Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

A.A. 2020/2021

Sessione di Laurea Dicembre 2021

Definizione di traiettorie per manipolatori robotici mediante interfacce in realtà mista

Relatori:
Andrea Sanna
Federico Manuri

Candidato:
Paolo Forteleoni

A nonno Franco e nonna Giacomina.

Anche se non potete essere qui,

tutto questo è anche merito vostro.

Indice

Elenco delle figure

Elenco delle tabelle

Abstract

A partire dalla continua evoluzione verso l'industria 4.0 e grazie agli innumerevoli progressi informatici degli ultimi anni si è assistito alla nascita e allo sviluppo di nuove e innovative tecnologie: tra queste hanno avuto una notevole diffusione la Realtà Virtuale e la Realtà Aumentata.

Esse forniscono un eccellente supporto in campo professionale, ivi per cui stanno riscontrando un grande successo: migliorano la sicurezza sul luogo di lavoro in quanto consentono, tra le varie mansioni, di rilevare in tempo reale eventuali errori relativi a un sistema e aumentare la precisione di determinati *task*.

L'obiettivo principale dell'elaborato è fornire un contributo concreto, se pur parziale, alla Human Robot Interaction (HRI) che si è concretizzato nella creazione di un programma dall'interfaccia semplice in modo che sia di facile comprensione e impiegabile dalle aziende, al fine di aumentare la sicurezza sul luogo di lavoro e accelerare il processo di training per la formazione del personale addetto all'uso di manipolatori robotici.

Invece, gli obiettivi secondari sono stati: fornire un'analisi accurata dei dati raccolti in merito alle principali tematiche affrontate in questo progetto, mettendone in evidenza le peculiarità, i pregi e i difetti; proporre delle nuove chiavi di lettura del tema illustrando, nello specifico, gli innumerevoli e svariati ambiti applicativi.

Al fine di raggiungere questi obiettivi è stato impiegato, lato software, il game engine Unity3D e il motore di cinematica inversa MoveIt!, mentre per la parte hardware, il manipolatore robotico e.Do e l'HMD HoloLens 2.

Alla luce di tale studio sarebbe interessante, in ricerche analoghe, affrontare alcuni temi non trattati in questa sede. Per esempio, si potrebbe realizzare uno studio simile approfondendo però differenze legate alle caratteristiche del software, al rapporto tra tecnologia e contesto;

oppure sarebbe curioso effettuare una ricerca futura che possa indagare quantitativamente l'impiego della Realtà Mista concentrandosi in dettaglio su un ambito di applicazione tra quelli discussi nella parte introduttiva dell'elaborato.

L'applicazione così sviluppata è fruibile esclusivamente con il robot e.Do. Quindi, uno studio futuro potrebbe indagare la possibilità di scegliere liberamente un manipolatore diverso da utilizzare tramite un'interfaccia virtuale o la scansione di un codice QR.

Capitolo 1

Introduzione

1.1 Contesto

I progressi digitali degli ultimi anni e la continua migrazione verso l'industria 4.0 hanno favorito lo sviluppo e la nascita di nuove e rivoluzionarie tecnologie. Tra queste, la Realtà Virtuale e la Realtà Aumentata stanno riscontrando notevole successo in quanto sono un ottimo supporto in ambito lavorativo: permettono di aumentare la sicurezza sul luogo di lavoro, rilevare in tempo reale eventuali errori relativi a un processo o al malfunzionamento di una macchina, aumentare la precisione di determinati *task*, etc.

Anche la robotica sta beneficiando della rivoluzione digitale: i robot sono utilizzati per compiere attività programmate finalizzate al miglioramento dell'efficienza e della qualità del lavoro, oltre a sostituire le persone in attività pericolose o ripetitive; ivi per cui sono ampiamente utilizzati in ambito industriale per operazioni di montaggio, assemblaggio o verniciatura. Di solito, la manipolazione del robot avviene da una stazione remota attraverso l'utilizzo di joystick e monitorando l'azione del manipolatore tramite un computer. Però, dato l'elevato costo delle macchine e la necessità di personale altamente qualificato per il controllo, al giorno d'oggi non tutte le imprese possono beneficiare dell'ausilio dei robot.

Negli ultimi anni, la *Human-Robot interaction* (HRI) è diventata un argomento molto importante e dibattuto nel settore dell'automazione: si stanno studiando e sperimentando nuove metodologie per rendere la manipolazione robotica più semplice e intuitiva e permettere al personale, con basse o nulle conoscenze tecnologiche, di poter utilizzare un manipolatore robotico. In questo ambito, in particolare, sta riscontrando un grande successo l'utilizzo della Realtà Mista, poiché permette alla forza lavoro di interagire con i robot in un ambiente simulato e sicuro.

Alla luce di questo contesto, trova origine tale tesi e il progetto sottostante. In particolare, il lavoro è stato svolto in parte al Politecnico di Torino, negli appositi laboratori, e in parte da remoto, a causa delle complicazioni dovute alla pandemia di COVID-19.

L'elaborato è diviso in due parti: la prima, composta dal capitolo 2 e dal capitolo 3, riguarda la parte teorica della tesi; mentre la seconda si concentra maggiormente sulla parte applicativa.

In particolare, nel secondo capitolo si affrontano le tematiche relative allo stato dell'arte e gli ambiti applicativi della Realtà Mista e della robotica. Nel terzo capitolo si discute delle tecnologie utilizzate per il progetto, mentre il quarto capitolo discute l'esposizione del progetto stesso. Infine, il quinto ed ultimo capitolo esplora i test utilizzati e i risultati a cui hanno condotto.

1.2 Motivazioni della tesi

Le motivazioni dietro la scelta di questo progetto di tesi sono molteplici: prima di tutto, una personale curiosità ad approfondire le tematiche relative alla Realtà Aumentata e alla Realtà Virtuale studiate durante il percorso magistrale; in secondo luogo, il desiderio di poter utilizzare uno strumento ancora poco presente in ambito industriale ma che, probabilmente, avrà un vasto utilizzo in futuro: l'*head-mounted display* (HMD) HoloLens 2.

Inoltre, si è dimostrato interessante utilizzare le sopracitate tecnologie e applicarle ad un progetto riguardante un tema senza dubbio attuale: in questo caso la HRI.

La presente tesi, quindi, ha permesso di affrontare questi temi e di inserire in un progetto reale diverse tematiche studiate durante gli anni di studi universitari. Dopo aver letto e approfondito le ricerche condotte a riguardo, l'analisi della letteratura esistente sul tema ha costituito le fondamenta su cui è stato creato l'elaborato.

Per quanto concerne invece la scelta del progetto, esso nasce da una moltitudine di fattori: innanzitutto, dall'opportunità di utilizzare e approfondire le conoscenze di un potente game engine come Unity3D; in secondo luogo, dalla possibilità di far pratica con le tecnologie legate all'AR e, infine, di imparare a programmare e utilizzare gli HoloLens.

Il progetto è stato suddiviso in tre fasi: setup, implementazione e test; le quali saranno adeguatamente discusse nel Capitolo 4 del presente elaborato.

Infine, è stata utilizzata la Realtà Mista al fine di pianificare le traiettorie di un manipolatore robotico che, nel caso in esame, è l'e.Do dell'azienda Comau di Grugliasco.

1.3 Obiettivi

Obiettivo principale o primario di questo elaborato è stato quello di fornire un contributo concreto, se pur parziale, alla Human Robot Interaction (HRI). Tale contributo si è concretizzato nella creazione di un programma fruibile e facilmente utilizzabile dalle aziende, al fine di incrementare la sicurezza sul luogo di lavoro e accelerare il processo di training per la formazione del personale addetto all'utilizzo di manipolatori robotici. Questo scopo ha riguardato, in particolare, la parte sperimentale della tesi.

Obiettivi secondari sono stati, invece: fornire un'analisi accurata dei dati raccolti in merito alle principali tematiche quali Realtà Mista, Realtà Aumentata, HoloLens 2, ecc., mettendone in evidenza le peculiarità, i pregi e i difetti; proporre delle nuove chiavi di lettura del tema illustrando, in particolare, gli innumerevoli e svariati ambiti applicativi. Questi obiettivi, invece, erano indirizzati alla prima parte del testo, ossia quella bibliografica.

In questo modo, è stato possibile indagare e ampliare la letteratura sul tema dimostrandone la complessità, oltre che l'importanza.

In una prospettiva più pratica, in relazione al progetto, l'obiettivo consisteva nel creare un'applicazione che permettesse di simulare e visualizzare in ambiente virtuale e sicuro il movimento che il robot, dato uno o più input, compierebbe nello spazio 3D. Attraverso tale tecnologia, l'utente sarebbe in grado di selezionare, tramite l'utilizzo degli HoloLens 2, un numero arbitrario di punti che definiscono il percorso desiderato.

Infine, durante la fase di creazione dell'applicazione, è stato importante implementare un codice robusto e di semplice lettura. Inoltre, è stato necessario garantire che fosse facile da modificare per agevolare eventuali aggiornamenti futuri.

Quindi, mediante l'elaborato si è tentato di evidenziare il tema della Realtà Mista applicata alla robotica sia ad un livello prettamente teorico sia ad uno maggiormente pratico.

Capitolo 2

Stato dell'arte

Accanto alle già conosciute Realtà Aumentata e Realtà Virtuale, esiste anche la Realtà Mista (o *mixed reality*). Quest'ultima è una tecnologia che nasce come combinazione tra AR e VR e permette agli utenti di percepire il mondo reale e gli oggetti posti al suo interno, ma anche di osservare oggetti virtuali verosimili e persino reattivi. MR è, quindi, un tentativo di coniugare insieme gli aspetti migliori di AR e VR.

Si propone come un filtro applicabile al mondo reale: a differenza delle altre due, la MR utilizza elementi della VR ma è maggiormente immersiva e dinamica poiché il virtuale è integrato al reale e a oggetti dell'ambiente circostante. Inoltre, permette di dare vita ad una prospettiva completamente nuova all'ambiente circostante, destrutturando i concetti cardini della realtà e dell'immaginazione, consentendo di esperire in modo nuovo la vita ludica e professionale (Generali, 2020; Intel Corporation, 2021).

In quanto combinazione tra mondo fisico e virtuale, la Realtà Mista consente la rappresentazione di informazioni digitali mediante ologrammi, ossia oggetti costituiti da luce e suono e presenti nell'ambiente circostante. Questi ologrammi rispondono a comandi e interagiscono con il mondo concreto in tempo reale per garantire un'esperienza naturale e intuitiva (Microsoft, 2021).

La fusione che nasce tra realtà fisica e digitale consente nuove interazioni tra uomo, computer e ambiente. È l'esito di decenni di progressi tecnologici in diversi ambiti applicativi: dall'intelligenza artificiale, attraverso l'elaborazione grafica e alla tecnologia di visualizzazione, sino ai sistemi di input (Microsoft, 2021).

Originariamente, il nome Realtà Mista nasce nel 1994 quando Paul Milgram e Fumio Kishino scrissero un articolo intitolato "A Taxonomy of Mixed Reality Visual Displays" (Tassonomia dei dispositivi di visualizzazione nella Realtà Mista). Lo scritto analizza il concetto di continuum di virtualizzazione e la categorizzazione della tassonomia applicata agli strumenti di visualizzazione (Microsoft, 2021).

Realtà aumentata e realtà virtuale sono, quindi, da considerare come una piccola parte dello spettro della Realtà Mista, attraverso cui è possibile creare rappresentazioni digitali di persone, luoghi e cose del mondo reale (es. Windows 10) (Microsoft, 2021).

2.1 Ambiti applicativi

La tecnologia ha da sempre molteplici ambiti applicativi e, con il progredire dello sviluppo, ciò è sempre più evidente. Di seguito saranno delineati i diversi ambiti di applicazione rispettivamente per le due tematiche principali in questo contesto: Realtà Mista e robotica.

2.1.1 Realtà Mista

Tra gli ambiti in cui la Realtà Mista è entrata a far parte con successo c'è sicuramente quello sportivo (*Figura 2.1*). Ad oggi, tale tecnologia è stata sfruttata per simulare una competizione, aiutando gli atleti a prepararsi mentalmente e psicologicamente, immaginando le possibili varianti. Anche questo campo applicativo si prospetta in continuo sviluppo ed evoluzione (Technogym S.p.A., 2021).

Figura 2.1: Simulazione di Realtà Mista in ambito atletico-sportivo

Fonte: Technogym S.p.A., 2021

Con l'avanzamento tecnologico, ad esempio, un centometrista potrà allenarsi correndo accanto agli atleti più abili al mondo oppure con atleti della propria categoria, realizzati tramite ologrammi (Technogym S.p.A., 2021).

Figura 2.2: Utilizzo applicativo della Realtà Mista in ambito sportivo-automobilistico

Fonte: Technogym S.p.A., 2021

Allo stesso modo, un pilota potrà simulare la corsa provando direttamente in pista, più e più volte finché non si sentirà pronto. Potrà simulare la partenza o un sorpasso difficile, o un qualsiasi altro momento della gara. Inoltre, potrà studiare i movimenti degli avversari e provare a batterli con diverse strategie, come mostrato in *Figura 2.2*.

Tutto ciò è possibile senza alcun rischio reale per la salute, ma avendo istantaneo feedback di traiettorie o manovre che avrebbero portato all'impatto, se le altre macchine fossero state reali (Technogym S.p.A., 2021).

Sinteticamente, altri importanti ambiti di applicazione sono:

- Turismo. È possibile realizzare tour personalizzati e coinvolgenti con l'integrazione di elementi tridimensionali, per esempio, all'interno di una visita guidata in un museo. In pratica, gli elementi fisici possono prendere "vita" medianti gli ologrammi, rendendo l'esperienza più immersiva. Il visitatore può interagire concretamente con gli oggetti virtuali (*Figura 2.3*) (Macina, 2020).

Figura 2.3: Utilizzo applicativo della Realtà Mista in ambito turistico

Fonte: Macina, 2020

- Retail. È possibile creare un ambiente espositivo sovrapposto a degli oggetti fisici, i quali diventano animati una volta iniziata l'esperienza interattiva. Ad esempio, un venditore potrebbe presentare ai clienti l'intera gamma di prodotti, anche in assenza di grandi superfici espositive e, allo stesso modo, permettere loro di conoscere le reali dimensioni del prodotto nello spazio reale (Macina, 2020).

- Gaming. Giocare una versione particolarmente realistica del proprio videogioco preferito ora è possibile. Indossando il dispositivo olografico il giocatore interagisce con il mondo ludico grazie a gesti e comandi vocali, sia attraverso una visuale in prima persona, sia osservando lo scenario dall'alto (Macina, 2020).

- Militare. L'informatica ha cambiato profondamente anche il funzionamento di svariati sistemi d'arma e, di conseguenza, l'approccio in battaglia. Le innovazioni inserite, però, non si sono ridotte all'ambito di gestione degli armamenti e dei relativi componenti: l'effetto impatto è stato molto più radicale (Tucci, 2014).

Gli ultimi decenni hanno visto le nuove tecnologie condurre allo sviluppo di un ambiente simulato o di un mondo alternativo che trova origine mediante l'uso del computer. Tale simulazione è stata resa maggiormente realistica con l'aggiunta di periferiche – quali guanti e visori – che "immergono" colui che li indossa in una realtà alternativa capace di coinvolgerlo. Lo sviluppo di programmi e periferiche ha interessato, quindi, anche l'industria militare (Tucci, 2014).

Difatti, circa la produzione di simulatori – di carri armati, di volo, di navi e di fanteria – possono collaborare con famosi software di videogiochi. È questo il caso dell'azienda tedesca Crytek che ha generato la piattaforma tecnologica alla base del *Dismounted Soldier Training System* di *Intelligent Decisions*, il sistema di fanteria adoperato dall'esercito americano.

La *Figura 2.4* mostra un soldato durante un'esercitazione mentre utilizza un visore di Realtà Aumentata.

Figura 2.4: Utilizzo applicativo della Realtà Mista in ambito militare

Fonte: Tucci, 2014

In relazione ai simulatori di mezzi, invece, l'apparecchiatura include anche sistemi idraulici ingombranti fondamentali per creare spinte e sollecitazioni realistiche anche se, in realtà, tale tipo di riproduzione non possiede ancora il grado di libertà desiderato. In particolare, per gli abitacoli simulati non è ancora possibile riprodurre le accelerazioni gravitazionali positive e negative, sopportate da un pilota da caccia durante l'esecuzione di manovre rapide o di evitamento.

Il vantaggio di tali strumenti è chiaro: permettono di limitare le spese di addestramento senza la necessità di utilizzare munizioni vere e di riprodurre diversi scenari di impiego operativo (Tucci, 2014).

Al contempo, le innovazioni digitali hanno avviato la ricerca di soluzioni nuove per il bisogno, ampiamente sentito in ambito militare, di aumentare le informazioni a disposizione, andando oltre i naturali limiti sensoriali.

In questo campo di applicazione, un inserimento della tecnologia qui esposta ha riguardato i veicoli aerei da caccia con l'aggiunta dell'HUD (*Head up display*, visore a sovrimpressione), che consente di osservare i dati di volo (quota, velocità e beccheggio) e di ingaggio del bersaglio (lock-on) senza il bisogno di distogliere lo sguardo per monitorare la strumentazione di bordo.

Un ulteriore innovazione è l'impiego di un monocolo nei caschi di piloti e cannonieri degli elicotteri d'attacco AH-64 Apache statunitensi. Suddetto strumento visualizza i dati di navigazione e per l'aggancio dei bersagli.

È ormai chiaro che le tecnologie legate alla realtà mista sono risorse di significativo interesse per il campo militare (Tucci, 2014).

- Anche la Marina Militare Italiana impiega la Realtà Mista per l'addestramento, ma si focalizza sullo sviluppo di capacità procedurali e comunicative.

La Marina ha dimostrato l'efficacia del programma di addestramento commissionando un adattamento come game mobile (disponibile per iOS e Android) giocabile in modalità MR, che consente agli utenti di interagire con mezzi reali assumendo il comando di navi, aerei ed elicotteri, in scenari nazionali reali accuratamente ricreati, comprese la navigazione e l'esplorazione della nave scuola Amerigo Vespucci.

- Il Corpo Forestale ha sviluppato "*Forest Fire Area Simulator*", una simulazione MR che forma e addestra figure professionali e competenti impiegate per contrastare gli esiti degli eco-reati. Molte aree boschive aggredite dal fuoco sono riprodotte mediante tutti i sistemi e le attrezzature ordinariamente usate nelle reali operazioni (Gori, 2016).

