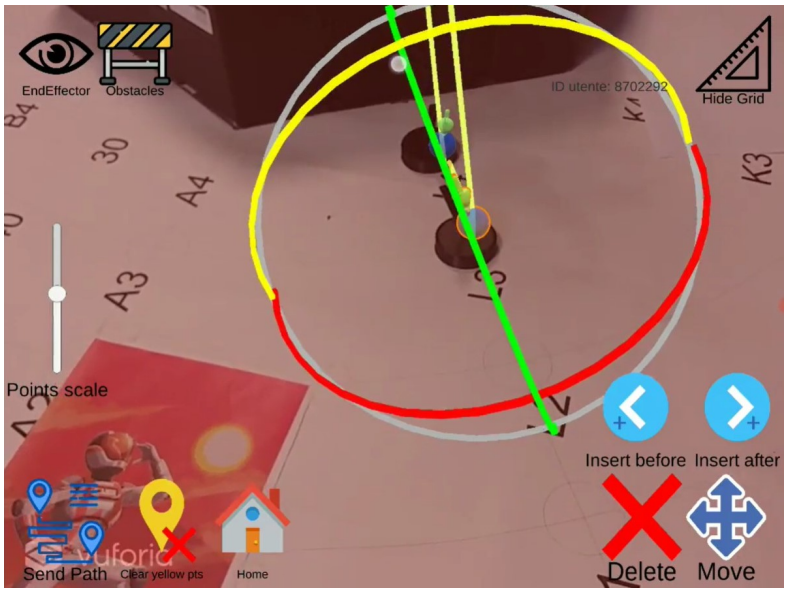

Figura 5.1: Esempio degli artefatti che indicano posizione e rotazione da applicare per il test

Nel primo caso è necessario solamente traslare i punti sul piano identificato dal tavolo su cui è posizionato il manipolatore, nel secondo viene introdotta la richiesta di modificare la terza coordinata posizionando i punti su livelli di altezza differenti pur senza variare l'orientamento che resta quello di default scelto dal sistema e cioè perpendicolare al piano.

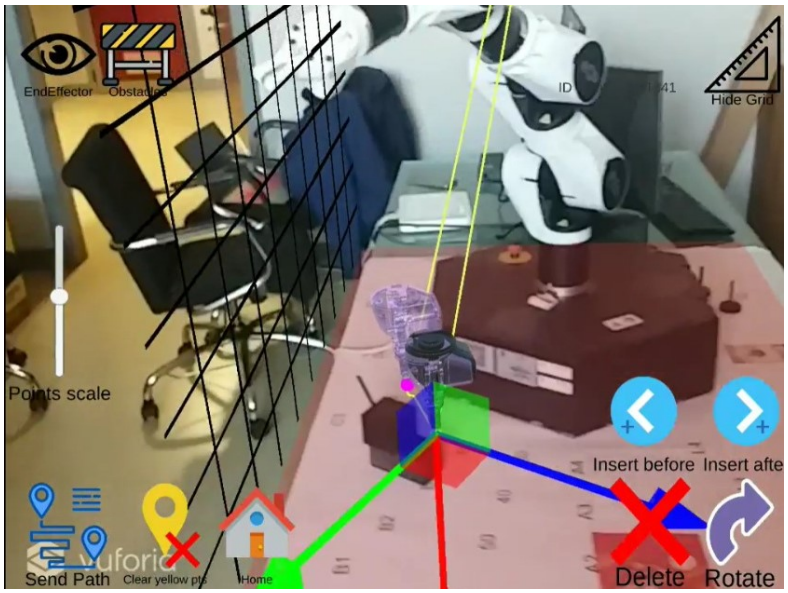

Figura 5.2: Esempio di test che richiede di posizionare un punto su tre dimensioni

Il terzo step contempla la sola rotazione in modo da far prendere familiarità all'utente con la nuova trasformazione; in questa occasione i punti vengono posizionati preventivamente alle coordinate corrette prima di iniziare il test.

A seguire vi sono i due task più complessi che prevedono di dover svolgere in contemporanea due delle operazioni già provate in precedenza fino ad ottenere, nel caso dell'ultimo compito, la scelta di posizione e orientamento con 6 gradi di libertà.

Per ogni utente sono stati raccolte diverse misure tra cui: la distanza tra il punto posizionato dall'utente e quello di riferimento inserito nell'ambiente virtuale, l'angolo tra i versori X dei rispettivi GameObject ad indicare la differenza di orientamento, il numero di tocchi sullo schermo ed il tempo impiegato per portare a termine le operazioni e le coordinate dell'end effector riportare sul terminale connesso al manipolatore tramite ssh.

5.2 Questionari

Prima di procedere con l'utilizzo dell'applicazione è stato chiesto ad ogni partecipante di compilare la prima parte del questionario, composto da dieci domande da valutare con un numero da 1 a 5 a seconda di quanto ci si ritenga d'accordo, il cui scopo è stato raccogliere informazioni sulle loro effettive conoscenze in ambito di robotica, Realtà Aumentata e dispositivi mobili.

Al termine della fase di testing si è infine completato il form rispondendo ai questionari SUS, Nasa TLX e attribuendo una valutazione unica con un numero da 1 a 7 alla difficoltà delle operazioni svolte.

SUS Questionnaire

Il SUS o System Usability Scale [58] è uno strumento per misurare l'usabilità di un sistema. È un questionario costituito da dieci frasi a cui attribuire un numero da 1 a 5. Creato originariamente nel 1986 è oggi utilizzato per valutare una grande varietà di prodotti tra cui hardware, software, applicazioni e siti web. Tra i suoi pregi vi è quello di essere affidabile anche su un campione ridotto di partecipanti.

Questo metodo di valutazione restituisce un valore in una scala che va da zero a cento che si può suddividere in fasce come rappresentato in Fig.5.2.