- Industriale e aziendale. Le tecnologie immersive sono ampiamente usate anche nell'ambito dell'industria 4.0, ossia un insieme assai ampio di strumenti digitali, atti a rendere più integrate le parti dell'impresa e, in particolare, a unire in modo efficace l'impresa con l'ambiente esterno. Alcuni risultati previsti dall'impiego di nuove tecnologie in ambito aziendale sono: maggiore produttività e flessibilità, incremento dell'occupazione, ottimizzazione dei processi aziendali, migliore efficienza nei confronti dei clienti. Nel setting della formazione aziendale, utilizzando una piattaforma di AR, l'operaio può dunque seguire un percorso di training tenendo sotto controllo aspetti delicati quali il rischio di infortuni e la sicurezza sul luogo di lavoro (*Figura 2.5*). Inoltre, mediante una comunicazione più visiva e diretta, la persona potrebbe ricordare con più facilità i passaggi procedurali. Infatti, con l'uso dell'AR è possibile interagire con gli applicativi in svariati modi. L'utente può: ripetere

le procedure per un migliore apprendimento, selezionare quella più veloce o quella più facile, essere seguito con istruzioni multimediali, controllare gli applicativi mediante gesti o controlli vocali (Remondino, 2018).

Figura 2.5: Utilizzo applicativo della Realtà Mista in ambito industriale-aziendale

Fonte: Remondino, 2018

In sintesi, sono diversi i benefici che possono essere immaginati nell'area della formazione aziendale, utilizzando queste tecnologie sia per le imprese sia per gli impiegati (Remondino, 2018).

- Medicina. In questi ultimi anni, anche il mondo sanitario sta inseguendo la via dell'innovazione digitale nell'ideazione di nuovi strumenti e servizi. In generale, i campi di applicazione della MR in ambito medico sono principalmente: la terapia di disturbi psichiatrici; la riabilitazione motoria e cognitiva; l'apprendimento in un contesto simulato. In tali ambiti applicativi, l'ideazione di un mondo virtuale in cui ciascuno può avere il controllo si dimostra essere assai utile (Riba, 2018).

La cosiddetta "cyberterapia" esordisce dal bisogno di individuare nuove soluzioni in ambito riabilitativo-psicologico che implicano molti vantaggi rispetto alle metodologie riabilitative tradizionali (*Figura 2.6*). Riguardo a ciò, l'Istituto di Ricovero e Cura a Carattere Scientifico (IRCCS) Auxologico Italiano ha prodotto una versione 2.0 della riabilitazione fisica e cognitiva: l'istituto ospedaliero dispone di due "Cave", ossia stanze virtuali, dove si

sperimenta la "Telepresenza immersiva virtuale" (TIV) e si può simulare i tipici scenari in cui sono trattati disturbi cognitivi, motori conseguenti a ictus e Parkinson, o psicologici come ansia, fobie e stress. L'intento è quello di favorire l'esercizio delle funzioni danneggiate mediante attività con livelli di difficoltà crescente atti a gestire, ridurre, compensare o, nel migliore dei casi, superare i deficit (Riba, 2018).

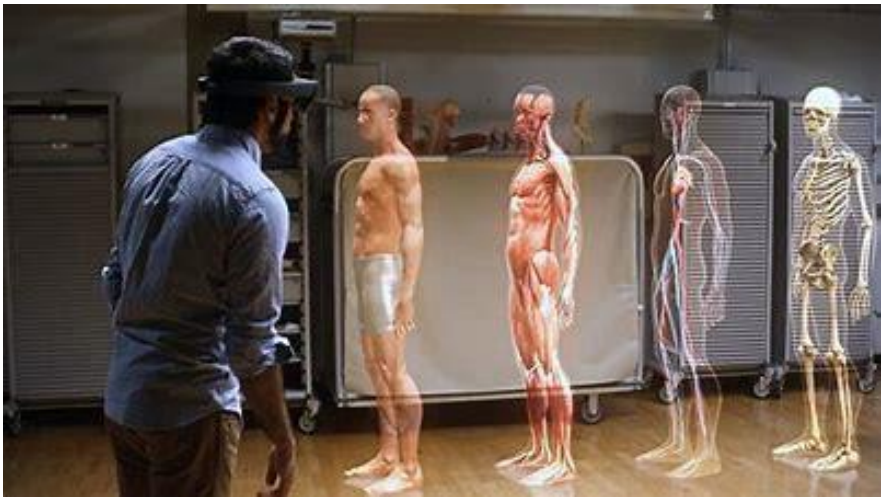


Figura 2.6: Utilizzo applicativo della Realtà Mista in ambito riabilitativo

Fonte: Riba, 2018

- Architettura. L'accesso alla conoscenza dei beni culturali rappresenta, ad oggi, un'occasione unica per le tecnologie: la sfida attuale è quella di modificare radicalmente il rapporto univoco tra patrimonio culturale e visitatore, proponendo, invece, nuovi processi cognitivi fondati innanzitutto sull'esperire in modo attivo e in cui le innovazioni tecnologiche divengono uno strumento necessario per stabilire una nuova accessibilità. Si immagina un mondo nel quale e del quale è possibile avere esperienza mediante modalità di approccio interattive che possano dimostrarsi valore aggiunto per la comunicazione, la diffusione e la costruzione di informazioni. La Realtà Mista, in tale ambito, propone diversi livelli di interattività ed immersività. L'obiettivo finale è quello di dare vita a nuove "rappresentazioni" di contenuti culturali capaci di predisporre modalità alternative di approccio alla conoscenza mediante tecnologie atte a stimolare un coinvolgimento emotivo (Rossi, Meschini, Feriozzi & Olivieri, 2018).

- Cultural heritage. La MR è, inoltre, adoperata per la conservazione e la diffusione virtuale del patrimonio immateriale. Tale tecnologia possiede innumerevoli potenzialità, ancora non completamente espresse, in campo culturale. La MR consente, in particolare, di creare modalità alternative di ripresa per registrare una particolare scena, quale per esempio

un'intervista, una cerimonia religiosa o un rito popolare. Offre la possibilità di essere immersi virtualmente all'interno della scena e di esperirla in prima persona, come mostrato in *Figura 2.7* (Porcari, 2018).

Figura 2.7: Utilizzo applicativo della Realtà Mista in ambito culturale

Fonte: Porcari, 2018

In conclusione, si rivela un utile strumento per i professionisti e un alternativo e interessante modo di visione per il cliente. Permette, poi, all'utente di cambiare il proprio *focus* di osservazione nella scena, girando semplicemente lo sguardo verso il punto che desidera guardare in dettaglio. Può sfruttare virtualmente la scena in Realtà Mista, la quale crea un'esperienza completamente immersiva e innovativa. In questo modo, l'utente diventa turista di qualsiasi monumento storico-culturale e visitatore di qualsivoglia luogo (Porcari, 2018).

2.1.2 Robotica

I robot sono strutture meccano-elettriche versatili e adattabili a diverse circostanze, abili nel produrre diverse attività elementari e costituiscono un enorme passo in avanti nella tecnologia.

Dal punto di vista scientifico, l'evoluzione della robotica nasce dalla congiunzione di svariate competenze, in origine appartenenti ad ambiti disciplinari distinti, con l'esito di sperimentare elementi di automazione sempre più adeguate ai bisogni dell'uomo (Istituto della Enciclopedia Italiana, 2017).

Difatti, gli ultimi decenni hanno consentito la diffusione della robotica, rendendola una delle tematiche più rinomate: sia la letteratura, sia il cinema ne hanno sfruttato appieno le potenzialità

per immergere lo spettatore in futuri mondi dominati da macchine intelligenti, capaci di accompagnare l'uomo in tutte le sue attività giornaliere.

Seppur la robotica sia una branca dell'ingegneria, in particolare della meccatronica, in essa confluiscono discipline e approcci molto diversi tra loro: la linguistica di natura prettamente umanistica, e al contempo, materie come biologia, fisiologia, psicologia, elettronica, fisica, informatica, matematica e meccanica appartenenti all'ambito scientifico (Tzafestas, 2020).

Questa tecnologia riveste, chiaramente, un ruolo alquanto fondamentale in ambito industriale; tantoché sta dando vita alla quarta rivoluzione industriale. L'industria 4.0 possiede sistemi automatizzati capaci di collaborare tra loro in una prospettiva di decentramento dell'uomo e di risparmio energetico dei consumi (Zimmermann, Berninger, Derkx & Rixen, 2020).

È chiaro che sia una modalità innovativa di produrre merce e di venderla in un mercato globale in continuo mutamento. I cambiamenti a cui l'industria e, in particolare, la robotica ha condotto negli ultimi anni sono innumerevoli.

Infatti, questo tipo di innovazione conduce ad un modo nuovo di pensare e creare i prodotti: un qualsiasi prodotto può essere personalizzato online, preso in carico da specifici sistemi informatici, ideato dalle macchine di produzione e, infine, realizzato automaticamente in fabbrica (Zimmermann et al., 2020).

Oggi, però, la robotica si applica anche all'ambito medico, coadiuvando così il lavoro dell'uomo, senza sostituirlo, e avendo come obiettivo principale e comune il benessere del paziente. Ne sono un chiaro esempio i robot, gestiti a distanza da chirurghi professionisti, che nelle sale operatorie di tutto il mondo possono raggiungere abilmente le zone anatomiche della persona, avendo a disposizione una visione precisa e minuziosa.

In particolare, i robot medici possiedono braccia meccaniche, dotate di strumenti ad hoc per effettuare le operazioni, e di una telecamera, al fine di accedere visivamente a tutte le zone anatomiche a cui è diretta l'operazione medica.

Le ipotesi più stravaganti e innovative reputano che, in un futuro non troppo remoto, le sale operatorie saranno dotate di micro-robot e nano-robot, specifici per singole tipologie di intervento. In egual misura, robot di dimensioni estremamente ridotte sono già usati per diagnosi e prevenzione di alcune patologie, poiché riescono ad accedere agli organi interni e a studiarli in modo minuzioso ma al tempo stesso non invasivo (Tzafestas, 2020).

Oltre al campo medico, la robotica si adatta finemente anche a quello riabilitativo, dove gli utenti con disabilità permanenti o temporanee possono sfruttare gli esoscheletri oppure robot fisioterapisti per cercare di recuperare la mobilità, parziale o totale, degli arti. In futuro, si presuppone un loro maggiore utilizzo e, di conseguenza, una sempre maggiore efficacia.

L'utilizzo della robotica, però, non si limita al settore medico e produttivo, bensì ha un effetto ingente su tutte le mansioni professionali e sulle aziende, senza limitare il ruolo dell'uomo in tale contesto. Infatti, studi dimostrano che la scelta di adoperare la robotica non limita il lavoro manuale, bensì conduce ad aumentare le assunzioni del personale.

Ne sono un chiaro esempio gli esoscheletri per arti superiori, i quali possono essere utilizzati per permettere all'uomo di sollevare fino a 100 kg senza alcuno sforzo, e i visori per la AR, proprio come gli HoloLens di Microsoft, che richiedono la presenza di specifiche figure professionali addette alla vendita di suddetti strumenti (Zimmermann et al., 2020).

È ormai chiaro che la robotica sia entrata, a tutti gli effetti, a far parte della vita quotidiana anche all'interno delle singole abitazioni: a partire dai numerosi kit dedicati alla sicurezza domestica che, mediante appositi sensori, consentono di tenere sotto controllo l'ambiente e notificare la presenza di eventuali intrusi anche a molti chilometri di distanza; sino a strumenti Smart, come prese della corrente, lampadine o termostati ma anche robot aspirapolvere o robot macchinette del caffè, che possono essere facilmente regolati dal proprio smartphone.

In particolare, per l'intrattenimento casalingo la robotica ha dato vita a speaker intelligenti, ossia casse connesse a sistemi di intelligenza artificiale, come il famoso Google Assistant o i celebri dispositivi Amazon, grazie ai quali è possibile ascoltare musica in modalità wireless, ricevere informazioni in tempo reale sul meteo o sul traffico, impostare sveglie o promemoria e interagire con altri prodotti Smart in casa, quali Smart TV o lampadine (Young & Peschel, 2020).

In conclusione, la robotica ha fatto grandi passi in avanti negli ultimi decenni, ma in realtà è solo all'inizio di una vera e propria rivoluzione che probabilmente condurrà alla diffusione di massa dei sistemi di Smart Home.

Grazie allo sviluppo della robotica, la manipolazione in tal senso è divenuta punto cardine di molti ambiti applicativi. In dettaglio, un manipolatore permette di spostare oggetti di diverse dimensioni, forma e orientamento e consente di lavorare con singole parti meccaniche quali, per esempio, schede elettroniche o vernici. La manipolazione può essere, inoltre, bimanuale come nel caso di alcuni robot industriali (Siciliano, 2018).

Tutte queste funzioni prendono il nome di "teleoperazioni di un robot": un utente controlla un robot da una stazione remota, manipolando quindi un robot non autonomo. Tendenzialmente, la manipolazione avviene da una postazione di controllo, dotata di tutti gli strumenti necessari per la guida es. joystick) e il monitoraggio (display) del robot (Laschi, 2015).

In particolare, può avvenire mediante differenti strumenti, tra cui tastierini e joystick che consentono di spostare il robot nello spazio. Il tastierino permette di spostare il robot, ideare percorsi e fornire manualmente le traiettorie. Nello specifico, le suddette traiettorie sono insegnate al robot, detto *collaborative robot*, utilizzando la modalità *Lead-Through*, ossia un metodo

di programmazione che richiede la guida del manipolatore compiendo specifici movimenti e registrandoli nella memoria del computer del robot (Groover, 2008; Conterno, 2019).

Il joystick, invece, è utilizzato per controllare la velocità di movimento e la direzione. Lo spostamento, avanti o indietro, del joystick corrisponde direttamente al movimento eseguito dal manipolatore. In particolare, lo spostamento, a destra o a sinistra, serve a riorientare il robot. L'utilizzo del joystick rappresenta indubbiamente il metodo più intuitivo (ABB, 2018).

La programmazione del robot, per garantire il movimento desiderato, avviene mediante uno specifico approccio. Il più famoso e utilizzato prende il nome di "Approccio Programmazione per Dimostrazione". Quest'ultimo consiste in diverse fasi a partire dalla dimostrazione dell'utente, costruendo una rappresentazione generalizzata dei dati segmentati e infine eseguendo il task dimostrato (Dillmann, 2004). Nella pratica, si presenta come una modalità tecnologica e innovativa di apprendimento per imitazione (Billard, Calinon, Dillmann & Schaal, 2018).

In particolare, la programmazione per dimostrazione (PbD) è diventata un argomento centrale della robotica e comprende molte aree di studio tra cui: interazione uomo-macchina, machine learning, machine vision e controllo del moto (Billard et al., 2018).

Storicamente, la PbD nasce all'incirca 30 anni fa ma è a partire dall'ultimo decennio che ha subito una crescita evidente. La strategia di preferire interfacce utente flessibili, a robot programmati, ha tre obiettivi essenziali:

- innanzitutto, la PbD è uno strumento efficace per limitare la complessità degli spazi di ricerca per l'apprendimento. Distinguendo, infatti, le buone soluzioni da quelle cattive o meno efficaci permette di ridurre in concreto la ricerca. L'apprendimento per imitazione consente, quindi, di migliorare e velocizzare il processo di acquisizione di informazioni;

- l'apprendimento per imitazione, inoltre, mette a disposizione un mezzo implicito per addestrare la macchina, cosicché la parte dedicata alla programmazione, esplicita e noiosa, sia minimizzata o addirittura eliminata per l'utente;

- infine, studiare e modellare l'associazione percezione-azione, focus dell'apprendimento imitativo, permette di comprendere meglio i meccanismi mediante i quali l'auto-organizzazione della percezione e dell'azione potrebbe sorgere durante lo sviluppo. L'interazione bidirezionale di percezione e azione potrebbe definire come la competenza nel controllo motorio possa essere profondamente inserita in una struttura ricca di variabili percettive e, al contempo, come i processi di percezione possano svilupparsi come strumenti utili a creare azioni di successo (Billard et al., 2018).

Considerando la velocità di sviluppo e diffusione della robotica, occorre monitorare altri ambiti applicativi che potrebbero rivelarsi affini, infatti tali ambiti non risultano oggi essere applicati alla robotica ma, in futuro, potrebbero dimostrarsi interessanti.

Soluzioni di customer self-service

Si tratta di applicazioni fisiche che, integrando sistemi di IA finalizzati, supportano meglio gli utenti: ad esempio un touchscreen intelligente con il quale la persona può interagire con un linguaggio naturale. Sono soluzioni che potrebbero diffondersi rapidamente sia come servizi digitali pubblici nel turismo sia come soluzioni innovative nel settore del retail.

Robot e assistenti alle vendite

Quello del retail, infatti, è uno degli ambiti nel quale l'intelligenza artificiale può evidenziare il suo massimo potenziale. I robot con capacità motoria simile a quella umana sono ancora in fase di sviluppo e sperimentazione ma sono già diverse le aziende che hanno iniziato progetti concreti in tal senso. Per esempio, sono stati impiegati robot per l'accoglienza e l'assistenza nella vendita al dettaglio (ICT Value Consulting, 2018).

Sicurezza pubblica

La possibilità di analizzare enormi quantità di dati in tempo reale e di "cogliere" abitudini, comportamenti, attitudini, sistemi e dati di geo-localizzazione e monitoraggio degli spostamenti di cose e persone, offre un grandissimo potenziale per migliorare l'efficacia della sicurezza pubblica. Ad esempio, è possibile ampliare la prevenzione di crimini in luoghi come aeroporti, stazioni ferroviarie e città metropolitane oppure, allo stesso modo, è possibile provvedere ad una migliore gestione di crisi in caso di calamità naturali (ICT Value Consulting, 2018).

Alla luce di quanto detto sinora, in conclusione, l'HoloLens 2 è indubbiamente il miglior dispositivo per la Realtà Mista accessibile sul mercato. Questo, infatti, può essere utile a obiettivi e figure professionali differenti all'interno della stessa azienda o dello stesso ambito applicativo. È chiaro che ogni settore sia definito da scopi specifici, ma non esiste un ambito per il quale possano non essere utili. Le aziende hanno sempre più bisogno di una soluzione semplice per le proprie sfide.

L'HoloLens 2 offre, inoltre, l'esperienza di Realtà Mista più semplice e immersiva possibile mediante soluzioni leader; garantisce affidabilità, sicurezza e scalabilità del cloud e dei servizi di intelligenza artificiale di Microsoft. È immersivo, ergonomico, istintivo e senza cavi superflui (Aurea s.r.l., 2020).

Inizialmente, fino al secolo scorso, l'automazione e i robot erano limitati al settore automobilistico ma con l'avvento della Globalizzazione, questo settore di nicchia si è diffuso, in virtù anche della riduzione dei costi economici ed energetici di tali tecnologie, e dell'aumento produttivo (Maldera, 2020).

2.2 Realtà Aumentata e Mista a supporto della Robotica

Le ricerche in ambito robotico sono in continuo svolgimento e lo studio, in tal senso, può dirsi tutt'altro che concluso. In particolare, gli articoli presentati qui di seguito servono a sottolineare l'importanza di indagini continue e a illustrare, se pur brevemente, i progressi in relazione alla Realtà Mista.

1. Human-robot interaction for robotic manipulator programming in Mixed Reality

L'articolo propone un approccio per l'interazione con un robot manipolatore utilizzando la Realtà Mista. Lo studio è stato compiuto mediante l'uso degli HoloLens, il framework ROS e Unity3D (Ostanin, Mikhel, Evlampiev, Skvortsova & Klimchik, 2020).

Lo studio ha previsto la scelta di due specifici robot: KUKA e UR10e. Per quanto riguarda, invece, il sistema di sviluppo, esso può essere diviso in tre parti fondamentali: la parte di visualizzazione e interfaccia, compiuta attraverso un dispositivo Microsoft HoloLens; la parte di computing svolta da ROS Kinetic; la parte "real-word", che comprende i robot fisici.

In particolare, l'applicazione consente di individuare nello spazio la posizione del robot reale attraverso l'analisi a nuvola di punti; per la creazione di task e di traiettorie eseguibili dal robot virtuale, utilizza marker virtuali e menù appositi. Essa è stata sviluppata utilizzando il game engine Unity3D e il framework *Mixed Reality Toolkit* (MRTK) per l'interazione con l'HMD HoloLens; è composta da un'interfaccia, un *geometrical path planning* per la creazione del path virtuale, una *spatial mapping* responsabile della visualizzazione del mondo reale e i modelli virtuali dei robot. Il path creato dall'utente, poi, è pubblicato su un nodo ROS e il framework MoveIt! è responsabile della pianificazione della traiettoria, ovvero lo studio delle velocità e delle accelerazioni di ogni giunto, istante per istante (Ostanin, Mikhel, Evlampiev, Skvortsova & Klimchik, 2020).

Il sistema dispone, nello specifico, delle seguenti funzionalità:

- *Menù*: utile per interagire con gli oggetti in Realtà Mista. L'utente ha la possibilità di scegliere tra le diverse opzioni che l'applicazione mette a disposizione e permette l'accesso

alle funzioni per modificare i settaggi dei robot o l'utilizzo dei comandi per la pianificazione della traiettoria;

- *Robot state*: consiste in un'interfaccia che permette di visualizzare lo stato del robot, identificata la posizione dei giunti e la posa dell'end-effector;

- *Geometrical path planning*: rappresenta il path compiuto dall'end-effector attraverso una sequenza di punti di controllo. Ognuno di essi rappresenta la posizione e l'orientamento della punta operativa della catena cinematica in uno spazio a sei gradi di libertà;

- *Trajectory generation*: crea la traiettoria del robot utilizzando il file *Unified Robot Description Format* (URDF), tenendo conto delle velocità massime, delle accelerazioni massime e del workspace del robot;

- *Simulator*: responsabile della simulazione in Realtà Mista del movimento del robot secondo le specifiche della traiettoria generata.

- *Workspace visualization*: permette di visualizzare lo spazio di lavoro del robot indossando gli HoloLens (Ostanin, Mikhel, Evlampiev, Skvortsova & Klimchik, 2020).

Oltre a queste funzionalità base, sono state sviluppate delle *feature* basate sull'integrazione con la Realtà Mista, tra cui (Ostanin et al., 2020):

- *Robot placement*: localizza il robot reale e la sua posa attraverso un algoritmo *iterative closest point* (ICP);

- *Shortest path avoiding obstacles*: se all'interno dello spazio di lavoro del robot sono presenti ostacoli, l'applicazione calcola automaticamente un path alternativo per collegare i punti di controllo sulla scena, con il minor costo possibile. Per consentire ciò, è utilizzato un algoritimo *n Rapidly-exploring Random Tree* (RRT).

- *Path scaling*: permette di scalare la grandezza degli ologrammi per creare path più accurati.

L'applicazione finale consente di svolgere operazioni di *pick and place,* ovvero raccogliere un oggetto nello spazio reale e spostarlo in un'altra posizione, e operazioni di contatto, in cui un oggetto è spostato attraverso un determinato path per operazioni come disegno, incollaggio o verniciatura (Ostanin et al., 2020).

2. An Augmented Reality Interface for Human-Robot Interaction in Unconstrained Environments

L'articolo propone uno studio che utilizza un'interfaccia in Realtà Aumentata per l'interazione uomo-robot in uno spazio di lavoro condiviso. L'applicazione utilizza sia tecnologie marked-based, sia tecnologie markerless AR, al fine di localizzare la posizione del robot reale e visualizzare il suo spazio di lavoro sul piano (Chacko & Kapila, 2019).

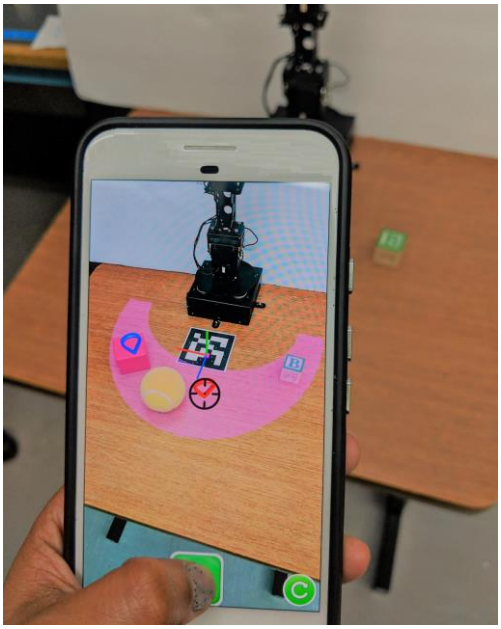


Figura 2.8: Esempio di utilizzo della AR in ambito robotico

Fonte: Chacko & Kapila, 2019

Lo scopo dell'applicazione è creare un ambiente virtuale che consenta all'utente di svolgere compiti di *pick and place* senza sforzo attraverso l'utilizzo di uno smartphone. I target virtuali posti sulla scena sono utilizzati per individuare le posizioni di *pick and place* degli oggetti fisici. I punti di partenza e di arrivo di questi ultimi sono calcolati inizialmente nello spazio operazionale della camera dello smartphone e poi convertiti nello spazio operazionale del robot che automaticamente compirà il task specifico.

La ricerca è stata condotta con l'uso di uno smartphone Google Pixel XL, un manipolatore robotico a quattro gradi di libertà e un computer per processare i dati derivanti dal device. L'applicazione è stata creata adoperando il game engine Unity3D e la libreria ARCore. Il robot è posto su un una superficie piana accanto ad un singolo marker. L'utente sceglie le posizioni in cui il robot dovrà eseguire la manovra di presa e spostamento con un'interazione touchscreen sul suo smartphone e invierà i dati al pc mediante collegamento bluetooth. Queste informazioni

saranno elaborate e inviate al robot per adempiere il compito predefinito (Chacko & Kapila, 2019).

L'interfaccia dell'applicazione, mostrata in *Figura 2.8*, è formata da due pulsanti: quello principale, posto centralmente sullo schermo, è utilizzato per scegliere le posizioni target nello spazio operativo. Il secondo bottone, posizionato in basso, è usato per resettare i target ed eliminare dalla scena gli oggetti virtuali precedentemente creati.

Avviata l'applicazione, la fotocamera dello smartphone individua il marker e pone un'ancora in quella precisa posizione che sarà utilizzata come punto di riferimento per il robot. Una volta conclusa la fase di localizzazione, l'utente può inserire manualmente i target all'interno della scena (Chacko & Kapila, 2019).

3. Augmented Reality for Interactive Robot Control

Scopo dello studio è stato quello di sfruttare la Realtà Mista al fine ideare un ambiente virtuale per controllare un manipolatore robotico. Questo processo ha inizio quando l'utente pone il robot virtuale nello spazio 3D e successivamente potrà scegliere di operare con il controllo manuale o con quello automatico (Manring, Pederson, Potts, Boardman, Mascarenas, Harden & Cattaneo, 2019).

Con il controllo manuale, è possibile spostare l'end-effector del manipolatore virtuale e, automaticamente, quello reale compirà il medesimo movimento. Nel controllo automatico, invece, l'utente può posizionare l'end-effector del robot virtuale in una qualsiasi posizione, osservare la simulazione e far eseguire lo stesso movimento al robot reale o modificare la posizione se non soddisfatto. Questo controllo consente una duplice funzione: vedere la traiettoria che sarà eseguita dal robot e, al contempo, la rotazione dei suoi giunti. Sarà così garantito un miglior feedback sul movimento prescelto.

Per implementare il controllo del robot sono stati adoperati alcuni strumenti: un HMD HoloLens per dare vita all'ambiente virtuale, un manipolatore robotico Yaskawa Motoman SIA5D, Unity3D per lo sviluppo applicativo e ROS per l'interazione con il robot (Manring, Pederson, Potts, Boardman, Mascarenas, Harden & Cattaneo, 2019).

Nello spazio virtuale, è possibile interagire con degli ologrammi, attraverso le API fornite dall'HoloLens, per selezionarli e spostarli da un punto di partenza ad uno di arrivo. L'HMD, invece, invia e riceve dati sul manipolatore mediante l'uso di un websocket ROSbridge.

In pratica, a seconda del metodo di controllo selezionato dall'utente, il dispositivo HoloLens pubblica su un topic ROS la posizione e l'orientamento scelti per l'end-effector del

manipolatore. Questi dati saranno elaborati e i risultati saranno pubblicati su un nuovo topic ROS di *motion planning* su cui è in ascolto il dispositivo.

L'HoloLens adopera i risultati per illustrare un'anteprima del movimento del robot reale mediante un ologramma, dopodiché l'utente potrà inoltrare liberamente le informazioni al robot reale cosicché riproduca il movimento precedentemente simulato in ambiente virtuale (Manring et al., 2019).

4. AR based robot programming using teaching by demonstration

L'articolo "*AR based robot programming using teaching by demonstration*" di Konstantinos Lotsaris, Christos Gkournelos, Nikos Fousekis, Niki Kousi e Sotiris Makri, risalente al 2021, presenta l'applicazione di uno strumento AR utile a supportare l'interazione uomo-macchina all'interno di un sistema di produzione.

Per questo studio è stata sviluppata un'applicazione in Realtà Aumentata che permette all'utente di interagire con un manipolatore e di spostarlo tramite dimostrazione. Lo scopo dell'applicazione è stato quello di accelerare e rendere più semplice il processo di produzione industriale introducendo un semplice strumento che consente l'interazione con il robot senza che sia necessaria una capacità di programmazione da parte dell'utente, o lunghi tempi di formazione. L'applicazione è stata sviluppata per l'HMD HoloLens e la parte di manipolazione robotica è stata testata tramite il toolkit ROS. In particolare, il manipolatore robotico utilizzato è stato UR10 (Lotsaris, Gkournelos, Fousekis, Kousi & Makris, 2021).

L'applicazione proposta, quindi, assiste l'operaio sia nella fase di programmazione del movimento, sia nella parte di esecuzione della traiettoria: l'utente può utilizzare l'applicazione per specificare dei punti di riferimento al manipolatore ma anche muoverlo manualmente nel caso di fallimento di un task assegnato.

Dal punto di vista dell'utente, l'applicazione deve essere semplice e intuitiva per facilitare l'interazione con il robot. Questo è possibile attraverso un'interfaccia molto minimale: presenta tre pulsanti locati sotto un gripper virtuale. L'utente può spostare la pinza nello spazio con i gesti di *Hold* e *Drag*, può ruotare il gripper attraverso il bottone *Rotate* per modificarne l'orientamento e, infine, premere il tasto *Plan* per generare una traiettoria e visualizzarla nello spazio fisico o premere il tasto *Execute* per spostare il robot nella posizione definita dalla pinza virtuale (Lotsaris et. al, 2021).

L'applicazione si può suddividere in tre fasi fondamentali:

1. Fase di inizializzazione: per fare in modo che l'applicazione funzioni, è necessario che il sistema di riferimento dell'HMD sia collegato a quello del robot. In questo studio, la connessione è eseguita mediante la scansione di un codice QR, fissato in una posizione nota rispetto alla base del robot. L'HoloLens, inoltre, utilizza tecniche markerless per il posizionamento degli oggetti nello spazio virtuale; il metodo marker-base è utilizzato solo in fase di inizializzazione per il collegamento dei sistemi di riferimento. Quindi, in fase di avvio dell'applicazione, l'utente deve posizionarsi di fronte al codice QR e scattare una foto. L'immagine sarà elaborata per calcolare la distanza tra HMD e robot, in questo modo ogni punto, inserito dall'utente nello spazio virtuale, potrà essere trasformato in un punto corrispondente nel sistema di riferimento del robot. Infine, un robot virtuale è sovrapposto al robot reale per avere un feedback sulla bontà del processo di calibrazione.

2. Fase di istruzione del robot: dopo la fase inizializzazione da parte dell'utente, quest'ultimo potrà manipolare una pinza virtuale, simile a quella presente sul robot. La pinza appare nella stessa posizione e nello stesso orientamento di quella fisica (Lotsaris et. al, 2021).

Per posizione il target virtuale nella posizione desiderata, l'utente potrà usare il tocco e successivamente modificare l'orientamento premendo il bottone Rotate. Una volta conclusa la fase di posizionamento, l'operatore può premere il tasto Plan per richiedere al robot di generare una traiettoria. Se la pinza virtuale si trova in una posizione raggiungibile dal manipolatore, questa diventa verde e un robot virtuale simulerà il path che il robot reale seguirebbe. Se, invece, la pinza sarà al di fuori della zona operativa del robot, si colorerà di rosso e l'utente sarà libero di riposizionarla nello spazio virtuale.

3. Fase di esecuzione: Una volta completata la fase precedente, l'utente può premere il pulsante Execute per inviare al robot il comando di movimento. Il manipolatore inizierà la sua traiettoria e, al contempo, apparirà un bottone "annulla" sull'interfaccia. Se l'operatore desidererà bloccare il robot dovrà semplicemente toccare il pulsante; in caso contrario, il manipolatore compirà l'intero movimento e si fermerà nella posa finale desiderata (Lotsaris et. al, 2021).

5. Interactive Robots Control Using Mixed Reality

Nel 2019, invece, Mikhail Ostanin, Alexandr Klimchik e Rauf Yagfarov hanno presentato un articolo intitolato "Interactive Robots Control Using Mixed Reality", il quale illustra un approccio basato su tecniche AR per il controllo di un manipolatore robotico in modo interattivo. In particolare, è stata progettata un'interfaccia semplice e intuitiva per l'interazione

uomo-robot: fornisce strumenti utili alla programmazione del path del robot e fornisce un feedback ai lavoratori per aiutarli a comprendere meglio il comportamento del robot durante il movimento (Ostanin, Klimchik & Yagfarov, 2019).

Per il progetto descritto nell'articolo è stato utilizzato un robot industriale KUKA, una piattaforma mobile Plato e l'HMD HoloLens. È stata evidenziata una difficoltà, relativamente al sistema multipiattaforma, nella sincronizzazione dei frame di coordinate per tutti gli elementi. Per la risoluzione del problema sono state realizzate tre opzioni di impostazione: manualmente, da una fotocamera con marker, attraverso elaborazione a nuvole di punti.

Il sistema utilizzato può essere diviso in tre parti principali: l'interfaccia, il controller e il robot. Tuttavia, il robot può essere presente sia nell'ambiente reale, sia in quello virtuale allo stesso tempo grazie al dispositivo Microsoft HoloLens, che permette la connessione tra i due ambienti. L'unità di controllo è collegata a entrambe le parti e realizza calcoli, trasferimento di comandi tra l'interfaccia e robot, movimenti di ologrammi e controllo del manipolatore.

Riassumendo, l'HoloLens rappresenta l'interfaccia MR e l'ambiente virtuale, mentre il framework ROS Kinetic è il controller (Ostanin et al., 2019).

Una parte importante del sistema è la risoluzione del problema del confronto di sistemi di coordinate. Questo può essere risolto in più modi: impostazione manuale dell'origine, rilevamento dell'origine con aiuto di marcatori (QR, Aruco, ecc.) o confrontando le nuvole di punti ottenute con l'utilizzo dell'HoloLens tramite il Lidar 3D Velodyne VLP-16. Nel primo caso, l'utente posiziona l'origine del sistema delle coordinate nel punto in cui la piattaforma robotica mobile è visualizzata dall'HoloLens; nel secondo, invece, si usa un'etichetta che è visualizzata dalla fotocamera degli HoloLens e che sarà utilizzata come riferimento. Il terzo metodo è il più flessibile e conveniente, in quanto i sistemi di coordinate sono rilevati e determinati automaticamente senza l'ausilio di target o intervento manuale.

Il confronto dei sistemi di coordinate si basa sul confronto delle nuvole di punti e richiede più fasi:

1. tramite l'HoloLens si esegue la scansione della stanza e si crea una nuvola di punti;

2. con 3D LIDAR, posizionato sulla piattaforma mobile, si ottiene una seconda nuvola di punti da una scansione;

3. con un algoritmo Iterative Closest Point (ICP), si confrontano le nuvole di punti e si ottiene una matrice di trasformazione di un sistema di coordinate a un altro.

All'avvio dell'applicazione, all'utente apparirà un menù virtuale con le opzioni per impostare e controllare il robot.

Il menù ha due schede principali, corrispondenti al robot KUKA e al robot Plato. Le schede sono composte dallo stesso insieme di comandi:

- *Set Up*: visualizzazione di un modello di robot virtuale sovrapposto a quello reale;
- *Add Point*: aggiunta di un punto di controllo per la traiettoria del robot;
- *Delete Point*: eliminazione dell'ultimo punto inserito;
- *Send*: invio del path al robot.

La modifica dei punti di controllo è possibile in qualsiasi momento e il percorso del robot è ricostruito automaticamente (Ostanin et al., 2019).

Data la diversa struttura dei due robot (il manipolatore KUKA ha 6 DOF e la piattaforma Plato 3 DOF), il processo di configurazione dei punti di controllo è diverso: per la piattaforma, la posizione è espressa in coordinate XY sul pavimento e l'orientamento intorno all'asse Z. Per l'end-effector del KUKA, invece, è necessario settare tutti sei i gradi di libertà di posizione e orientamento nello spazio.