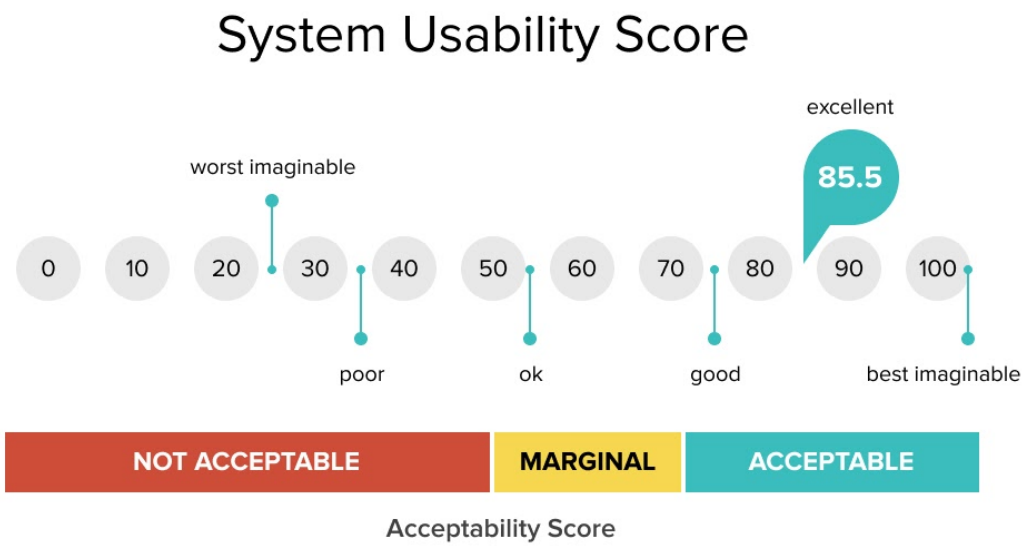

Figura 5.3: Valutazione SUS

Nasa TLX

Il Nasa TLX (Task Load Index) [59] è un metodo per misurare il carico di lavoro di una certa interfaccia di interazione tra umani e macchine. Si basa su una media pesata dei voti attribuiti a sei differenti categorie:

1. Mental demand
2. Physical demand
3. Temporal demand
4. Performance
5. Effort
6. Frustration

A ognuno di questi fattori deve essere assegnato un giudizio in una scala da zero a venti che rappresenta idealmente una votazione in centesimi con step di 5 punti.

A seguire vi è una seconda parte della valutazione in cui sono proposti al candidato quindici confronti a coppie tra gli elementi che ha appena votato. Questo processo è necessario per calcolare dei pesi che determinano l'importanza che secondo l'utente quel determinato fattore ha ricoperto nell'utilizzo dell'applicazione.

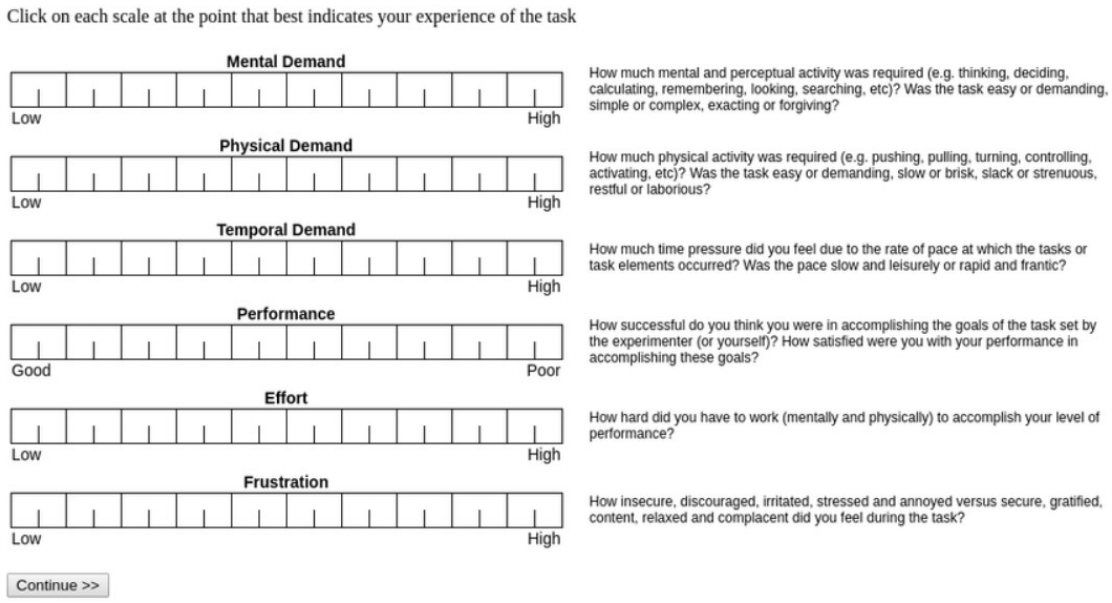

Figura 5.4: Nasa TLX

Il risultato di questo questionario è un punteggio in centesimi in cui, a differenza del SUS, si punta ad ottenere un numero quanto più basso possibile.

Le tabelle 5.1, 5.2 e 5.5 contengono tutti i dati raccolti per distanze e tempi.

Di seguito invece, nelle tabelle 5.3 e 5.4 sono riportati i valori di media e deviazione standard ottenuti dai dati raccolti in fase di testing. Ogni colonna corrisponde a uno dei singoli task svolti e i numeri su cui sono stati ottenuti questi risultati derivano dalla distanza tra le scelte effettuate dall'utente e le coordinate ideali scelte preventivamente in fase di sviluppo nell'ambiente Unity.