La selezione per inserire un punto di controllo per la piattaforma mobile consiste in un semplice tocco sul pavimento, questo farà apparire un cursore indicante la posizione del punto; l'utente può manipolarlo per modificare l'orientamento. Per il manipolatore KUKA, invece, ci sono due opzioni. Nella prima, la posizione è inserita dall'utente tramite tocco; il punto di controllo inserito è composto da dei cerchi che permettono di modificarne l'orientamento. La seconda opzione consente di selezionare un punto nell'area di lavoro del robot. La posa del target sarà impostata con orientamento lungo la normale alla superficie e resta da impostare solo l'orientamento attorno alla normale dell'asse (Ostanin et al., 2019).

6. Interactive Robot Programing Using Mixed Reality

È ormai chiaro che, grazie al costante sviluppo tecnologico, si assiste sempre più all'incremento dei robot utilizzati in ambito industriale: sono utilizzati per l'automatizzazione di vari processi. Lo studio "*Interactive Robot Programing Using Mixed Reality*" di Mikhail Ostanin e Alexandr Klimchik, del 2018, presenta un sistema per la programmazione interattiva per i robot industriali basata sulla Realtà Mista. In particolare, sono utilizzati gli occhiali Microsoft HoloLens per l'immersione e, con il sistema sviluppato, qualsiasi utente senza competenze di programmazione può assegnare un task al manipolatore. Le parti principali del sistema sono la pianificazione del

percorso geometrico, l'ottimizzazione dei tempi, la pianificazione della traiettoria e la simulazione basata su MR (Ostanin & Klimchik, 2018).

L'applicazione, poi, ha un'interfaccia intuitiva che aiuta gli utenti a interagire con l'ambiente circostante e con i due manipolatori, che si differenziano per il modello cinematico, utilizzati per lo studio: il robot KUKA Agilus e il robot KUKA IIWA. Il sistema è costituito da oggetti fisici e oggetti virtuali: gli oggetti fisici sono l'HMD HoloLens e i due manipolatori, il controller, un computer e l'ambiente di lavoro; gli oggetti virtuali del sistema sono i modelli dei robot, l'interfaccia, la mappa virtuale dello spazio di lavoro, costruita tramite l'HoloLens utilizzando i parametri geometrici dell'ambiente reale, e gli strumenti di interazione. Inoltre, per creare il sistema basato su HoloLens è stato utilizzato il game engine Unity3D con Mixed Reality Toolkit (MRTK) (Ostanin & Klimchik, 2018).

La parte software del sistema si può dividere in quattro blocchi:

- *Application Manager*: responsabile del controllo di tutti i blocchi;
- *Trajectory Planner*: calcola le velocità e le accelerazioni dei giunti che saranno applicate alla simulazione;
- *Geometrical Path Planner*: invia al Trajectory Planner la sequenza di punti contenenti la propria posizione cartesiana e l'orientamento con gli angoli YPR;
- *Simulations*: la simulazione effettiva del software.

L'interazione uomo-robot virtuale avviene utilizzando gli HoloLens, i gesti, i menù e gli strumenti virtuali. Ci sono diversi metodi per programmare la traiettoria del robot: per esempio, per un movimento PTP, l'operatore specifica i punti di partenza e di arrivo del manipolatore e configura il loro orientamento. Conclusa la parte di posizionamento, il path geometrico è calcolato ed è restituita la traiettoria che il robot deve seguire. La traiettoria può essere visualizzata sul robot virtuale e, se l'utente è soddisfatto, inviata al manipolatore reale (Ostanin & Klimchik, 2018).

L'interazione attraverso l'HoloLens avviene tramite gesti:

- Tocco, un semplice click;
- Tocco e tenere premuto, utilizzato per interazioni più complesse quando combinato con il movimento del braccio;

- Manipolazione, utile per spostare gli ologrammi nella scena.

All'avvio dell'applicazione l'utente visualizza il menù da cui è possibile: selezionare il modello e la configurazione del robot, visualizzare lo stato del manipolatore e controllare la traiettoria. Il soggetto visualizza, inoltre, una mappa spaziale dell'ambiente circostante, utile per analizzare il mondo e trovare dello spazio libero per pianificare il movimento. Consente, inoltre, all'utilizzatore di selezionare un punto direttamente sulla superficie di un oggetto reale. Un oggetto virtuale è utilizzato per specificare il frame di coordinate che descrive la matrice di trasformazione che l'end-effector del manipolatore avrà nella posa finale desiderata.

Il percorso geometrico del manipolatore è descritto da una sequenza di punti di controllo. Questi punti possono essere posizionati e orientati in due modalità: il punto è inserito con orientamento normale al piano, oppure l'utente può inserire un punto in una qualsiasi posizione e modificarne l'orientamento manualmente (Ostanin & Klimchik, 2018).

Il percorso tra due posizioni target può essere effettuato in diversi modi:

- *Point to point* (PTP), quando ogni punto corrisponde a una configurazione. Il manipolatore si sposterà dalla configurazione iniziale a quella finale;

- *Line*, se la linea corrisponde ad una sequenza di punti intermedi. Il passaggio da un punto intermedio all'altro è compiuto tramite un movimento PTP;

- *Arc*, è simile alla linea, ma l'utente inserisce un punto intermedio senza orientamento tra il punto iniziale e finale;

- *User path*, quando l'utente può disegnare un percorso tra due punti usando un puntatore virtuale;

- *Without collision path*, se il sistema trova il path più breve nello spazio cartesiano.

Per il percorso, o path, più breve nello spazio cartesiano, si costruisce un grafo pesato in cui ogni vertice corrisponde a un cubo tra due target. Un arco connette i cubi più vicini e il suo peso corrisponde alla distanza tra i centri dei cubi. Inoltre, se uno dei cubi è posizionato su un ostacolo, questo è ignorato.

Il percorso più breve del grafo corrisponde alla traiettoria geometrica desiderata. Mentre, se il path non è trovato, l'analisi spaziale è incrementata.

Oltre ai sopra citati studi, ne esistono molti altri. Se ne illustrano qui alcuni, per ulteriori informazioni occorre fare riferimento alla vasta letteratura esistente sul tema.

Il documento intitolato "*Augmented Reality (AR) Applications for Supporting Human-robot Interactive Cooperation*" (*Figura 2.9*), proposto da studiosi quali Michalosa G., Karagiannisa P., Makrisa S., Tokçalarb Ö., Chryssolourisa G. nel 2016, presenta uno strumento di Realtà Aumentata (AR) per il supporto degli operatori in situazioni in cui uomini e robot cooperano in un luogo di lavoro industriale condiviso. Il sistema fornisce la visualizzazione AR del processo di assemblaggio, istruzioni basate su video e testo e aggiornamenti sullo stato della produzione. Il prodotto è pensato per migliorare la sicurezza dell'utente e l'accettazione di ambienti di assemblaggio ibridi attraverso le capacità di immersione della tecnologia AR (Michalosa, Karagiannisa, Makrisa, Tokçalarb & Chryssolourisa, 2016).

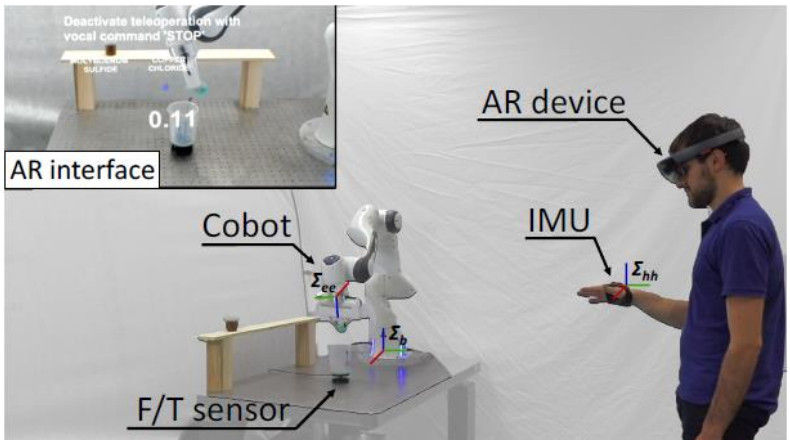


Figura 2.9: Esempio di interazione uomo-robot

Fonte: Michalosa, Karagiannisa, Makrisa, Tokçalarb & Chryssolourisa, 2016

Un ulteriore studio sul tema, risalente al 2018, che prende il nome di "*Robot Programming Through Augmented Trajectories in Augmented Reality*", presenta un approccio orientato alla programmazione di robot basato su traiettorie in Realtà Aumentata (Quintero, Li, Pan, Chan, Van der Loos & Croft, 2018). Utilizzando l'HMD HoloLens e un braccio robotico 7-DOF, è stata progettata un'interfaccia robotica di Realtà Aumentata (AR) con quattro funzioni interattive per facilitare la programmazione del robot:

1. specifica della traiettoria;
2. anteprime virtuali del movimento del robot;
3. visualizzazione dei parametri del robot;

4. riprogrammazione in linea durante la simulazione e l'esecuzione (Quintero et al. 2018).

L'articolo del 2020, scritto da Soh Khim Ong e colleghi, intitolato "*Augmented reality-assisted robot programming system for industrial applications*". presenta un sistema di programmazione robot attraverso l'utilizzo della Realtà Aumentata (ARRPS) che permette una programmazione robotica più rapida e intuitiva rispetto alle tecniche convenzionali. ARRPS consente agli utenti, con poche conoscenze sul tema, di programmare attività per un robot seriale. Il sistema trasforma la cella di lavoro di un manipolatore industriale in un ambiente AR circoscritto. Attraverso un'interfaccia utente AR e un puntatore palmare per l'interazione, gli utenti possono muoversi nella cella di lavoro per definire target 3D e percorsi da far eseguire al robot reale. I dati e gli algoritmi dei sensori sono utilizzati per la pianificazione del moto del robot, il rilevamento delle collisioni e la convalida del piano (Ong, Yew, Thanigaivel & Nee, 2020).

Capitolo 3

Tecnologie

Dopo aver delineato lo stato dell'arte e descritto gli ambiti applicativi della Realtà Mista e della robotica, nel presente capitolo di questo elaborato occorre approfondire il tema delle tecnologie. In particolare, si discuterà il tema della Realtà Mista, degli strumenti e dei programmi utilizzati al fine di illustrare in modo preciso tutti gli elementi fondamentali per la parte sperimentale della tesi.

3.1 Realtà Mista

Come precedentemente affermato, la Realtà Mista, o Mixed Reality, è la fusione tra mondo reale e virtuale, che unisce la Realtà Aumentata e Realtà Virtuale con la realtà fisica. La MR ha l'obiettivo di (KG Partners srl, 2021):

1. creare ambienti nuovi e visualizzazioni innovative in cui oggetti fisici e digitali coesistono;

2. progettare metodi al fine di interagire in tempo reale, sia con gli oggetti fisici, sia con quelli digitali;

3. coinvolgere diversi ambiti applicativi per ottimizzare i processi aziendali.

A partire da questa definizione, la MR è stata applicata a (Microsoft, 2021):

- input ambientale;
- audio spaziale;
- posizioni e posizionamento negli spazi reali e virtuali.

Al fine di usufruire al meglio di tale tecnologia, occorre però disporre di visori simili ad occhiali come *Magic Leap* o Microsoft HoloLens (Generali, 2020).

I visori sono fortemente caratterizzati dalla capacità di collocare contenuti digitali nel mondo fisico, in relazione alle caratteristiche tecniche e al *field of view* adoperabile (Macina, 2020).

Questi device presentano alcune componenti fondamentali:

- display per vedere l'ambiente fisico e gli ologrammi;
- sensori al fine di analizzare e tracciare il setting circostante e i movimenti in modo dettagliato;
- audio per cogliere la voce con un insieme di microfoni, posizionati in specifici punti, e consentire un effetto audio spaziale.

Nella pratica, quindi, la Realtà Mista consente di manipolare oggetti e ambienti sia reali, sia virtuali, attraverso l'utilizzo di tecnologie sensoriali e di imaging innovativi (Intel Corporation, 2021).

Con il passare dei decenni e il progresso tecnologico, si è sviluppata sempre più la relazione tra l'essere umano e la macchina, giungendo a definire l'HCI, ossia una disciplina conosciuta come interazione uomo-computer.

L'interazione sta beneficiando dei continui miglioramenti apportati ai sensori e all'elaborazione, creando così nuove opportunità di sviluppo. Questi sensori consentono di elaborare informazioni come la posizione di una persona o un oggetto nell'ambiente, delineare superfici e confini, cogliere illuminazione e movimenti (Microsoft, 2021).

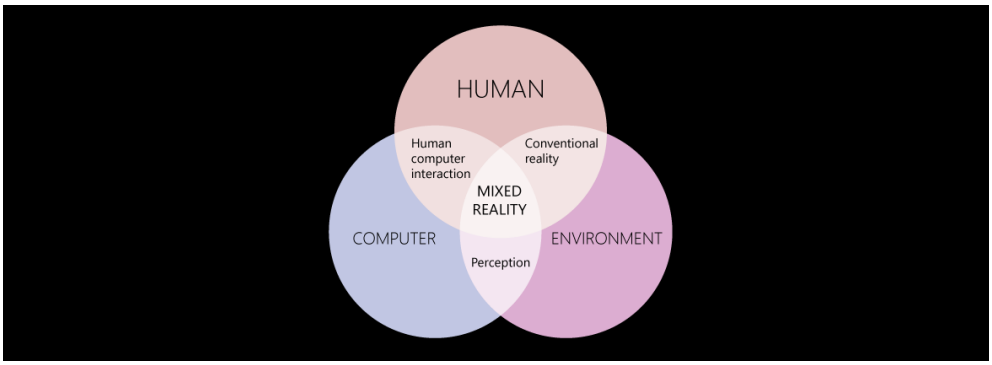


Figura 3.1: Interazione tra computer, persone e ambienti.

Fonte: Microsoft, 2021

È proprio a partire dalla congiunzione tra computer, input umano e ambiente che è possibile creare esperienze di Realtà Mista (*Figura 3.1*). Se il movimento fisico si converte in movimento digitale, al contempo i confini del mondo reale influenzano l'esperienza di gioco nel mondo digitale. Infatti, senza la combinazione di questi tre elementi l'esperienza virtuale verrebbe necessariamente meno (Microsoft, 2021).

Le esperienze di sovrapposizione di grafica sui flussi video del mondo fisico costituiscono la *Realtà Aumentata*; quelle che occludono la vista per presentare un'esperienza digitale costituiscono la *Realtà Virtuale*. Infine, le esperienze possibili tra la Realtà Aumentata e la Realtà Virtuale costituiscono la *Realtà Mista* (*Figura 3.2*). Ne sono un semplice esempio:

- un ologramma di un oggetto digitale
- un avatar di una persona
- muri e mobili digitali (Microsoft, 2021).

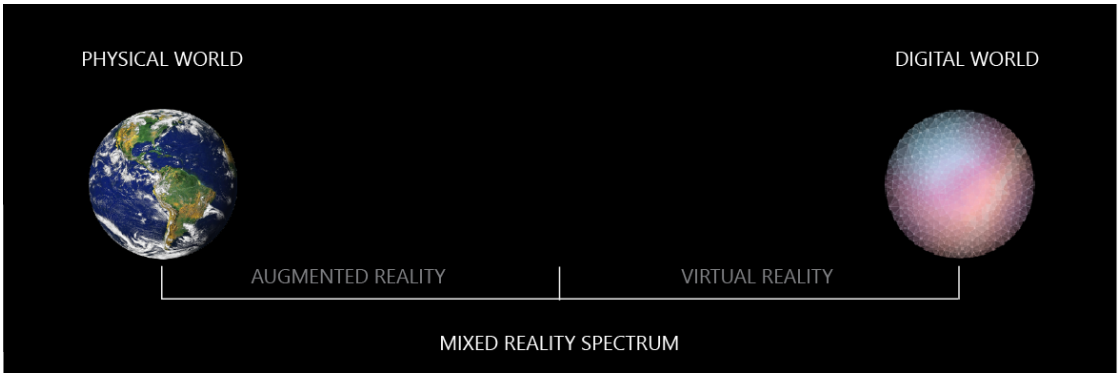


Figura 3.2: Spettro della Realtà Mista

Fonte: Microsoft, 2021

In particolare, le esperienze di Mixed Reality sono possibili mediante l'uso di due tipologie di dispositivi (Microsoft, 2021):

1. I dispositivi olografici sono caratterizzati dalla capacità di collocare contenuti digitali nel mondo reale come se vi fossero concretamente;

2. I dispositivi immersivi sono, invece, caratterizzati dalla possibilità di dare un senso di "presenza", celando il mondo fisico e sostituendolo con un'esperienza digitale.

Le differenze tra i due dispositivi sono riportate in maniera dettagliata nella Tabella 3.1.

DISPOSITIVI ED ESPERIENZE		
Caratteristica	**Dispositivi olografici**	**Dispositivi immersive**
Dispositivo di esempio	Microsoft HoloLens 	Samsung HMD Odyssey+ 
Schermo	Display trasparente. Permette alla persona di osservare l'ambiente fisico mentre indossa il visore VR.	Display opaco. Limita la vista dell'ambiente fisico mentre si indossa il visore VR.
Movement	Movimento di sei gradi di libertà, con rotazione e traslazione.	Movimento di sei gradi di libertà, con rotazione e traslazione.

Tabella 3.1: Dispositivi

Fonte: Microsoft, 2021

È ormai evidente che il progresso tecnologico consenta di esperire la Realtà Mista ma, è pur vero, che ancora non sono disponibili strumenti capaci di offrire esperienze per l'intero spettro

di tale realtà. Al momento, per esempio, Windows 10 offre una piattaforma di Realtà Mista che accomuna produttori e sviluppatori; i dispositivi in uso, inoltre, supportano una precisa gamma in tale spettro ma, in futuro, si prevedono dispositivi olografici ulteriormente immersivi.

L'esperienza ideata dallo sviluppatore indica un punto preciso nello spettro della Realtà Mista. In particolare, quella basata sulla realtà fisica è eseguita in maniera ottimale attraverso l'HoloLens.

Come è possibile vedere dalla *Figura 3.3*, sul lato sinistro, in prossimità della realtà fisica, gli utenti rimangono ancorati al mondo reale; al centro dello spettro della Realtà Mista, si mescolano insieme le esperienze reali e digitali; a destra, invece, in prossimità della realtà digitale, è possibile sperimentare un ambiente totalmente digitale perdendo la consapevolezza di ciò che realmente accade nel mondo fisico (Microsoft, 2021).

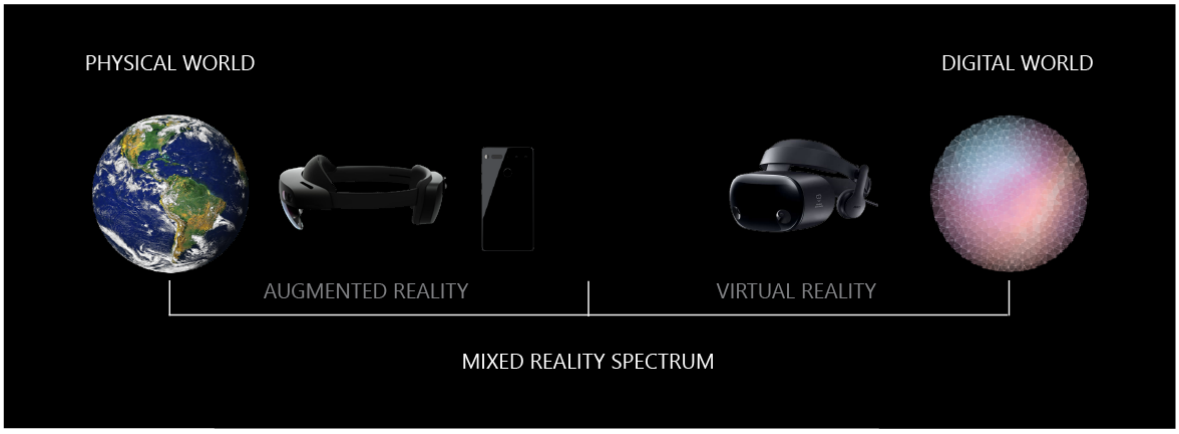


Figura 3.3: Posizione occupata dai dispositivi nello spettro della Realtà Mista

Fonte: Microsoft, 2021

In un'ottica a lungo termine, questa tecnologia può rappresentare la successiva evoluzione dell'interazione tra uomo, computer e ambiente, aprendo la strada a possibilità finora immaginate e realizzate solo in film e serie tv fantascientifiche (ad esempio, il film Jumanji) (Microsoft, 2021).

Come già affermato, la Realtà Mista permette di manipolare oggetti e ambienti al fine di applicare un filtro alla realtà fisica e integrarla con quella virtuale per consentire un'esperienza innovativa e immersiva (Siciliano, 2018).

3.2 Strumenti: HoloLens 2 (HMD)

L'HMD HoloLens 2 è un computer olografico autonomo di Realtà Mista che migliora le caratteristiche di elaborazione iniziate con l'HoloLens di prima generazione. Offre soluzioni innovative del settore che permettono all'utente un'esperienza coinvolgente, associata all'affidabilità, alla sicurezza e alla scalabilità sia dei servizi cloud, sia dell'intelligenza artificiale garantita dai prodotti Microsoft (Microsoft, 2021).

Esistono diverse edizioni dell'HoloLens 2, ognuna con le proprie caratteristiche e ambiti di applicazione:

- HoloLens 2 (solo dispositivo) è la versione base del dispositivo;

- HoloLens 2 con *Dynamics 365 Remote Assist* è pensato per l'ambito commerciale, al fine di incrementare la produttività e l'innovazione dell'azienda;

- HoloLens 2 *Industrial Edition* è ideato e testato per supporto agli ambienti delle camere bianche, dette anche laboratorio pulito, con standard dalla classe ISO 6 alla ISO 8. HoloLens 2 Industrial Edition ha la certificazione UL Classe I, Divisione 2, una garanzia di 2 anni e un programma di sostituzione veloce;

- *Trimble XR10* con HoloLens 2 è nato appositamente per i lavoratori in contesti controllati in cui è importante monitorare livelli di sporcizia, rumore e sicurezza. Inoltre, supporta un'innovativa tecnologia di elaborazione spaziale, pensata per aumentare la sicurezza sul lavoro, con una soluzione certificata da usare con i caschi rigidi;

- HoloLens 2 *Development Edition*, infine, è utile a creare soluzioni di *mixed reality*. Quest'offerta integra HoloLens 2 con versioni di valutazione di Unity3D e crediti Azure per i servizi cloud.

La tecnologia HMD HoloLens 2 si costituisce, come qualsiasi strumento in ambito IT, di due parti fondamentali: il software e l'hardware.

Figura 3.4: Applicazione HoloLens 2

Fonte: GlobalData Technology, 2019.

HoloLens 2, quindi, offre una composizione rivoluzionaria di software, hardware, Realtà Mista e intelligenza artificiale (AI) per condurre la produttività umana a livello mai raggiunti prima d'ora (Insight Technology Solutions S.R.L., 2021).

3.2.1 Software

In particolare, il software dell'HoloLens 2 è dotato di (Microsoft, 2021):

- un sistema operativo *Windows Holographic*, che consente agli utenti di utilizzare applicazioni e giochi in ambiente di Realtà Mista;
- un Visualizzatore 3D per la percezione visiva di modelli e animazioni 3D in tempo reale;
- Cortana come assistente vocale per la produttività personale;
- Dynamics 365 Remote Assist, una feature che permette ai tecnici di collaborare e risolvere problemi in modalità remota con altre figure professionali;
- Microsoft Edge, come web browser e OneDrive, per accedere e modificare file da tutti i dispositivi dell'utente.

Inoltre, l'HoloLens 2 supporta il riconoscimento del fattore umano tramite degli algoritmi che consentono: il tracciamento della mano mediante un modello per la manipolazione diretta, il tracciamento oculare per il rilevamento in tempo reale della posizione degli occhi e la chiamata

vocale (Microsoft, 2021). Oltre a questo, l'HMD è in grado di ricevere informazioni sull'ambiente circostante utilizzando: sei gradi di libertà (6 DOF) per il rilevamento posizionale su scala globale; mapping spaziale per creare delle *mesh* dell'ambiente in tempo reale e l'acquisizione in Realtà Mista (*Mixed Reality Capture*) per sovrapporre foto o video dell'ologramma misto e dell'ambiente fisico.

Grazie alle API fornite e alla strumentazione tecnica degli HoloLens, è possibile manipolare gli ologrammi all'interno dello spazio virtuale. Quando un ologramma è in prossimità dell'utente, sulla punta dell'indice sarà visualizzato un cursore di tocco a forma circolare. Questo permette di interagire con gli ologrammi con precisione ed estrema facilità: un elemento può essere selezionato attraverso il tocco in modo simile al touchscreen. Inoltre, ogni oggetto, oltre a essere selezionato, può essere afferrato con la medesima facilità. Infatti, è sufficiente avvicinare il pollice e l'indice all'ologramma e tenere premuto: l'oggetto afferrato può essere spostato, ridimensionato, ruotato e rilasciato tramite l'apertura delle dita. Nel caso in cui non fossero presenti ologrammi nelle immediate vicinanze dell'utente, il cursore tocco non sarà più visualizzato e sarà automaticamente sostituito da due raggi, uno per mano, che permettono di interagire con gli oggetti a distanza.

Per selezionare un oggetto distante, invece, è necessario usare il raggio della mano per puntare l'ologramma desiderato, posizionare il dito indice verso l'alto ed eseguire uno specifico movimento, ovvero avvicinare il pollice e l'indice e rilasciarli velocemente. Similmente, per afferrare un oggetto lontano, sarà necessario ripetere la stessa sequenza di operazioni ma senza rilasciare le dita (Microsoft, 2021).

Ricapitolando, utilizzando il tocco dell'aria è possibile:

- scorrere. Per scorrere il contenuto di una finestra bisogna toccare e tenere premuto il contenuto, spostare il raggio della mano verso l'alto o il basso, verso destra o sinistra;

- afferrare. Per afferrare una finestra o un ologramma bisogna selezionare la barra del titolo o l'ologramma stesso con il raggio della mano, premere e tenere premuto.

- aprire i menu di scelta rapida. Per aprire i menu di scelta rapida bisogna toccare e tenere premuto il raggio dalla mano (Microsoft, 2021).

3.2.2 Hardware

L'hardware dell'HoloLens di seconda generazione, mostrato in *Figura 3.5*, è innovativo e completo. Non necessità di collegamenti e possiede un ampio display ad alta risoluzione. Tutto ciò per offrire all'utente un'immagine completa dell'ambiente circostante (Microsoft, 2021).

In particolare, si possono distinguere diversi componenti:

- visiera: contiene i sensori e i display dell'HMD;
- fascia: componente pratico, funzionale alla comodità e alla sicurezza durante l'uso del dispositivo;
- pulsanti di luminosità: consentono di regolare la luminosità dello schermo;
- pulsanti del volume: consentono di regolare il volume del dispositivo;
- pulsante di alimentazione: consente l'accensione e lo spegnimento del dispositivo;
- porta usb-c: posizionata sotto il pulsante di alimentazione, permette di caricare il dispositivo e di collegarlo a un PC (Microsoft, 2021).

Il dispositivo presenta una HPU (*Holographic Processing Unit*) personalizzata di seconda generazione che consente l'utilizzo di algoritmi di *computer vision* in tempo reale e, allo stesso tempo, un basso consumo energetico. Sul visore sono installate quattro camere in scala di grigio, due camere di tracciamento degli occhi, una di profondità e una IMU (*Inertial Measurement Unit*) che incorpora un accelerometro, un giroscopio e un magnetometro. L'audio è catturato da un array del microfono con cinque canali e gli altoparlanti sono incorporati nel dispositivo (Ungureanu et al.. 2020; Microsoft, 2021). Il dispositivo ha un peso di circa 570g, ha un sistema di raffreddamento passivo (senza ventole) ed è alimentato con batterie al litio con un'autonomia di 2-3 ore (Microsoft, 2021).

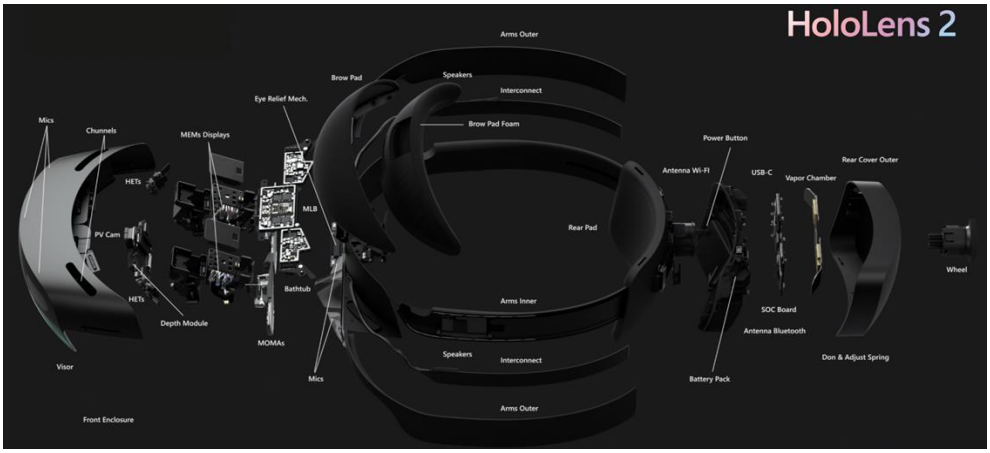


Figura 3.5: Hardware HoloLens 2

Fonte: Microsoft, 2021

3.3 Programmi: Unity3D e MoveIt!

Per l'attuazione del progetto sono stati scelti e utilizzati principalmente due programmi: Unity3D per la parte relativa alla simulazione della traiettoria del robot e MoveIt! per la pianificazione del moto e per il calcolo della cinematica inversa.

Il codice utilizzato per gli script di Unity3D è stato sviluppato e testato tramite il software Visual Studio 2019.

3.3.1 Unity3D

Unity3D è un game engine e un ambiente di sviluppo integrato *(Integrated development environment*, IDE) di proprietà della Unity Technologies Company per la creazione di media interattivi, in particolare videogame. Inoltre, è la tecnologia che esegue la fisica, la grafica, le interazioni, l'audio e il networking (Haas, 2021).

La prima versione del software (Unity 1.0.0) è stata messa a punto da David Helgason, Joachim Ante e Nicholas Francis in Danimarca e lanciata sul mercato nel giugno 2005 con lo scopo di creare un game engine a buon mercato ma che, allo stesso tempo, implementasse tool professionali in modo da "democratizzare" l'industria dello sviluppo di giochi.

Nella sua prima versione, Unity3D era disponibile solo per Mac OS X e gli sviluppatori avevano la possibilità di creare contenuti solo per un numero limitato di piattaforme. Oggi, a differenza del passato, è possibile utilizzare Unity3D su diversi sistemi operativi come Windows e Linux e, al contempo, sono aumentate le piattaforme su cui è possibile distribuire le applicazioni sviluppate tramite il game engine, per esempio: PC, PS4, XBOX 360, iOS, Windows Phone, Android. Oltre a queste è possibile sviluppare applicazioni per piattaforme simili come Web Player e Google Native per Internet.

Unity3D dà la possibilità agli sviluppatori software di programmare in tre differenti linguaggi all'interno dell'IDE: Unity JavasScript (conosciuto anche come UnityScript), C# e Boo. In teoria, il software dovrebbe permettere agli sviluppatori di implementare i propri algoritmi attraverso uno dei tre linguaggi a seconda delle preferenze del singolo. Questo nella realtà dei fatti non accade sempre in quanto i linguaggi sono eseguiti su Mono, una versione open-source di .NET che consente la *cross compatibility* su differenti piattaforme, e hanno diversi tempi di compilazione (Haas, 2021).

Di seguito, sono riportate le caratteristiche dei diversi linguaggi:

- UnityScript è un linguaggio di programmazione Javascript-like ed è il linguaggio maggiormente utilizzato dagli sviluppatori. È semplice da imparare e di facile digitazione,

implementa molti tipi di casting e permette all'utilizzatore di scegliere tra una tipizzazione dinamica (*dynamic typing*) e una tipizzazione forte (*strict typing*). Per via di questa tipizzazione, UnityScript è meno efficiente di C# in quanto il compilatore deve tener conto che il tipo degli oggetti può cambiare nel tempo.

- C# è un linguaggio più complesso di UnityScript ma, grazie alla forte tipizzazione, permette ai programmatori di avere un controllo completo sul progetto.

- Boo è un linguaggio di programmazione Python-like e ha una struttura simile a UnityScript. È usato da pochi sviluppatori e questo rende difficile il suo utilizzo, in quanto è complesso trovare assistenza in caso di errori o problemi (Haas, 2021).

Vuforia

Vuforia è un SDK per *devices mobile* che permette di creare applicazioni in Realtà Aumentata. Ideato da Qualcomm nel 2006 e poi acquistato da PTC nel novembre 2015, utilizza la camera per riconoscere e tracciare immagini e oggetti 3D in tempo reale. Attraverso il *benchmark test*, è stato dimostrato che l'algoritmo di tracciamento utilizzato da Vuforia è il più veloce e possiede un'ottima stabilità dell'immagine (Chapagain, 2018).

Vuforia ha un'interfaccia utente semplice e le sue funzionalità sono di facile utilizzo anche per i principianti; questo, insieme alla sua politica open-source, consente di avere una community di utenti assai variegata.

L'SDK offre una limpida interazione tra utente e ambiente reale; infatti, con l'utilizzo dei componenti dell'ARCamera, fornisce feedback visivi dell'ambiente circostante con le coordinate tracciabili sia dell'ambiente reale, sia dell'oggetto tracciato. L'ARCamera traccia, inoltre, l'orientamento dei dispositivi e degli oggetti in modo preciso così da semplificare e rendere più godibile l'esperienza della Realtà Aumentata.

Vuforia fornisce, poi, API per linguaggi di programmazione Java, C++ e .NET attraverso l'estensione del motore Unity3D. Grazie a quest'ultimo, è possibile utilizzare Vuforia per lo sviluppo di giochi e applicazioni 3D e 2D sia per piattaforme iOS, sia Android (Chapagain, 2018).

Include diversi componenti che permettono di rendere fluide e realistiche le applicazioni sviluppate. Tra questi componenti è possibile individuare:

- ARCamera: l'iniziale approccio verso lo sviluppo moderno consisteva nella portabilità dei dispositivi, oltre alle loro prestazioni e capacità. Il primo prototipo di questo metodo,

basato sulla Realtà Aumentata, risale al 1969: si utilizzava la lente dell'occhio per la Realtà Aumentata ma, grazie al recente sviluppo della tecnologia della fotocamera, questa può essere utilizzata per il tracciamento degli oggetti e, allo stesso modo, fornire le coordinate in tempo reale degli stessi. Similmente, l'ARCamera usa il *live feed* dell'ambiente attraverso la fotocamera del dispositivo e calcola valori in tempo reale per il toolkit che saranno successivamente utilizzati per le elaborazioni delle immagini.

- Le immagini target sono una parte essenziale di Vuforia e sono necessarie per il suo funzionamento. Rappresentano, inoltre, le immagini che Vuforia è in grado di rilevare e tracciare. Il motore grafico rileva e tiene traccia dell'immagine stessa, confrontando le caratteristiche naturali estratte dall'immagine dell'ARCamera con un database di target noto. Quando il target è inquadrato dalla camera, i contenuti AR compaiono sulla scena, si sovrappongono all'immagine e si dispongono su tutto lo spazio disponibile. Le estensioni supportate per le immagini target sono JPEG e PNG in RGB o in scala di grigi (Chapagain, 2018; PTC Inc., 2021).

3.3.2 MoveIt!

MoveIt! è un software open source per ROS (*Robot Operating System*), anch'esso un software utilizzato per la manipolazione robotica. MoveIt! include molteplici tool per velocizzare il lavoro dei bracci robotici e si basa sul riutilizzo del codice per evitare agli sviluppatori l'onere di dover ogni volta reinventare la ruota.

Al giorno d'oggi, MoveIt! sta diventando uno standard nel mondo della robotica mobile, in quanto questa tecnologia è utilizzata da oltre cento robot (Robotnik, 2014; Sucan & Chitta, 2021).

Il software permette di costruire la scena di pianificazione utilizzando mesh, precedentemente progettate con programmi CAD, e consente al robot di interagire con esse. Tramite MoveIt! è possibile pianificare il moto del manipolatore verso qualsiasi obiettivo target nello spazio di lavoro tenendo conto dell'ambiente, degli ostacoli presenti e interagendo con loro includendoli nel processo di pianificazione: ogni *Collision Object* può essere agganciato al braccio robotico desiderato e, così facendo, l'oggetto si muoverà insieme al robot (Robotnik, 2014).

Queste funzionalità aiutano a determinare fin da principio se il manipolatore è in grado di raggiungere determinati oggetti situati in diversi punti dello spazio e, indirettamente, aiutano lo sviluppatore a capire, all'interno dello spazio reale, in che posizione il robot risulti maggiormente performante.

Grazie alle molte *utility* che MoveIt! mette a disposizione, è possibile eseguire il movimento del robot fino alla posa desiderata con elevata flessibilità; quindi, è consentito modificare l'algoritmo di pianificazione del moto in base alle proprie esigenze al fine di ottenere le migliori prestazioni. Questo risulta utile nel caso in cui il movimento del braccio robotico debba soddisfare criteri specifici come il movimento in spazi ristretti o in ambiente condiviso con esseri umani (Robotnik, 2014).