	T 2D		T 3D		T + R 3D		T + R 2D	
	A	B	A	B	A	B	A	B
user_0	0.010961344	0.011301491	0.00164825	0.001037609	0.017006701	0.011004801	0.021365641	0.015557771
user_1	0.016548263	0.014529174	0.025741177	0.017356731	0.331133187	0.319967061	0.011216108	0.052397422
user_2	0.0379919	0.02930332	0.019243464	0.034959335	0.028974116	0.082117483	0.010297245	0.005035816
user_3	0.022175238	0.044795256	0.026443815	0.017015666	0.02700053	0.060323216	0.020843651	0.024929997
user_4	0.054870974	0.066031203	0.007	0.121344008	0.013126997	0.007358325	0.013127018	0.016575554
user_5	0.013728242	0.005104734	0.03823882	0.079178408	0.023510799	0.02943838	0.01217085	0.036970183
user_6	0.020214986	0.007637579	0.011329406	0.020243576	0.023074264	0.024797378	0.010758351	0.012903938
user_7	0.051971342	0.022537518	0.050375469	0.033483423	0.023074264	0.024797378	0.010758351	0.012903938
user_8	0.041971342	0.020375183	0.011329406	0.020243576	0.025594855	0.012002916	0.011216108	0.052397422
user_9	0.031842597	0.031842597	0.029180224	0.022001624	0.023074234	0.010694969	0.010758322	0.012358322
user_10	0.031842597	0.015279325	0.028480224	0.021901624	0.042141061	0.008182679	0.024470251	0.011537703
user_11	0.00687626	0.010139422	0.027441997	0.134107113	0.186683193	0.10749729	0.024584549	0.010142386

Tabella 5.1: Tabella delle distanze ottenute da ogni utente per ogni task. Le lettere A e B indicano i due punti da raggiungere

Per quanto riguarda l'errore di posizione si può notare come siano sostanzialmente molto simili tra i primi due task e l'ultimo mentre il terzo, ossia la traslazione in tre dimensioni con rotazione, abbia comportato la maggiore incertezza. Questo

	R 2D		T + R 3D		T + R 2D	
	A	B	A	B	A	B
user_0	0.1	3.669598103	0.71846056	11.96269894	4.242794037	3.400069952
user_1	0.334550232	8.291500092	5.219497681	4.953414917	8.981263161	4.871861935
user_2	16.38941002	12.01486588	9.385838509	24.65594482	3.294317484	8.744645119
user_3	2.019648075	1.609246373	3.825410128	10.01040649	6.341926575	14.94974041
user_4	19.99998093	20.00000191	1.664363623	3.862191439	1.664363623	3.104695082
user_5	4.306070805	9.680583	1.562853336	2.151981544	3.231739283	2.824869156
user_6	13.95022297	21.42988014	1.43176496	28.79395103	5.10165596	18.35648727
user_7	9.15444088	5.036411762	1.43176496	28.79395103	5.10165596	18.35648727
user_8	9.15444088	5.036411762	0.660567999	3.123867512	4.871861935	8.981263161
user_9	7.545629217	1.897444424	1.43176496	13.90943813	5.10165596	15.70109081
user_10	9.774710655	8.107746124	9.074976921	4.389678001	15.13337898	11.35364056
user_11	12.92527866	9.711182594	8.781609535	9.711202621	2.73313942	1.162744045

Tabella 5.2: Tabella delle distanze in gradi ottenute da ogni utente per ogni task che ha richiesto la rotazione

	T 2D	T 3D	T + R 3D	T + R 2D
AVG	0.028416257	0.023037688	0.063699517	0.015130537
DEV	0.015911833	0.013775368	0.096492732	0.005817318

Tabella 5.3: Tabella delle distanze tra punti posizionati e riferimenti

	R 2D	T + R 3D	T + R 2D
AVG	8.804531943	3.765739431	5.483312698
DEV	6.345208404	3.458694433	3.57332932

Tabella 5.4: Distanze in gradi tra i versori dei punti posizionati e dei riferimenti

è chiaramente dettato dal fatto che la difficoltà di quest'ultima operazione è decisamente superiore a alle altre e implica un numero maggiore di errori da parte degli utenti.

La distanza tra i versori è invece più alta nel primo task che dovrebbe corrispondere a quello più semplice. Questo risultato potrebbe essere causato dal fatto che è stato il primo esempio di rotazione ad essere sottoposto ai tester che di conseguenza non avevano ancora preso confidenza con l'interfaccia formata dalle gimbal. Un altro motivo che potrebbe spiegare i dati ottenuti è la deviazione standard, che è decisamente superiore in quel task rispetto agli altri e implica quindi una grande dislivello tra i vari test.

In tabella 5.6 sono invece contenute le medie dei tempi impiegati dai tester per svolgere ciascun compito e le relative deviazioni standard. Come previsto i tempi sono distribuiti in modo crescente all'aumentare della difficoltà del task e pertanto possiamo notare come il più semplice, caratterizzato dalla sola traslazione sul piano, abbia richiesto in media solamente 58 secondi, mentre il più difficile circa due tre minuti. In questo caso compaiono dei valori notevoli come deviazione standard

e sono probabilmente dovuti alla differenza di abilità dei tester nell'utilizzo di interfacce di questo tipo, gli utenti più familiari con i programmi di grafica 3D come blender hanno in media impiegato molto meno tempo a completare le operazioni rispetto agli altri.

	R 2D		T + R 3D		T + R 2D	
	A	B	A	B	A	B
user_0	0.1	3.669598103	0.71846056	11.96269894	4.242794037	3.400069952
user_1	0.334550232	8.291500092	5.219497681	4.953414917	8.981263161	4.871861935
user_2	16.38941002	12.01486588	9.385838509	24.65594482	3.294317484	8.744645119
user_3	2.019648075	1.609246373	3.825410128	10.01040649	6.341926575	14.94974041
user_4	19.99998093	20.00000191	1.664363623	3.862191439	1.664363623	3.104695082
user_5	4.306070805	9.680583	1.562853336	2.151981544	3.231739283	2.824869156
user_6	13.95022297	21.42988014	1.43176496	28.79395103	5.10165596	18.35648727
user_7	9.15444088	5.036411762	1.43176496	28.79395103	5.10165596	18.35648727
user_8	9.15444088	5.036411762	0.660567999	3.123867512	4.871861935	8.981263161
user_9	7.545629217	1.897444424	1.43176496	13.90943813	5.10165596	15.70109081
user_10	9.774710655	8.107746124	9.074976921	4.389678001	15.13337898	11.35364056
user_11	12.92527866	9.711182594	8.781609535	9.711202621	2.73313942	1.162744045