Il software MoveIt! consente di pianificare il movimento verso un target sia nello spazio cartesiano, sia nello spazio dei giunti:

- la pianificazione nello spazio cartesiano si utilizza quando si desidera far seguire all'*end-effector* una traiettoria precisa.

- la pianificazione nello spazio dei giunti si utilizza per consentire al robot di seguire traiettorie più naturali, come lo spostamento tra due punti senza vincoli di orientamento.

Le applicazioni più avanzate del software includono l'integrazione di sensori 3D per il riconoscimento di oggetti per le attività di pick and place o l'uso di algoritmi di deep learning (Robotnik, 2014).

In conclusione, MoveIt! serve a facilitare il lavoro con bracci e manipolatori robotici mobili, consentendo agli utenti di implementare le funzionalità di alto livello in fase di sviluppo senza preoccuparsi di alcuni calcoli complessi già risolti nativamente dal software (Robotnik, 2014; Sucan & Chitta, 2021).

3.3.3 Visual Studio

Microsoft Visual Studio è un ambiente di sviluppo multi-linguaggio integrato sviluppato da Microsoft Corporation. Il software supporta la creazione di progetti per varie piattaforme, tra cui Mobile e Console. Inoltre, consente di creare e utilizzare estensioni e componenti aggiuntivi (Naini, 2019; Microsoft, 2021).

Dalla nascita della piattaforma .NET, tramite Visual Studio è possibile programmare in diversi linguaggi di programmazione tra cui: C#, C++, Visual Basic .Net. Nonostante non sia compatibile con il linguaggio Java, nelle precedenti edizioni, l'IDE rendeva disponile il supporto per il linguaggio J#.

Visual Studio integra nativamente *IntelliSense*, tecnologia che consente la correzione di eventuali errori sintattici e, in alcuni casi, logici, senza la necessità di compilare l'applicazione. Ha, al suo interno, un debugger per la correzione o il rilevamento degli errori in *runtime* e mette a disposizione strumenti per analizzare le prestazioni dei programmi. Inoltre, si integra con

l'ambiente di sviluppo *Team Foundation Server* per effettuare il versioning sul codice (Naini, 2019; Microsoft, 2021).

L'IDE supporta diversi template per ogni linguaggio di programmazione, come Applicazione desktop, servizio di Windows, libreria di classi e alcuni sottomenù per indirizzarsi sulla piattaforma su cui si vuole sviluppare, tra cui: Windows Store, Microsoft Azure, Android e iOS.

Dalla versione 2015, Visual studio si è enormemente ingrandito fino a raggiungere una dimensione massima di circa 80 GB per l'installazione completa. Questo è stato possibile grazie all'introduzione di diverse funzioni come l'integrazione di Unity3D per lo sviluppo dei videogiochi, la possibilità di modificare e gestire i cursori, le icone e il testo dell'applicazione e il supporto per gli strumenti nativi di Python e applicazioni Linux.

Infine, l'IDE consente di installare dei template e dei componenti aggiuntivi di terze parti, come le estensioni per il supporto del linguaggio PHP (Naini, 2019; Microsoft, 2021).

3.4 Manipolatore industriale: e.Do

Il manipolatore e.Do, raffigurato nella *Figura 3.6*, è un robot modulare, articolato multiasse con intelligenza integrata. Realizzato dall'azienda Comau per rendere interattivo e divertente l'apprendimento, è composto da sei assi motorizzati e sostiene un payload massimo di 1 kg. L'e.Do è un manipolatore robotico in grado di soddisfare le esigenze di diverse categorie: dagli insegnanti agli studenti, dalle scuole alle università, dagli appassionati ai novizi della robotica. Al suo interno utilizza la logica di controllo ROS e grazie a questo offre, ai suoi utilizzatori, la possibilità di creare ed eseguire i propri programmi e le proprie applicazioni. I bracci del manipolatore sono inseriti su una base esagonale contenente una scheda madre Raspberry Pi e il suo sistema operativo è basato su Linux. Gli utenti con più esperienza nel campo della robotica hanno la possibilità di entrare nel sistema di controllo integrato e modificarlo; questo fornisce al robot la flessibilità e l'intelligenza per realizzare traiettorie complesse e automatizzare i processi nel mondo reale (Comau SpA, 2021).

Le caratteristiche principali del manipolatore e.Do sono:

- robot a sei assi rotoidali con architettura hardware e software open source;
- struttura modulare e flessibile per consentire la configurazione personalizzata;
- unità di movimento con motori a corrente continua, rivestimenti in plastica composita, logica di controllo e memoria integrate.

Grazie all'integrazione delle librerie ROS, gli sviluppatori hanno la possibilità di programmare le loro applicazioni per l'e.Do utilizzando diversi linguaggi di programmazione tra cui: Python, C, C++, Java. Inoltre, è possibile fare interagire il robot con qualunque software o hardware compatibile con ROS (Comau SpA, 2021).

Secondo la Direttiva Macchine 2006/42/CE, il manipolatore e.Do è considerato una macchina e, secondo la norma EN ISO 1348:2014, un "Robot per Personal Care", cioè può essere utilizzato in totale sicurezza da chi ha più di 14 anni (Comau SpA, 2021).

Figura 3.6: Robot e.Do

Fonte: Comau SpA, 2021

3.5 Linguaggi di programmazione

L'utilizzo dei software e, più in generale, della programmazione informatica è idealizzato come una funzione chiave, utile a coordinare e sottoporre le varie tipologie di codici e algoritmi ai bisogni specifici delle persone e delle società multibusiness, le quali ambiscono ad avere il pieno controllo del mercato, stando sempre al passo con i tempi e con la tecnologia (Advanced Knowledge & Technology, 2017).

Analizzando nel dettaglio la definizione, fornita di linguaggio di programmazione, si percepisce in maniera evidente la sua assoluta importanza nei termini di innovazione, cambiamento e competizione.

Non a caso i linguaggi di programmazione sono considerati dei veri e propri codici di lettura, che possono essere impiegati per la scrittura e la compilazione di programmi al computer come nel caso di applicazioni, *utility programs* e *system program*, i quali permettono all'utente di rompere gli schemi, nonché avere una visione diversa del futuro e della nostra società (Advanced Knowledge & Technology, 2017).

Un linguaggio di programmazione in informatica è un linguaggio formale che specifica un insieme di istruzioni che possono essere usate per produrre dati in uscita: esso è utilizzabile per il controllo del comportamento di una macchina formale o di un'implementazione di essa (tipicamente, un computer), ovvero in fase di programmazione di questa attraverso la scrittura del codice sorgente di un programma: un linguaggio di programmazione è considerato a tutti gli effetti tale se è Turing-completo.

Ogni linguaggio di programmazione ha delle caratteristiche specifiche, che lo rendono perfetto per risolvere determinate situazioni, ma inadatto per altre. Insomma, non esiste un linguaggio che possa essere il migliore, sempre, in ogni occasione (Gabbrielli & Martini, 2011).

3.5.1 C#

Il linguaggio di programmazione C# è semplice, moderno, indipendente dai tipi, orientato agli oggetti e ai componenti. Ha la sua origine dalla famiglia dei linguaggi C e C++ e combina la potenza espressiva di questi ultimi con l'elevata produttività di linguaggio di sviluppo rapido. Inoltre, consente ai programmatori di sviluppare diversi tipi di applicazioni affidabili e sicure all'interno del sistema .NET. Dal suo primo rilascio, nel 2000, C# continua ad aggiungere funzionalità in grado di supportare maggiori carichi di lavoro e procedure per i nuovi software.

"The C# Programming Language" è il riferimento tecnico per il linguaggio, scritto dall'architetto del linguaggio Anders Hejlsberg e dal team di progettazione dello stesso (Hejlsberg, Wiltamuth & Golde, 2006).

Recentemente C# è stato accettato come standard dall'International Organization for Standardization (ISO) e dall'ECMA. Questo ha reso la comprensione del linguaggio fondamentale per i programmatori informatici (Microsoft, 2021).

Grazie alle sue caratteristiche, C# permette di creare applicazioni solide e durevoli: utilizza il *Garbage Collector* per recuperare in modo automatico la memoria occupata da oggetti inutilizzati e non più raggiungibili, gestisce le eccezioni e fornisce un approccio strutturato per il rilevamento e il ripristino degli errori. Supporta la programmazione funzionale tramite l'utilizzo delle espressioni lambda e, grazie alla sintassi *Language Integrated Query* (LINQ), consente di creare un modello comune per l'interfacciamento con i database. Il linguaggio gestisce, inoltre, funzioni asincrone che lo rendono ottimo per la creazione di sistemi distribuiti.

Tutti i tipi definiti dal linguaggio, comprese le primitive come *int* e *float*, ereditano da un unico tipo *object* e condividono un set di applicazioni comuni. C# permette l'allocazione dinamica della memoria e l'archiviazione online di strutture semplici, supporta i tipi e i metodi generici e fornisce gli iteratori, utili per l'implementazione di classi Collection (Microsoft, 2021).

Riassumendo, le caratteristiche principali del linguaggio C# sono:

- moderno
- elegante
- semplice da imparare
- sicuro
- versatile e multi-ruolo
- veloce
- completo
- multipiattaforma
- in continua evoluzione
- dotato di un ottimo ambiente di sviluppo
- in buona parte Open Source.

C# necessita di tempo e impegno per essere appreso in modo adeguato, essendo un linguaggio utile e sofisticato, ma, avendo una sintassi simile ad altri linguaggi, non risulta di difficile comprensione. Un aiuto al suo apprendimento è dato dalla chiarezza e dall'organizzazione dei suoi framework, in aggiunta alla grande quantità di documentazione teorica e pratica reperibile dalla letteratura esistente.

Grazie alla versatilità del linguaggio e al gran numero di librerie messe a disposizione per lo sviluppatore, con C# è possibile sviluppare sia software indipendenti sia applicazioni web o mobile (Rossi, 2019).

3.5.2 ROS

Robot Operating System (ROS) è un software middleware sviluppato per la prima volta nei laboratori della Stanford University, nel 2007. In origine, non è stato pensato per scopi industriali ma, grazie alla sua semplicità d'utilizzo e al fatto di essere un software open-source, oggi è utilizzato sia all'interno di progetti amatoriali, sia da aziende leader del settore.

ROS è usato per le applicazioni di robotica, ha al suo interno un insieme di librerie utili per la programmazione e lo sviluppo di robot, in particolare robot di servizio, e offre tutte le funzioni di un sistema operativo come: controllo di dispositivi tramite driver, debug, comunicazione tra diversi processi, astrazione hardware e gestione delle applicazioni. I diversi processi attivi all'interno di ROS sono rappresentati come i nodi di un grafo e possono ricevere, inviare e multiplexare messaggi provenienti da e verso sensori, attuatori o altri nodi. ROS, nonostante l'importanza di avere una bassa latenza e un'elevata reattività per le operazioni di controllo di un robot, non è un sistema operativo real-time; tuttavia, è possibile ovviare a questo tramite l'integrazione del software con moduli real-time (AA. VV., 2014; Croccolino, 2019).

I software, all'interno dell'architettura ROS, possono essere divisi in tre differenti categorie:

- strumenti per lo sviluppo e per la pubblicazione di software basato su ROS;
- librerie per i processi ROS client come roscpp, rospy e roslisp;
- pacchetti con applicazioni e codice che utilizzano delle librerie per processi ROS client.

Le librerie e gli strumenti per lo sviluppo sono pubblicati sotto licenza BSD e sono indipendenti dal linguaggio di programmazione usato (Python, C++, LISP): questo li rende adatti sia per scopi commerciali, sia per scopi di ricerca, in quanto facenti parte del software open-source. Allo stesso modo, anche la maggior parte dei pacchetti è stato rilasciato con licenze open-source.

I pacchetti realizzano diverse funzioni di uso comune in ambito informatico, tra queste (AA. VV., 2014):

- driver;
- simulatori 2D e 3D;
- localizzazione;
- modelli 3D del robot;
- strumenti per l'utilizzo di mappe.

Le librerie per processi ROS client sono utilizzate maggiormente dai sistemi operativi Linux per via della loro dipendenza da altri software liberi. Al momento, lo sviluppo principale è portato avanti su Ubuntu e Debian, mentre altri sistemi come Gentoo, Fe.Dora e Arch Linux sono supportati in maniera sperimentale dalla comunità. Alcune librerie, come roslibjs o rosjava, possono supportare MATLAB e i maggiori web browser (AA. VV., 2014; Croccolino, 2019).

L'obiettivo principale di ROS è quello di realizzare le parti di basso livello dell'applicazione di controllo del robot, in modo che il programmatore possa concentrarsi sulle funzionalità più avanzate. In questo modo, i tempi di sviluppo sono notevolmente limitati e la quantità di bug ridotta il più possibile, grazie anche al continuo aggiornamento delle librerie da parte della grande comunità di sviluppatori che contribuiscono al progetto (AA. VV., 2021).

L'architettura interna di ROS prevede che il codice sia strutturato attraverso il concetto di nodo: ogni nodo gestisce la logica di funzionamento di una determinata parte del progetto e può comunicare con gli altri nodi tramite strumenti messi a disposizione dal framework stesso.

Una particolarità di ROS è l'essere *language agnostic*, cioè ciascuno dei nodi ROS può essere sviluppato con un linguaggio di programmazione diverso. Questo è possibile perché i tool di comunicazione usati per interconnessione dei nodi non dipendono da un linguaggio specifico. Attualmente, possono essere utilizzati i linguaggi Python, C++, Lisp, Java e Lua (AA. VV., 2021).

Al fine di rendere il tema chiaro e completo sarà ora descritto RosSharp che permette di coniugare i due argomenti precedentemente trattati: ROS e C#. Infatti, RosSharp è un insieme di librerie software open source sviluppate in C# che consentono la comunicazione tra ROS e le applicazioni .NET, soprattutto Unity3D. In particolare, il framework offre all'utente una vasta gamma di opzioni tra cui (Bischoff, 2021):

- comunicazione tra ROS e applicazioni windows;
- import del modello URDF del robot come gameobject in Unity3D;
- controllare un robot reale attraverso Unity3D;
- visualizzare lo stato del robot su Unity3D;
- effettuare simulazioni in Unity3D del proprio robot senza connettersi con le librerie ROS, con il solo utilizzo delle informazioni contenute nel file URDF;
- fare training di una rete neurale.

La versione ufficiale di ROS è stata resa disponibile esclusivamente per sistemi Linux, mentre Unity3D può essere utilizzato su diversi sistemi operativi. In virtù di questo, RosSharp è indispensabile per consentire lo sviluppo di applicazioni robotiche in ambienti non Linux, in quanto permette la connessione tra il game engine e l'interfaccia ROS.

Attraverso il framework, usando l'URDF-transfer, è possibile importare il modello del robot utilizzato su ROS all'interno di Unity3D. L'URDF file contiene la lista dei bracci del manipolatore e, per ognuno di essi, i rispettivi parametri fisici tra cui: lunghezza, peso, inerzia, *collision geometric*, frizione, etc (Bischoff, 2021).

Oltre a questo, RosSharp contiene un pacchetto chiamato *rosbridge client*, un insieme di script che garantisce la connessione tra ROS e Unity3D tramite l'utilizzo di un rosbridge sulla rete. In particolare, un rosbridge è un protocollo che fornisce un'interfaccia JSON a ROS e consente ai client di inviare JSON al fine di connettersi ai software ROS pubblicando e sottoscrivendo topic ROS. Tramite gli script inclusi in rosbridge client è possibile, inoltre, inviare e ricevere messaggi ROS come pose, odometry, joy.

Come precedentemente detto, ROS è disponibile solo per sistemi operativi Linux ed è quindi possibile che l'applicazione Unity3D sia sviluppata su un diverso sistema operativo. Per far sì che la comunicazione avvenga correttamente è necessario installare ROS su una macchina virtuale Linux e che, su questa macchina, sia in esecuzione il processo ROS master. Tutti i nodi presenti sul sistema Linux e gli script di Unity3D sono collegati a questo processo. Mentre l'applicazione Unity3D è in esecuzione, uno script di controllo è eseguito sulla macchina virtuale: lo script pubblicherà dei messaggi su un topic di controllo che saranno ricevuti e analizzati da Unity3D attraverso rosbridge e i corrispondenti script RosSharp. Questi messaggi sono gli stessi che saranno usati da Unity3D durante la simulazione di movimento del robot (Bischoff, 2021).

Unity3D e ROS, quindi, cooperano ma si distinguono per il sistema di riferimento adoperato: Unity3D utilizza un sistema di riferimento *left-handed*, mentre ROS usa un sistema *right-handed*.

In conclusione, RosSharp mette a disposizione dei tools che compiono in automatico la trasformazione da un sistema all'altro, semplificando lo svolgimento del procedimento stesso (Bischoff, 2021).

3.5.3 Python

Python è un linguaggio di programmazione moderno, di alto livello e general-purpose. È pensato per essere facilmente leggibile e facile da ricordare; infatti, Python, a differenza di altri linguaggi di programmazione, risulta avere un apprendimento molto rapido che consente agli utenti di produrre codice molto velocemente (Lubanovic, 2015; Melotti, 2016).

Il linguaggio trova origine nei primi anni Ottanta quando, al *National Research Institute for Mathematics and Computer Science* (CWI) di Amsterdam, è stato sviluppato ABC, un linguaggio molto potente diventato popolare nel mondo Unix, da un gruppo di ricercatori, tra cui Guido Van Rossum. Pochi anni dopo, quest'ultimo iniziò a lavorare a un miglioramento di ABC, creando così Python.

Nel 2000 fu rilasciata la versione 2.0 che, oltre a un miglioramento generale delle prestazioni, arricchì il linguaggio con la "*list comprehension*"; l'anno successivo, inoltre, con il rilascio della versione 2.1, è stata ridefinita la licenza come "*Python Software Foundation License*". Nel dicembre 2008 si assiste a una vera rivoluzione con il rilascio della versione 3.0 perché, nonostante la versione sia molto simile alla precedente, il linguaggio è stato semplificato e ha introdotto diversi cambiamenti (come, per esempio, le stringhe Unicode di default). Per questi cambiamenti, Python 3 e Python 2 non sono compatibili (Melotti, 2016).

Grazie alla sua grammatica molto sintetica, il linguaggio permette di scrivere in un numero ridotto di righe il codice che svolge la medesima funzione rispetto a un linguaggio statico. Diversi studi hanno dimostrato che i programmatori producono, in media, lo stesso numero di righe di codice per giorno indipendentemente dal linguaggio utilizzato; quindi, ridurre il numero di righe necessarie per eseguire un compito tende a raddoppiare la produttività (Lubanovic, 2015).