Tabella 5.5: Tempo impiegato dagli utenti in ogni task

	T 2D	T 3D	R 2D	T + R 2D	T + R 3D
AVG	58.61666667	101.1666667	124.1333333	152.1158333	199.1525
DEV	38.1739043	42.21863727	67.27843547	98.31542517	65.19724884

Tabella 5.6: Tempo in secondi per ogni task

La tabella 5.7 riporta i valori ottenuti dai questionari compilati prima e dopo i test. Nelle prime due colonne, con il colore azzurro sono indicati i valori ottenuti dal questionario SUS e dalla votazione singola data alla difficoltà del test, mentre l'ultima colonna riportata in verde indica il questionario Nasa TLX. La differenza di colore sta ad indicare una differente lettura dei risultati, infatti sarebbe preferibile per le colonne azzurre ottenere un valore quanto più alto possibile, mentre per quella verde vale l'esatto contrario.

Ciò che abbiamo ottenuto per il questionario SUS ricade esattamente nella definizione di "Good" secondo la scala riportata in figura 5.2 e fornita da [54]

Il valore della valutazione singola, che ricordiamo indicare la difficoltà complessiva nell'utilizzo del sistema secondo una scala da uno a sette in cui i valori più grandi indicano una semplicità percepita maggiore, è 4 ed è quindi leggermente positivo.

Il questionario Nasa ha invece riportato un valore di 49 che è considerato medio essendo ottenuto in una scala in centesimi.

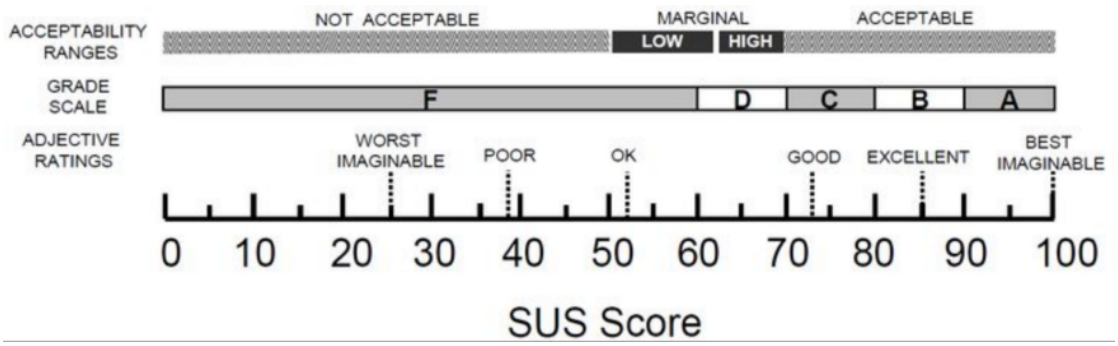

Figura 5.5: Risultati SUS da [54]

	SUS	SINGLE	NASA
SUS	73.95833333	4	49.27333333
DEV	13.46115342	1.044465936	12.67448499

Tabella 5.7: Risultati dei questionari SUS, valutazione singola e Nasa TLX

Oltre alla valutazione singola per la difficoltà è stato chiesto di fornire agli utenti il proprio responso per ogni singolo task effettuato, questa volta in una scala da uno a cinque invertita rispetto al caso precedente. I risultati sono riportati in tabella 5.8 ed indicano una separazione abbastanza netta tra i compiti riguardanti la sola traslazione, ritenuta più semplice, e quelli misti, ritenuti mediamente più complessi.

	T 2D	T 3D	R 2D	T + R 3D	T + R 2D
AVG	1.75	2.583333333	3.166666667	3	3.25
DEV	1.138180366	1.240112409	0.937436867	1.044465936	0.866025404

Tabella 5.8: Valutazione della difficoltà di ogni task

L'ultima tabella indica invece il profilo medio degli utenti ottenuto dalla valutazione di diversi quesiti riportati nella copia del questionario fornita all'ultima pagina della tesi. Le risposte sono state valori compresi tra uno e cinque e le domande sono state raggruppate per categorie come segue:

1. Quesiti 1, 2, 3: livello di esperienza nell'ambito della robotica
2. Quesito 4: familiarità con le trasformazioni tridimensionali (traslazione, rotazione, scalamento)
3. Quesiti 5 e 6: esperienza nell'utilizzo di software di modellazione 3D e Game Engine

4. Quesito 7: conoscenza della Realtà Aumentata

5. Quesiti 8, 9, 10: familiarità con smartphone, tablet e applicazioni AR sviluppate per gli stessi

	AVG	DEV
Q1	2.083333333	0.792961461
Q2	2.083333333	1.378954369
Q3	2.5	1.507556723
Q4	4.166666667	1.114640858
Q5	3.833333333	1.403458931
Q6	3.5	1.507556723
Q7	4.25	0.753778361
Q8	4.5	1.167748416
Q9	3.333333333	1.61432977
Q10	3.083333333	1.443375673

Tabella 5.9: Profilo utente medio

Capitolo 6

Conclusione e sviluppi futuri

Lo scopo del progetto proposto in questa tesi è stato lo sviluppo di un sistema per la programmazione di un manipolatore industriale attraverso un'interfaccia AR che non richiedesse particolari competenze in campo robotico né informatico.

La progettazione di questo sistema ha reso necessario l'approfondimento di vari argomenti tra cui le tecnologie di Realtà Aumentata, la comunicazione client server basata su ROS e la pianificazione del percorso in cinematica inversa tramite MoveIt.

La semplicità d'uso dell'interfaccia è stato uno degli elementi principali dell'applicazione ed è stato necessario trovare un modo per svincolarsi dai problemi legati alla difficoltà di applicare alcune delle trasformazioni possibili nello spazio tridimensionale attraverso gesture dello schermo touchscreen.