Alla luce di ciò, i punti di forza di Python sono qui elencati e descritti:

- È un linguaggio gratuito e utilizzabile senza restrizioni di copyright. Ha una comunità molto attiva ed è costantemente aggiornato e migliorato;

- È multi-paradigma, supporta sia la programmazione procedurale, sia la programmazione ad oggetti, sia la programmazione funzionale;

- È un linguaggio portatile sviluppato in ANSI C. Si può utilizzare su diverse piattaforme tra cui: Windows, Linux, Unix, Macintosh, DOS, sistemi Real Time, Android e iOS. Questo è possibile perché Python è un linguaggio interpretato;

- È potente e facile da utilizzare. La sintassi e le funzioni native sono consistenti, intuitive e facili da imparare;

- È ricco di librerie che consentono di svolgere svariati compiti, come l'interazione con il sistema operativo e il file system, oltre alle migliaia di librerie create e aggiornate dalla comunità;

- È performante. Nonostante sia un linguaggio interpretato, i programmi sono compilati in formato *bytecode* prima di essere eseguiti. Questo formato è compatto ed efficiente e garantisce prestazioni elevate;

- Come C#, Python utilizza il meccanismo di Garbage Collection per la gestione automatica della memoria;

- È integrabile con altri linguaggi di programmazione come C, Java e i linguaggi del framework .NET.

Inoltre, grazie alla sua duttilità e al suo grande numero di librerie disponibili gratuitamente, Python è utilizzato in svariati ambiti applicativi come (Melotti, 2016):

- Programmazione GUI: è possibile creare interfacce grafiche tramite diversi toolkit, tra cui: Tkinter, incluso nella standard library; PyQt/PySide che permettono di lavorare con il toolkit QT; PyGtk, basato sul toolkit GTK; wxPython, un'interfaccia per wxWidgets. I programmi che utilizzano queste librerie possono essere eseguiti su tutti i maggiori sistemi operativi.

- Sviluppo web: tramite Python è possibile fare sviluppo web sia ad alto, sia a basso livello. Sono messe a disposizione diverse librerie per realizzare siti e applicazioni web come: Django, framework molto popolare per sviluppare siti e webapp; Flask, una libreria per creare semplici siti internet; Web2Py per le applicazioni web;

- Accesso ai database: la standard library di Python supporta un'interfaccia per SQLite. Inoltre, è possibile installare moduli per l'interfacciamento con altri database come Oracle e MySQL;

- Realizzazione di videogiochi: tramite Pygame è possibile sviluppare videogiochi in modo semplice e intuitivo.

- Realizzare applicazioni scientifiche: Scipy fornisce numerosi tool per la matematica, l'ingegneria e le scienze (Melotti, 2016).

Inoltre, Python possiede delle librerie che facilitano l'utilizzo dei framework ROS e MoveIt! e, in particolare, l'integrazione con ROS è garantita dalla libreria rospy.

L'*application programming interface* (API) rende disponibile, per i programmatori, un semplice interfacciamento con i costrutti di ROS quali *topic*, *service* e *parametres*. Al contempo, il design di rospy favorisce la velocità di implementazione rispetto alle prestazioni runtime così che gli algoritmi possano essere velocemente testati all'interno di ROS (Open Source Robotics Foundation, 2020).

Nel presente progetto è stato, infatti, utilizzato un codice rospy per l'inizializzazione di un nodo ROS con il nome specificato come primo parametro della funzione. Ecco alcune righe di codice a scopo esemplificativo:

```
rospy.init_node('move_group_Python_interface_tutorial',
anonymous=True)
```

La libreria rospy consente però anche altre funzionalità: per esempio da un lato, permette di pubblicare, tramite un Publisher, dati sul nodo e, dall'altro, di riceverne utilizzando un Subscriber (Open Source Robotics Foundation, 2017).

```
pub = rospy.Publisher('chatter', String, queue_size=10)
rospy.Subscriber("chatter", String, callback)
```

Il primo codice inizializza un Publisher, passando come parametri il nome del topic su cui pubblicherà i dati, il tipo di dati pubblicati e il massimo numero di messaggi che possono essere in coda, in attesa di essere pubblicati. Il secondo codice, invece, inizializza un Subscriber, specificando nei parametri il nome del topic su cui sarà in ascolto, il tipo di dato che riceverà e la funzione di *callback* che sarà chiamata quando giungeranno nuovi dati (Open Source Robotics Foundation, 2020).

La libreria moveit_commander fornisce, invece, le funzionalità necessarie per l'utilizzo di MoveIt! in ambiente di sviluppo Python: diverse interfacce per il motion planning, calcolo della traiettoria cartesiana e strumenti per il pick and place (Sucan & Chitta, 2021).

L'utilizzo di questa libreria consente di interfacciarsi attraverso specifici passaggi:

```
moveit_commander.rospy_initialize(sys.argv)
robot = moveit_commander.RobotCommander()
scene = moveit_commander.PlanningSceneInterface()
group = moveit_commander.MoveGroupCommander(group_name)
```

Questa parte di codice è essenziale per l'istanziazione e per inizializzare dei componenti fondamentali al fine di consentire lo scambio di informazioni tra il nodo ROS e MoveIt!.

In particolare, permette l'utilizzo di moveit_commander, crea un'interfaccia esterna del robot con la funzione RobotCommander(), crea un'interfaccia del mondo esterno al robot con la funzione PlanningSceneInterface() e inizializza la variabile che contiene il gruppo dei giunti della catena cinematica del robot tramite la funzione MoveGroupCommander(group_name) che riceve, come parametro, il nome della catena cinematica stessa.

Il framework MoveIt!, poi, consente la pianificazione della traiettoria tramite l'uso della cinematica inversa, ovvero mediante un processo di determinazione dei parametri di posizione, velocità e accelerazione che soddisfino il raggiungimento della posa desiderata. Il codice sotto riportato è un esempio di tale processo (Sucan & Chitta, 2021):

```
waypoints = []
wpose = group.get_current_pose().pose
wpose.position.z -= scale * 0.1
wpose.position.y += scale * 0.2
waypoints.append(copy.deepcopy(wpose))
wpose.position.x += scale * 0.1
waypoints.append(copy.deepcopy(wpose))
wpose.position.y -= scale * 0.1
waypoints.append(copy.deepcopy(wpose))
(plan,fraction)=group.compute_cartesian_path(waypoints, 0.01,
0.0)
return plan, fraction
```

Nella variabile wpose è salvata la posa iniziale della punta operativa del robot; dopodiché sono apportate modifiche alla posizione di quest'ultima e salvate all'interno dell'array waypoints. La funzione *compute_cartesian_path*, applicata alla catena cinematica *group* e ricevente come parametri la serie di punti da raggiungere, la risoluzione dell'interpolazione (nel caso in esempio pari a 1 cm) e la jump_threshold, si occupa del calcolo effettivo della cinematica inversa (Sucan & Chitta, 2021).

Capitolo 4

Descrizione del sistema

Il progetto in esame è stato sviluppato utilizzando un'architettura Client-Server: il software installato sull'HMD HoloLens 2 costituisce il client, mentre il server è il sistema ROS che opera su una macchina virtuale Linux.

Il sistema ha due connessioni: la prima associa l'applicazione al server ROS, la seconda serve per l'interfacciamento con il robot reale. In particolare, la prima è necessaria per la fase di simulazione della traiettoria: il software pubblica i dati dei punti da raggiungere su un nodo ROS, in modo che il server possa calcolare, tramite il risolutore di cinematica inversa di MoveIt!, il percorso che il manipolatore virtuale deve seguire per raggiungere i punti indicati dall'utente. Una volta completato il calcolo, il server pubblica a sua volta i dati necessari per il movimento del robot dando la possibilità di pre-visualizzare il path compiuto dal manipolatore. La seconda, invece, permette l'invio dei dati precedentemente calcolati al robot reale.

Non è possibile effettuare tale procedura tramite un'unica connessione poiché, per motivi di sicurezza, la versione di ROS installata sul manipolatore e.Do non consente connessioni multiple al Rosbridge server.

4.1 Server Linux

Per questo progetto si è deciso di utilizzare come server, una macchina virtuale Linux con installato il sistema operativo Ubuntu 18.04 LTS e il software ROS Noetic Ninjemys.

Lo scopo del server è mettere a disposizione dell'applicazione, installata sugli HoloLens 2, tutti gli strumenti necessari a consentire la connessione, il passaggio dei dati e l'elaborazione di questi ultimi attraverso un motore di cinematica inversa.

La connessione avviene tramite un WebSocket ROS che deve essere avviato sulla macchina virtuale mediante il comando:

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

Quest'ultimo consente alla macchina di essere il nodo Master (Roscore) e pone il server in ascolto sull'indirizzo IP della macchina stessa sulla porta predefinita che, nel caso standard, è la 9090.

Dopo la creazione del WebSocket, occorre eseguire il motore di cinematica inversa tramite il software MoveIt!, avviato dal comando:

```
$ roslaunch edo_moveit demo.launch
```

edo_moveit è un package custom, in quanto non è compreso nella versione standard di MoveIt!. È stato generato a partire dal file URDF del manipolatore e.Do grazie all'utilizzo del Setup Assistant messo a disposizione dal tool.

Per ultimo, è necessario avviare uno script Python contenente la logica del server: la ricezione dei punti, l'elaborazione dei dati, il rilevamento di errori e la pubblicazione dei risultati della pianificazione del moto. Questo script si interfaccia direttamente con l'applicazione e rimane in ascolto sui vari topic necessari per la comunicazione. Lo script è avvitato tramite il comando:

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

Dove script_package è la cartella contente il Catkin Workspace in cui è incluso lo script.

Per inizializzare correttamente il server occorrono questi tre comandi: l'ordine di avvio prevede che i primi due siano intercambiabili, mentre il terzo deve necessariamente essere lanciato dopo l'avvio del motore di cinematica inversa MoveIt!.

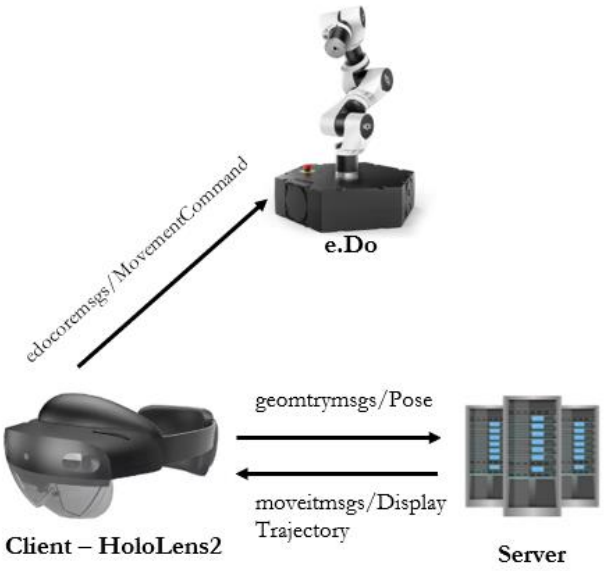


Figura 4.1: Architettura Client-Server

4.1.1 Logica del server

La logica del server è gestita mediante uno script Python, posizionato nel Catkin Workspace di MoveIt!. All'interno dello script, sono importate le librerie *rospy*, *moveit_commander*, *moveit_msg*, che sono indispensabili per il funzionamento del server e per l'interfacciamento con l'applicazione. In particolare, la libreria moveit_commander ha una classe chiamata MoveGroupInterface che contiene al suo interno le variabili necessarie al motore di cinematica inversa per funzionare correttamente:

```
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)
```

Questa parte di codice è necessaria per istanziare il server in sé. In particolare, le prime due righe inizializzano il MoveitCommander e il nodo rospy. La terza serve per creare una struttura che contiene tutte le informazioni del robot, tra cui lo stato del manipolatore e il suo modello cinematico. La quarta riga, invece, consente la creazione della scena, ovvero fornisce un'interfaccia al fine di ottenere, impostare e modificare la comprensione interna del manipolatore dell'ambiente circostante. Il group_name "edo" è una variabile definita dall'utente e deve combaciare con il nome della catena cinematica assegnata al manipolatore virtuale

durante la fase di setup su MoveIt!. La variabile è utilizzata dal costruttore del MoveGroupCommander, punto focale per la logica del server perché mediante tale classe si possono impartire i comandi di esecuzione al manipolatore.

Publisher e Subscriber

Come descritto nel capitolo precedente, ROS fa uso di nodi che, al loro interno, impiegano dei topic per comunicare tra loro. Ogni nodo può essere Publisher o Subscriber a seconda del ruolo che assume nella comunicazione. In particolare, il Publisher funge da mittente, per il tipo di dato associato al topic, mentre il Subscriber da ricevente.

Il server, per com'è concepito, ha tre Publisher e due Subscriber che saranno ora illustrati:

```
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)
```

Come si evince dal codice, tutti i Publisher pubblicano delle strutture DisplayTrajectory ma hanno degli scopi differenti: il primo pubblica i dati necessari per il calcolo della cinematica inversa e la visualizzazione del moto su MoveIt!; il secondo informa il client riguardo a eventuali errori nel calcolo della traiettoria; il terzo pubblica i dati processati dal motore di cinematica inversa utili al movimento del robot.

```
def listener(self):
   rospy.Subscriber("/geometry_msgs/Pose",Pose,self.callback)
   rospy.Subscriber("/geometry_msgs/FinalPose",Pose,self.call
    back2)
   rospy.spin()
```

Al pari dei Publisher, che operano sulla stessa tipologia di dati, i due Subscriber sono in ascolto per il medesimo tipo di informazione. Infatti, entrambi ricevono dei messaggi Pose. Il primo riceve la posa, quindi la posizione e l'orientamento 3D di ogni target inviato dal client, mentre al secondo sono inviati dei codici noti, incapsulati dentro strutture Pose, che ricoprono diverse funzioni tra cui:

- Informare il server che non saranno più ricevuti punti, quindi di iniziare il calcolo della cinematica inversa;
- Informare il server dell'arrivo di una nuova serie di punti. Questa funzione utile per cancellare la lista precedentemente calcolata;
- Riportare il robot virtuale alla posizione di partenza per iniziare un nuovo percorso o per settare la posizione di partenza all'accensione dell'applicazione.

4.2 Interfaccia grafica

L'applicazione è stata sviluppata tramite il game engine Unity3D ed è suddivisa in due scene distinte. Nella prima l'utente inserisce gli indirizzi IP e le porte su cui sono in ascolto i robot, reale e virtuale; la seconda è incentrata sull'interazione tra l'utente e i manipolatori: è possibile inserire, modificare ed eliminare i target e sono presenti diverse utility che permettono di visualizzare l'anteprima della traiettoria seguita dal robot.

4.2.1 Prima scena

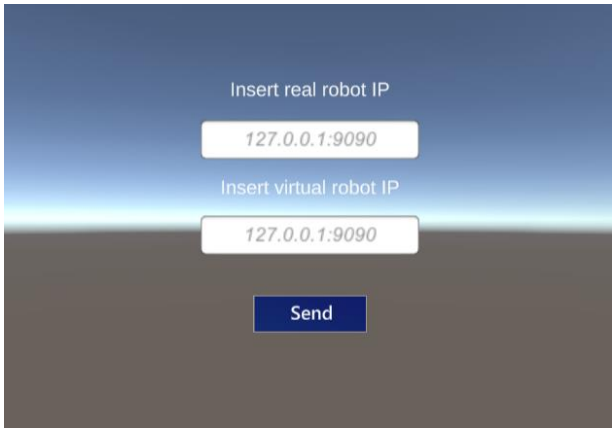


Figura 4.2: Prima scena dell'applicazione

La prima scena, quella di avvio, è caricata all'apertura dell'applicazione e comprende due *InputField* e un *PressableButton.*

L'utente, cliccando su uno dei due InputField, può inserire l'indirizzo IP e la porta su cui sono in ascolto il robot reale e il robot virtuale. Una volta inseriti i dati e cliccato sul *Button* "Send", l'applicazione controlla la correttezza e la validità dei dati inseriti. Se il controllo da esito positivo, è caricata la seconda scena.

4.2.2 Seconda scena

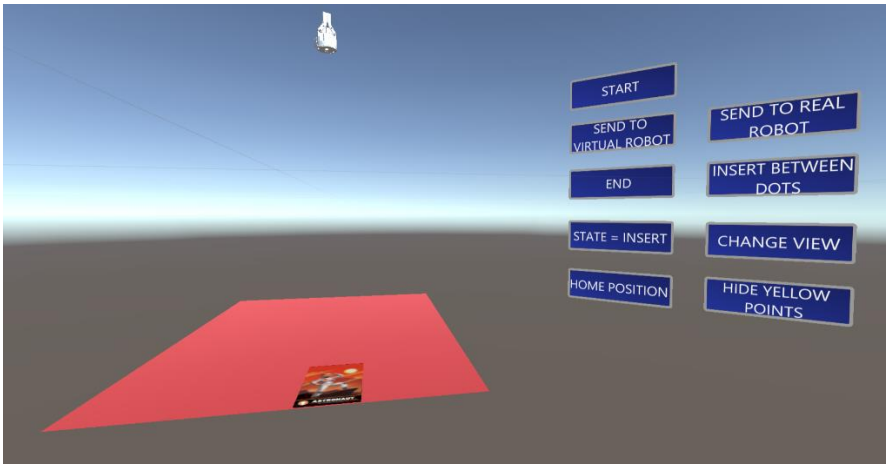


Figura 4.3: Seconda scena dell'applicazione

La seconda scena è composta dalla pinza del manipolatore virtuale, dal suo workspace, dal target Vuforia e dalla *UI*.

All'avvio dell'applicazione, l'utente visionerà unicamente il tasto "Start" e non gli sarà possibile inserire dei target all'interno della scena. Una volta cliccato il Button, saranno abilitate tre funzionalità di interazione con il manipolatore: "End", "State" e "Home Position". Allo stesso tempo, il Button Start è nascosto all'utente.

Inserendo il primo punto, saranno visualizzati i Button "Send to Virtual Robot" e "Change View"; mentre se sono presenti due o più target, comparirà il Button "Insert between dots". Se l'utente avvia la simulazione della traiettoria, premendo il Button "Send to Virtual Robot", e non ci sono errori di calcolo, il manipolatore virtuale compie il movimento calcolato dal server; similmente saranno abilitati i Button "Send to Real Robot" e "Hide Yellow Point".

Si delinea, mediante un elenco puntato, lo scopo di ogni Button.