Un altro aspetto determinante di questo progetto è la comunicazione tra i tre dispositivi che ne fanno parte: le limitazioni della libreria ROS# che richiede l'utilizzo di un serializzatore diverso in lettura e in scrittura, unitamente all'impossibilità da parte della versione custom di Rosbridge installata sul manipolatore di accettare multipli client ha reso necessario spostare la logica che gestisce la creazione dei punti intermedi dei percorsi su una terza macchina modificando quindi l'architettura hardware.

Proprio questa problematica ha però dato la possibilità di separare il manipolatore dal server rendendo il sistema indipendente dal manipolatore in uso. È infatti possibile ripetere la procedura di setup di Moveit fornendo la descrizione di altri robot tramite file .urdf o .xacro nel package ROS ed effettuarne la simulazione. L'aver utilizzato Moveit come simulatore ha altresì permesso di utilizzare diversi algoritmi per la cinematica inversa permettendo di adattarsi meglio alle prestazioni della macchina che ospita il server o del manipolatore in uso.

Il sistema è stato valutato attraverso test in cui i partecipanti hanno dovuto svolgere cinque diversi scenari in cui sono state inserite in modo incrementale le varie funzionalità proposte dall'applicazione AR. Ognuno dei compiti da portare a

termine è stato composto dal posizionamento e dalla rotazione di 2 punti in modo da formare, seppur in modo approssimato, un path per il manipolatore.

I partecipanti sono stati selezionati tra i laureandi e i dottorandi del Politecnico di Torino e hanno compilato, al termine dell'esperienza, un questionario atto a valutare gli aspetti di usabilità e performance del sistema ed indicando inoltre il proprio livello di esperienza nell'ambito della robotica e della Realtà Aumentata.

Alcuni degli aspetti centrali del client possono sicuramente essere migliorati. Partendo dall'SDK per la Realtà Aumentata sarebbe infatti possibile sfruttarne diverse generando più versioni dell'app con il fine di valutare le prestazioni di ognuna e determinare quale permetta di raggiungere una migliore stabilità durante l'utilizzo e una minor incertezza nel posizionamento dei punti.

Un altro aspetto che sarebbe possibile sostituire sarebbe inoltre la modalità di tracking, passando per esempio dalla tecnica basata su marker a quelle che comportano l'estrazione di feature dall'ambiente reale come il riconoscimento di spigoli e texture, sicuramente più complesse a livello computazionale ma più adatte a uno scenario reale in cui non ci sarebbe più bisogno di effettuare il setup della tavola situata alla base del manipolatore con l'immagine target.

Per quanto riguarda l'interfaccia grafica e l'interazione con i modelli AR, alcune feature, come ad esempio le gimbal, per mezzo delle quali è possibile modificare l'orientamento dei punti, sono risultate invasive nello svolgimento di altre funzioni tra cui la selezione di sfere che alcune volte risultavano occluse, costringendo gli utenti a passare in modalità traslazione per cambiare l'oggetto attivo. Sempre in questo ambito è stata lamentata la mancanza di un comando per passare dal sistema di riferimento del robot a quello globale del mondo virtuale in quanto l'applicare una rotazione prima di una traslazione modifica la posizione degli assi rendendoli diversi da quelli che contraddistinguono il piano di lavoro e la dimensione dell'altezza.

Un ultimo aspetto su cui senza dubbio sarebbe estremamente utile lavorare il sistema di riconoscimento degli ostacoli. Attualmente l'applicazione permette di inserire nell'editor di Unity dei modelli 3D contrassegnati come ostacoli ed è in grado di determinarne le intersezioni con i punti intermedi del percorso pianificato per il robot impedendo quindi di avviarlo e contrassegnandolo con il colore rosso.

Questo è tuttavia inadatto all'utilizzo in un caso reale in cui questi ostacoli non sono noti a priori né nella posizione né nella forma. Sarebbe interessante fornire un metodo per ricreare a runtime delle approssimazioni degli oggetti reali disseminati nello spazio di lavoro per usufruire della funzionalità di rilevamento delle collisioni già introdotta.

Questo nuovo aspetto potrebbe essere inserito in diverse modalità.

Una prima approssimazione potrebbe essere constituita dall'introdurre un sistema simile a quello già proposto e utilizzato per posizionare i punti in cui si posizionano delle primitive geometriche come sfere e parallelepipedi nello scenario virtuale e se ne controllano, oltre alla posizione e l'orientamento, anche i fattori di

scala lungo i tre assi. In questo modo sarebbe quindi possibile, all'avvio dell'applicazione, andare a creare dei collider che rappresentino eventuali ostacoli reali approssimandone la forma tramite la funzione di scalamento.

Una versione più avanzata sarebbe invece da sviluppare tramite algoritmi di computer vision per essere in grado di stimare la dimensione e forma degli oggetti reali.

Bibliografia

[1] P. Milgram and F. Kishino, "A Taxonomy of Mixed Reality Visual Displays", IEICE Transactions on Information and Systems, Vol. 77, 1994, pp. 1321-1329,

[2] Ronald Azuma, "A survey of augmented reality", Presence: Teleoperators and Virtual Environments, 6, 02 1996, DOI 10.1162/pres.1997.6.4.355

[3] D. Amin and S. Govilkar, "Comparative Study of Augmented Reality SDK's", 5, 11-26, 2015, DOI https://doi.org/10.5121/ijcsa.2015.5102 [Paper reference 2]

[4] D.Jooste V.Rautenbach and S.Coetzee , "Results of an Evaluation of Augmented Reality Mobile Development Frameworks for Addresses in Augmented Reality", Spatial Information Research, 24 1-13 2016, DOI https://doi.org/10.1007/s41324-016-0022-1 [Paper reference 2]

[5] Herpich Guarese Tarouco, "A Comparative Analysis of Augmented Reality Frameworks Aimed at the Development of Educational Applications", Creative Education, 8, 1433-1451, 2016, DOI https://doi.org/10.4236/ce.2017.89101

[6] Spivak Sophia, "ARticular: an augmented reality game concept for a museum setting", Creative Education, 02, 1433-1451, 2015, DOI 10.13140/RG.2.2.17344.33281

[7] Jack Loomis, Reginald Golledge, Roberta Klatzky, Jon Speigle, and Jerome Tietz. "Personal guidance system for the visually impaired", pp. 85–91, 01 1994, DOI 10.1145/191028.191051.

[8] H. Durrant Whyte and T. Bailey, "Simultaneous localization and mapping: part I", IEEE Robotics e Automation Magazine. vol. 13, no. 2, pp. 99-110, June 2006, DOI 10.1109/MRA.2006.1638022.

[9] H. F. Durrant-Whyte, "Uncertain geometry in robotics", IEEE Trans. Robot. Automat., vol. 4, no. 1, pp. 23-31, 1988.

[10] R. Smith and P. Cheesman, "On the representation of spatial uncertainty", Int. J. Robot. Res., vol. 5, pp. 56-68, 1987 no. 4,

[11] A. Sanna and F. Manuri, "A Survey on Applications of Augmented Reality", Advances in Computer Science: an International Journal, Vol. 5, no. 1, pp.18-27, January 2016

[12] D. Schmalstieg and T. Hollerer, "Augmented Reality: Principles and Practice", Addison-Wesley Professional,, June 2016

[13] S. C.-Y. Yuen, G. Yaoyuneyong, and F. Johnson, "Augmented reality: an overview and five directions for AR in education", Journal of Educational Technology Development and Exchange, Vol. 4, no. 1, pp. 119-140, 2011

[14] H.-K. Wu, S.W.-Y. Lee, H.-Y. Chang, and J.-C. Liang, "Current status, opportunities and challenges of augmented reality in education", Computers and Education, Vol. 62, pp. 41–49, 2013

[15] Tansel Tepe, Devkan Kaleci and Hakan Tüzün, "Virtual Reality Applications in Education", 12 2017., DOI 10.1007/978-3-319-08234-9_166-1.

[16] G. Schall, S. Junghanns, and D. Schmalstieg, "The Transcoding Pipeline: Automatic Generation of 3D Models from Geospatial Data Sources", Workshop on Trends in Pervasive and Ubiquitous Geotechnology and Geoinformation in conj. with the 5th International Conference on GIScience (GISCIENCE 2008), Park City, Utah (USA), 23-26 September 2008

[17] R. Schoenfelder and D. Schmalstieg, "Augmented Reality for Industrial Building Acceptance", In Proceedings of IEEE Virtual Reality, Reno NV (USA), March 2008, pp. 83-90

[18] Svetlana Bialkova and Enrique Bigne, "haping the future of virtual reality marketing: perspectives and challenges", 06 2017.

[19] T. Sielhorst, M. Feuerstein, and N. Navab, "Advanced medical displays: a literature review of augmented reality", Journal of Display Technologies, Vol. 4, no. 4, pp. 451-467, 2008

[20] N. Navab, S. M. Heining, and J. Traub, "Camera augmented mobile Carm (CAMC): calibration, accuracy study, and clinical applications", IEEE Transactions on Medical Imaging, Vol. 29, no. 7, pp. 1412-1423, 2010, DOI 10.1109/TMI.2009.2021947

[21] S. Henderson and S. Feiner, "Exploring the benefits of augmented reality documentation for maintenance and repair", IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 10, pp. 1355–1368, 2011

[22] Katharina Buckl, Stefan Misslinger, Piero Chiabra, and Glyn Lawson "Augmented Reality for Remote Maintenance", IEEE pages 217–234. 07 2011. ISBN 978-1-84996-171-4. DOI 10.1007/978-1-84996-172-1_13

[23] www.toptal.com/robotics/introduction-to-robot-operating-system

[24] Bayati, Mostafa "Using cuckoo optimization algorithm and imperialist competitive algorithm to solve inverse kinematics problem for numerical control of robotic manipulators. Proceedings of the Institution of Mechanical Engineers, Part I", IEEE Journal of Systems and Control Engineering, Vol. 229, pp.375-387. DOI 10.1177/0959651814568364

[25] https://edo.cloud/the-robot/

[26] Bowman, D., Mcmahan, R., and Ragan, "Questioning naturalism in 3D user interfaces", Communications of the ACM,2012, September, 78-88.

[27] Funk, Markus and Kritzler, Mareike and Michahelles, Florian "HoloLens is more than air Tap: natural and intuitive interaction with holograms", 2017. DOI 10.1145/3131542.3140267

[28] Asier Marzo, Benoît Bossavit, and Martin Hachet, "Combining multi-touch input and device movement for 3D manipulations in mobile augmented reality environments. In Proceedings of the 2nd ACM symposium on Spatial user

interaction (SUI '14)", 2014, Association for Computing Machinery, New York, NY, USA, 13–16. DOI https://doi.org/10.1145/2659766.2659775

[29] Kim, M., Lee, J.Y, "Touch and hand gesture-based interactions for directly manipulating 3D virtual objects in mobile augmented reality", Multimed Tools Appl 75, 16529–16550 (2016). DOI https://doi.org/10.1007/s11042-016-3355-9

[30] Jarkko Polvi, Takafumi Taketomi, Goshiro Yamamoto, Arindam Dey, Christian Sandor, Hirokazu Kato, "SlidAR: A 3D positioning method for SLAM-based handheld augmented reality", Computers and Graphics, Volume 55, 2016, Pages 33-43, ISSN 0097-8493, DOI https://doi.org/10.1016/j.cag.2015.10.013

[31] Liu, J., Au, O.K.-C., Fu, H. and Tai, "Two-Finger Gestures for 6DOF Manipulation of 3D Objects", Computer Graphics Forum, 31: 2047-2055. DOI https://doi.org/10.1111/j.1467-8659.2012.03197.x

[32] Hancock, Mark and Cate, Thomas and Carpendale, Sheelagh, "Sticky tools: Full 6DOF force-based interaction for multi-touch tables", 133-140, 2019 DOI 10.1145/1731903.1731930

[33] Liu, J., Au, O.K.-C., Fu, H. and Tai, "Integrality and Separability of Multitouch Interaction Techniques in 3D Manipulation Tasks", IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 3, pp. 369-380, March 2012, DOI doi: 10.1109/TVCG.2011.129

[34] Reisman, Jason and Davidson, Philip and Han, Jefferson, "A screen-space formulation for 2D and 3D direct manipulation", UIST 2009 - Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology. 69-78. DOI 69-78. 10.1145/1622176.1622190

[35] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann, "3DTouch and HOMER-S: intuitive manipulation techniques for one-handed handheld augmented reality", In Proceedings of the Virtual Reality International Conference: Laval Virtual (VRIC '13). Association for Computing Machinery, New York, NY, USA, Article 12, 1–10. DOI https://doi.org/10.1145/2466816.2466829

[36] Doug A. Bowman and Larry F. Hodges, "An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments", In Proceedings of the 1997 symposium on Interactive 3D graphics (I3D '97). Association for Computing Machinery, New York, NY, USA, 35–ff. DOI https://doi.org/10.1145/253284.253301

[37] R. Bischoff and A. Kazi,"Perspectives on augmented reality based human-robot interaction with industrial robots", 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, 2004, pp. 3226-3231 vol.4,DOI 10.1109/IROS.2004.1389914

[38] S. M. Chacko and V. Kapila,"An Augmented Reality Interface for Human-Robot Interaction in Unconstrained Environments", 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), New Delhi, India, 2019, pp. 1-8,DOI 10.1109/RO-MAN46459.2019.8956466

[39] A. Gaschler, M. Springer, M. Rickert and A. Knoll,"Intuitive robot tasks with augmented reality and virtual obstacles", 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 6026-6031, DOI 10.1109/ICRA.2014.6907747

[40] Makhataeva, Zhanat; Varol, Huseyin A,"Augmented Reality for Robotics: A Review", Robotics 9, no. 2: 21, DOI https://doi.org/10.3390/robotics9020021

[41] K. Ishii, Y. Takeoka, M. Inami and T. Igarashi,"Drag-and-drop interface for registration-free object delivery", 19th International Symposium in Robot and Human Interactive Communication, Viareggio, 2010, pp. 228-233, DOI 10.1109/ROMAN.2010.5598722

[42] C. Hutton,"Augmented Reality Interfaces for Semi-Autonomous Drones", 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Osaka, Japan, 2019, pp. 1361-1362, DOI 10.1109/VR.2019.8797893

[43] S. M. Chacko and V. Kapila, "Augmented Reality as a Medium for Human-Robot Collaborative Tasks", 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), New Delhi, India, 2019, pp. 1-8, DOI 10.1109/RO-MAN46459.2019.8956466

[44] Frank, Jared & Moorhead, Matthew & Kapila, Vikram, "Mobile Mixed-Reality Interfaces That Enhance Human–Robot Interaction in Shared Spaces", 2017 Frontiers in Robotics and AI, 4, DOI 10.3389/frobt.2017.00020

[45] J. A. Frank, M. Moorhead and V. Kapila, "Realizing mixed-reality environments with tablets for intuitive human-robot collaboration for object manipulation tasks", 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), New York, NY, 2016, pp. 302-307, DOI 10.1109/ROMAN.2016.7745146

[46] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. F. Machiel Van der Loos and E. Croft, "Robot Programming Through Augmented Trajectories in Augmented Reality", 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 1838-1844, DOI 10.1109/IROS.2018.8593700

[47] S. Rawat, S. Vats, and P. Kumar, "Evaluating and exploring the myo armband", in System Modeling & Advancement in Research Trends (SMART), International Conference. IEEE, 2016, pp. 115–120.

[48] T. H. Michael, "Scientific computing: an introductory survey", The McGraw-540 Hill Companies Inc.: New York, NY, USA, 2002.

[49] Lahr, Gustavo and Soares, João and Garcia, Henrique and Adriano, and Siqueira, A.A.G. and Caurin, Glauco, "Understanding the Implementation of Impedance Control in Industrial Robots", 10, 2016. DOI 10.1109/LARS-SBR.2016.52

[50] Siciliano, Bruno, "Parallel Force/Position Control of Robot Manipulators", 1996. DOI 10.1007/978-1-4471-0765-1_9

[51] C. P. Quintero, M. Dehghan, O. Ramirez, M. H. Ang, and M. Jagersand, "Flexible virtual fixture interface for path specification in telemanipulation",

in Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017, pp. 5363–5368.

[52] M. Kapinus, V. Beran, Z. Materna and D. Bambušek, "Spatially Situated End-User Robot Programming in Augmented Reality", 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), New Delhi, India, 2019, pp. 1-8, DOI 10.1109/RO-MAN46459.2019.8956336

[53] Edwin Catmull, Raphael Rom, "A class of interpolating splines", Computer Aided Geometric Design, Academic Press, 1974, Pages 317-326, DOI https://doi.org/10.1016/B978-0-12-079050-0.50020-5

[54] https://uxpajournal.org/sus-a-retrospective

[55] https://developer.vuforia.com/

[56] https://unity.com/

[57] https://github.com/siemens/ros-sharp/wiki

[58] Brooke, John. "SUS: A quick and dirty usability scale." Usability Eval. 1995. Ind. 189.

[59] https://humansystems.arc.nasa.gov/groups/tlx/

Glossario

GameObject: Elemento fondamentale di Unity che rappresentano personaggi, oggetti e scenari che agisce da contenitore per i Component

Script: Documento contenente il codice C# (nel caso di Unity) che descrive il comportamento di un oggetto o una scena

Component: script che implementa una funzionalità aggiuntiva per un GameObject

Transform: Component che contiene le informazioni di posizione, orientamento e scala di un GameObject

Mesh renderer: Component che gestisce il processo di rendering di un GameObject a partire dalla sua geometria e posizione

Asset: Un oggetto utilizzato in un progetto Unity. Può anche essere un file creato da un programma esterno e importato nell'editor come modelli 3D, file audio o immagini

Unity Package: Contenitore che racchiude alcuni Asset.

Vector3: Classe che implementa un vettore di 3 elementi corrispondenti ai tre assi cartesiani e generalmente utilizzato per indicare la posizione di un GameObject

End Effector: Dispositivo montato in corrispondenza della fine di un braccio robotico progettato per interagire con l'ambiente circostante

ROS Package: Contiene nodi, messaggi e servizi insieme a un manifesto in XML che ne descrive nome, versione e dipendenze

Stack: Insieme di ROS Package

Dispositivo wearable: Dispositivo indossabile dall'utilizzatore

Dispositivo Handheld: Dispositivo palmare, da utilizzare con le mani

Gesture: Il movimento di una parte del corpo, generalmente mani o testa, che esprime un'idea o un significato

Ray Casting: In Unity è il processo con cui si lancia un raggio invisibile da un punto in una specifica direzione con il fine di rilevare eventuali collisioni lungo il percorso

Gimbals: Un'interfaccia costituita da tre anelli allineanti lungo i tre assi cardinali utilizzata per specificare l'orientamento di un oggetto.

Runtime: Il periodo di tempo che intercorre tra l'inizio di un programma e la sua fine

Pick and Place: Compito di presa e spostamento di un determinato oggetto

Thresholding: Operazione di sogliatura con la quale di determina se una certa quantità è maggiore o inferiore al riferimento

Percezione Aptica: È il processo di riconoscimento degli oggetti attraverso il tatto

Best Practice: Linee guida che indicano il modo più efficiente con cui svolgere un'azione

Open Source Software (OSS): Codice destinato al pubblico accesso, chiunque può leggerlo, modificarlo e ridistribuirlo a piacere

Topic: Argomento

Thread: Flusso di esecuzione

Plugin: Estensione di un programma che ne migliora o aggiunge delle funzionalità. Anche chiamati Addon

Callback: Funzione passata come argomento ad un'altra e invocata all'interno di quest'ultima per completare un qualche tipo di operazione

App: Applicazione

Tag: Etichetta

Field of View: Campo visivo, in particolare ne indica i gradi di ampiezza

Acronimi

AR: Realtà aumentata

MR: Realtà mista

VR: Realtà virtuale

ROS: Robotic Operating System

DOF: Degree of Freedom

DOM: Degree of Motion

TCP (nella parte di robotica): Tool Centre Point

LIDAR: Light Detection and Ranging o Laser Imaging Detection and Ranging

URDF: Unified Robot Description Format

IK: Inverse Kinematics o Cinematica inversa

LTS: Long Term Support

HMD: Head Mounted Display

FOV: Field of View

TOF: Time of Flight

GPS: Global Positioning System

TOA: Time of Arrival

RLG: Ring Laser Gyroscope

FOG: Fiber Optics Gyroscopes

TLX: Test Load Index

SUS: System Usability Scale

QUESTIONARIO

ID	
Age	
Sex	

1 (not at all) - 2 (slightly) - 3 (moderately) - 4 (fairly) - 5 (extremely)	
How much do you consider yourself expert of robotic?	
Do you know how to program a robotic arm?	
Do you know how industrial robotic arms work?	

1 (not at all) - 2 (slightly) - 3 (moderately) - 4 (fairly) - 5 (extremely)	
How much do you consider yourself expert of computer graphics (in terms of virtual object manipulation, transformation, etc.)?	

1 (never) - 2 (once per month) - 3 (once per week) - 4 (several times per week) - 5 (every day)	
How often do you use 3D computer graphics software (such as Blender, Maya) ?	
How often do you use game engine (such as Unity3D, Unreal) ?	

1 (not at all) - 2 (slightly) - 3 (moderately) - 4 (fairly) - 5 (extremely)	
Do you know Augmented Reality (AR) ?	

1 (never) - 2 (once per month) - 3 (once per week) - 4 (several times per week) - 5 (every day)	
How often do you use a mobile devices (smartphones) ?	
How often do you use a tablet ?	
How often do you use AR applications (gaming, amazon, google maps etc..)?	

1. SUS

Please evaluate the usability of the system

1 (not at all) - 2 (slightly) - 3 (moderately) - 4 (fairly) - 5 (extremely)		
I think that I would like to use this system frequently.		Penso che mi piacerebbe utilizzare frequentemente questa
I found the system unnecessarily complex.		Ho trovato l'app inutilmente complesso
I thought the system was easy to use.		Ho trovato l'app molto semplice da usare
I think that I would need the support of a technical person to be able to use this system.		Penso che avrei bisogno del supporto di una persona esper
I found the various functions in this system were well integrated.		Ho trovato le varie funzionalità del'app bene integrate
I thought there was too much inconsistency in this system.		Ho trovato incoerenze tra le varie funzionalità dell'app
I would imagine that most people would learn to use this system very quickly.		Penso che la maggior parte delle persone possano imparare
I found the system very cumbersome to use.		Ho trovato l'app molto difficile da utilizzare
I felt very confident using the system.		Mi sono sentito a mio agio nell'utilizzare l'app
I needed to learn a lot of things before I could get going with this system.		Ho avuto bisogno di imparare "molte cose" prima di riuscire

2. TASK PERFORMANCE

Please rank the following tasks according to their difficulty

1 (easy) - 2 (slightly difficult) - 3 (moderately difficult) - 4 (fairly difficult) - 5 (extremely difficult)	
Translation 2D	
Translation 3D	
Rotation	
Translation + Rotation (2D)	
Translation + Rotation (3D)	

3. Single Use Questionnaire

From 1 (very difficult) to 7 (Very Easy)	
Overall, how difficult or easy did you find the task?	

4. NASA TLX

	Rating	**Tally**	**Weight**
Mental Demand			
Physical Demand			
Temporal Demand			
Performance			
Effort			
Frustration			

Overall

5. COMMENTS

6. Robot EE		COORDINATE	TEMPO
T 2D			
T 3D			
R 2D			
T + R 3D			
T + R 2D			