- Start: abilita l'inserimento dei target all'interno del workspace;
- End: elimina tutti i target presenti sulla scena, sia del robot, sia dell'utente, e riavvia l'applicazione;
- State: permette all'applicazione di essere una macchina a stati; all'avvio della scena il valore è su Insert di default e permette l'inserimento dei punti all'interno dello spazio di lavoro; Cancel consente la cancellazione dei target e Modify la modifica della loro posizione e del loro orientamento;
- Home Position: consente di posizionare il robot reale e il robot virtuale alla posizione di partenza;
- Insert between dots: permette di scegliere un punto presente sulla scena e inserire un secondo punto prima o dopo di esso;
- Change View: consente la scelta del GameObject (pinza, cilindro oppure nulla) utile a pre-visualizzare l'orientamento del target;
- Hide/Show Yellow Point: permette di nascondere o mostrare i punti gialli istanziati dal robot virtuale durante la simulazione del percorso;
- Send to Virtual Robot: consente di pubblicare su un topic ROS le informazioni sui target inseriti dall'utente;
- Send to Real Robot: permette di pubblicare su un topic ROS, collegato direttamente al robot reale, i dati necessari al movimento.

4.3 Applicazione HoloLens 2

Per poter usufruire correttamente dell'applicazione è necessaria una prima fase di setup in cui è necessario configurare il server in modo da rendere possibile la connessione e il passaggio dei dati. Successivamente, è necessario che l'applicazione e il robot reale siano connessi sulla medesima rete.

4.3.1 Setup

Come precedentemente descritto, durante la fase di setup, è stata installata una macchina virtuale con sistema operativo Ubuntu 18.04. All'interno dell'ambiente Linux è stato creato un workspace su cui sono stati aggiunti due elementi basilari: da un lato, il framework ROS per la comunicazione con il robot e, dall'altro, il software MoveIt! per calcolare la cinematica inversa del manipolatore attraverso il modello del robot e.Do precedentemente importato.

Lato Unity3D, è stato necessario installare il package *Mixed Reality Toolkit* (MRTK) per simulare le funzionalità degli HoloLens 2, e le librerie ROS# per permettere la comunicazione tra Unity

e la macchina virtuale. Dopodiché è stato importato il modello del manipolatore tramite il file URDF, modificando infine alcuni settaggi per rendere possibile il movimento del robot all'interno dell'applicazione. Infine, è stato installato il package Vuforia per il supporto alla videocamera dell'HMD.

4.3.2 Connessione

La connessione tra client e server avviene a cavallo tra le due scene: nella prima scena, i dati richiesti all'utente sono gli indirizzi IPv4 del robot reale, del robot virtuale e delle rispettive porte su cui sono in ascolto. Sui dati inseriti sono svolti dei controlli per la validazione degli indirizzi IP e per la correttezza delle porte. Superati i controlli, i dati sono salvati in un *GameObject* di Unity3D e inoltrati alla seconda scena, dove saranno utilizzati per la connessione tramite WebSocket alla macchina virtuale. Le connessioni necessarie per far funzionare l'applicazione sono due: la prima serve per la comunicazione tra l'applicazione e il server, mentre la seconda è diretta verso il robot reale ed è indispensabile per consentire l'invio dei punti al manipolatore.

Nella seconda scena dell'applicazione è necessario che la macchina virtuale e Unity3D possano comunicare, scambiare dati tramite dei messaggi e, successivamente, questi dati siano inoltrati al robot reale. Per fare in modo che questo accada, si attivano due istanze dello script *RosConnector* della libreria ROS# che si connettono, utilizzando i dati passati dalla prima scena, uno al robot virtuale e uno al robot reale.

Se la connessione va a buon fine, l'utente ha la possibilità eseguire diversi compiti:

- inserire un nuovo punto;
- inserire un nuovo punto tra due preesistenti;
- visualizzare l'anteprima dell'orientamento dell'end-effector in ogni punto inserito;
- eliminare un punto;
- spostare un punto precedentemente inserito;
- inviare al robot virtuale la lista dei punti;
- inviare al robot reale la lista dei punti.

Quando l'utente termina la parte di inserimento punti e conferma la sua scelta, inizia un processo che invia, uno per volta, tutti i punti selezionati nell'ordine in cui sono stati salvati nella lista. Ogni punto è incapsulato all'interno di un messaggio Pose e pubblicato su un *Topic* di cui l'applicazione Unity3D funge da *Publisher* e uno script sulla macchina virtuale da *Subscriber*.

Il messaggio *Pose* è elaborato dallo script e, se il punto rientra nello spazio di lavoro del manipolatore, attraverso MoveIt! si calcola il *path* cartesiano che il manipolatore dovrà seguire in ogni istante. I dati, così raccolti, sono salvati all'interno di una struttura *DisplayTrajectory* e pubblicati su un secondo Topic dove lo script, sulla macchina virtuale, sarà il Publisher e l'applicazione Unity il Subscriber.

Il messaggio DisplayTrajectory contiene un insieme di posizioni angolari, espresse in radianti, per ogni giunto. Le informazioni all'interno del messaggio sono elaborate in tempo reale dall'applicazione Unity3D e inviate al robot virtuale che simulerà il percorso che dovrà essere compiuto dal robot reale.

Durante la simulazione di movimento, il robot virtuale istanzierà sulla scena alcuni target (di colore diverso rispetto a quelli inseriti manualmente dall'utente) utili a indicare il path compiuto dal robot. L'utente ha la possibilità di interagire con i target e inserirli nella lista dei punti per poterne modificare la posizione e l'orientamento nell'ambiente virtuale e, di conseguenza, modificare la traiettoria del robot virtuale. A partire dalla nuova lista sarà possibile eseguire un'altra simulazione.

A questo punto l'utente, se soddisfatto, può confermare la scelta dei punti e ordinare al robot reale di eseguire la traiettoria pianificata.

4.3.3 Fasi

Come già accennato, le fasi previste per il progetto sono quattro: la prima fase corrisponde alla connessione ed è già stata discussa; mentre, in tale sede, saranno dettagliatamente illustrate le tre fasi successive in correlazione con lo scopo generale. Infatti, ogni passaggio è caratterizzato da un obiettivo secondario o specifico che, fatta eccezione per la fase di controllo dei target nello spazio di lavoro, in assenza del suo superamento, non consente l'accesso alla fase successiva e conseguentemente il raggiungimento dell'obiettivo generale.

Inserimento target

L'inserimento dei target è una fase fondamentale per l'applicazione, in quanto permette di specificare la posizione e l'orientamento che il robot dovrà avere in uno specifico punto dello spazio 3D.

Dopo aver eseguito con successo la connessione, l'utente è portato sulla seconda scena dell'applicazione e ha possibilità di inserire uno o più target al suo interno: per farlo, utilizzando le *feature* messe a disposizione degli HoloLens 2 e dal *framework* MRTK, è necessario inizialmente premere il *Button* "Start" per abilitare le funzioni di inserimento punti.

Successivamente occorre selezionare, tramite tocco, un punto all'interno dello spazio di lavoro e, su esso, comparirà un target che rappresenterà un punto nello spazio 3D che l'end-effector del robot dovrà raggiungere durante il suo percorso. Il target inserito è di colore blu, ha una forma sferica e un orientamento predefinito, con la punta dell'end-effector direzionata verso il basso. Insieme al target, è istanziato sulla scena un secondo oggetto che servirà come *feedback* all'utente al fine di capire l'orientamento del robot e, di conseguenza, posizionare il target nel modo più preciso possibile.

Internamente, l'applicazione inserisce ogni target in una lista specifica e li collega tra loro tramite l'utilizzo di una curva *spline* interpolata tramite l'algoritmo *Catmull-Rom*. La spline è un componente di Unity3D chiamata LineRenderer ed è calcolata in tempo reale, utilizzando la posizione iniziale dell'end-effector del robot virtuale e quella dei singoli target.

La funzione consente di dare un'idea di massima del percorso che sarà seguito dal robot virtuale prima che i dati siano pubblicati e, quindi, calcolati dal server. Inoltre, risulta essere molto utile quando sono inseriti numerosi target, in quanto aiuta a visualizzare l'ordine in cui sono stati posti i punti.

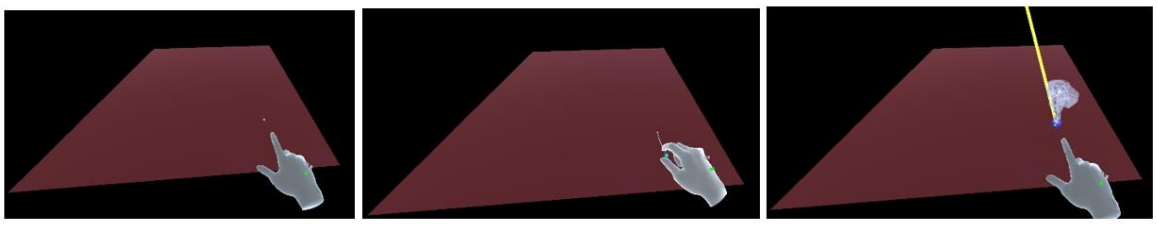


Figura 4.4: Sequenza di inserimento di un target

Controllo dei target nello spazio di lavoro

I target inseriti all'interno della scena sono istanziati con un orientamento predefinito per semplicità e immediatezza. Tuttavia, è possibile interagire con ogni target presente sulla scena

per manipolarlo o eliminarlo. L'applicazione funziona come una macchina a stati e lo stato iniziale è quello di "*Insert*". Cliccando sul Button "State", l'utente può cambiare stato e, in particolare, premendo una volta, attiva la modalità "*Cancel*" e, successivamente, la modalità "*Modify*". Sinteticamente, l'utente avrà la possibilità di:

- quando l'applicazione è nello stato "Modify", selezionare un target utilizzando il tocco e tenendo premuto. Questo permette di afferrare l'oggetto, di spostarlo e ruotarlo a piacere nello spazio 3D della scena;

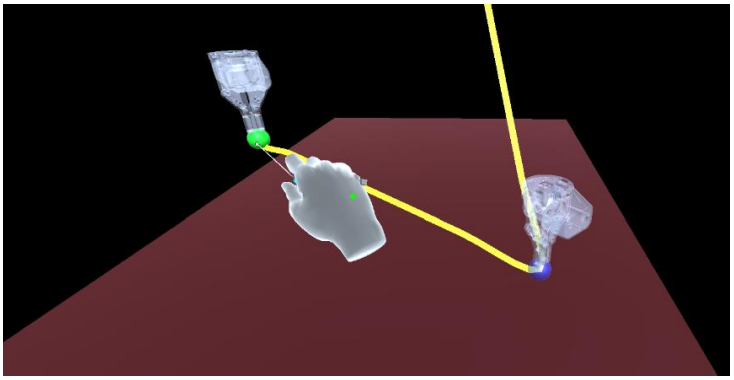


Figura 4.5: Modifica di un target

- quando l'applicazione è nello stato "Cancel", selezionare un oggetto tramite tocco consente all'utente di eliminare un target presente sulla scena.

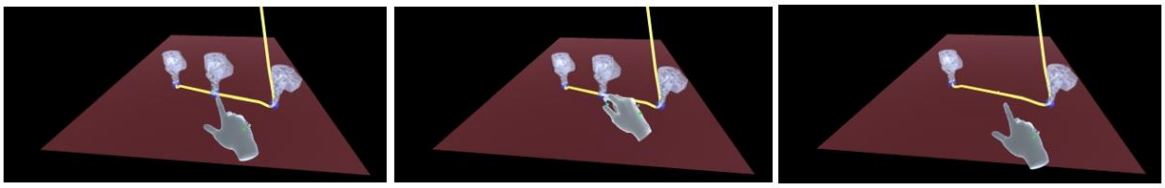


Figura 4.6: Sequenza di cancellazione di un target

- modificare la visualizzazione dell'end-effector virtuale. Ogni target è visualizzato con un end-effector virtuale posizionato sopra di esso per aiutare l'utente a individuare l'orientamento del punto. Premendo sul Button "Change view", si sostituiscono gli end-effector virtuali con i cilindri, consentendo di dare un feedback meno affidabile dell'orientamento dei singoli punti ma, allo stesso tempo, si occupa meno spazio nella scena e può risultare semplice, soprattutto per punti vicini, differenziare i target tra loro.

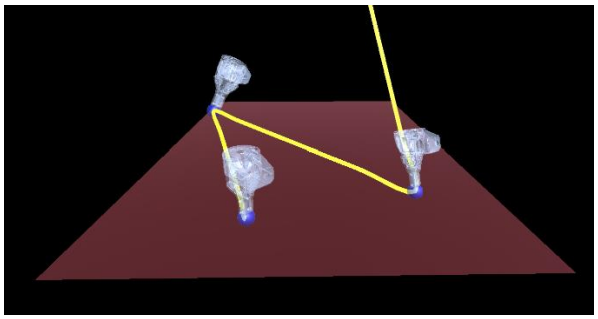


Figura 4.7: Orientamento dei target

- inserire un punto tra due già esistenti sulla scena. I punti sono elaborati dal robot nell'ordine in cui sono stati inseriti dall'utente. Se si vuole inserire un target tra due preesistenti, è necessario premere il Button "Insert between dots", dopodiché si seleziona un target presente sulla scena e su di esso comparirà una piccola finestra che permette di inserire un punto precedente o successivo a quello selezionato. Se il punto è il primo della lista, l'utente avrà la possibilità di inserire esclusivamente un punto successivo, similmente, se seleziona l'ultimo, potrà inserirne uno solo precedente.

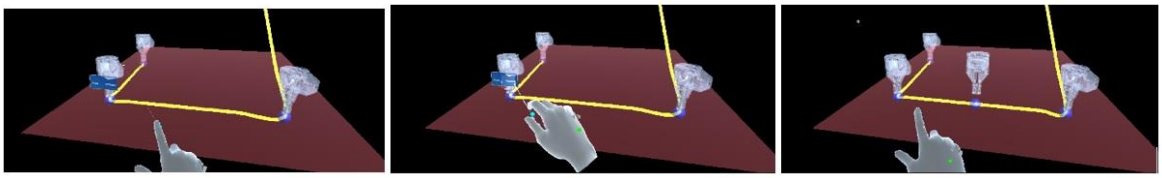


Figura 4.8: Sequenza di inserimento tra due target

- modificare la visualizzazione dei target gialli. Dopo la simulazione del robot virtuale, l'utente può nascondere la serie di target istanziati durante il movimento premendo il Button "Hide yellow points". Allo stesso modo, può attivare nuovamente la visualizzazione tramite lo stesso Button.

Invio dei punti e visualizzazione del movimento

Una volta posizionati sulla scena tutti i target e aver definito il loro orientamento, l'utente potrà far partire la simulazione di movimento del robot virtuale. Per ogni punto è calcolata la posizione e l'orientamento rispetto al sistema di riferimento del robot virtuale; i dati così raccolti sono incapsulati, tramite le feature di *ROS#*, in un messaggio di tipo *Pose* e pubblicati, uno per volta, attraverso il *Pose Publisher* su un topic *ROS*.

Successivamente, i messaggi così inviati sono catturati da un nodo del server su cui gira lo script Python collegato a MoveIt!, con il compito di processare i dati ricevuti. Tramite il calcolo della cinematica inversa, si elabora per ogni punto il moto dei giunti del manipolatore in ogni istante temporale e si salvano i dati su una struttura DisplayTrajectory. Il server ne istanzia una, dove possibile, per ogni punto ricevuto e pubblica i risultati su un nuovo topic *ROS*. Quindi, una volta pubblicati i punti e avendo ricevuto l'elaborazione degli stessi tramite il *Planning Trajectory Subscriber*, il robot virtuale simula il percorso descritto dalle strutture DisplayTrajectory.

È possibile che l'utente selezioni dei punti che non sono raggiungibili dal robot a causa di passaggi per posizioni singolari del manipolatore o per punti fuori dalla zona di lavoro; oppure, a causa di una scarsa connessione, è possibile che dei pacchetti siano persi o che arrivino incompleti, rendendo impossibile il calcolo della cinematica inversa. In questi casi, si rilevano degli errori di calcolo: l'errore è segnalato all'applicazione dal server per mezzo di un messaggio vuoto pubblicato su un topic specifico. Il punto di cui non è stato possibile calcolare la cinematica inversa assume un colore rosso all'interno della scena, così che l'utente abbia un feedback visivo e immediato dell'errore e possa agire al fine di modificare la posizione o l'orientamento del target e, successivamente, far partire una nuova simulazione.

Una volta inviati i dati relativi ai target, l'applicazione si metterà in attesa di ricevere tutti i risultati relativi ai punti e, quando ricevuti, il robot virtuale simulerà il movimento che sarà intrapreso dal robot reale. Durante il movimento, saranno istanzi dei target gialli in corrispondenza della punta operativa del robot. Questi avranno un duplice scopo: da una parte, serviranno a dare una stima più precisa e accurata del path seguito; dall'altra, possono essere inseriti nel processo di pianificazione del moto. Infatti, questi target sono interagibili tramite tocco e, se selezionati, diventeranno di colore blu e saranno aggiunti alla lista dei target, in posizione corretta rispetto alla loro posizione sulla scena, diventando riutilizzabili per una nuova simulazione.

L'utente, se soddisfatto della traiettoria visualizzata, può inviare la serie di punti al robot reale, tenendo comunque conto che la traiettoria di quest'ultimo potrebbe differire da quella simulata poiché il manipolatore virtuale presuppone un modello ideale del robot.

Capitolo 5

Test e risultati

Per valutare le funzioni e le performance dell'applicazione sviluppata sono stati condotti, all'interno del Politecnico di Torino, dei test a cui hanno partecipato 7 utenti, di cui 6 maschi e 1 femmina, con età compresa tra i 25 e i 31 anni. Il campione, se pur limitato, è stato interrogato sulle personali conoscenze in merito ai dispositivi di Realtà Aumentata e alla robotica. In relazione alle loro risposte si può affermare, supponendo, che le loro conoscenze fossero buone per le tecnologie AR/VR ed eterogenee per la seconda tematica.

Lo svolgimento dei test è stato effettuato con l'ausilio di alcuni artefatti, ottenuti tramite l'utilizzo di una stampante 3D, e posizionati all'interno del workspace del robot. Ogni artefatto rappresenta una posa che l'end-effector deve raggiungere durante il test.

Durante lo svolgimento dei test, gli utenti hanno dovuto compiere delle operazioni sui target, quali l'inserimento, la traslazione la rotazione e la cancellazione.

Sono stati condotti quattro test: i primi due sono stati pensati per condurre delle operazioni sul piano Z=0 del robot, mentre per gli altri è stato necessario lavorare nello spazio tridimensionale. In particolare, il primo test prevede l'inserimento di un singolo target, coincidente con la posizione di un artefatto posizionato da un operatore, avente orientamento predefinito rivolto come la normale al piano. Il secondo test richiede di inserire due target e di modificarne l'orientamento secondo specifiche. Per il terzo test è previsto l'inserimento di due punti e la loro traslazione nello spazio 3D, con l'orientamento specificato da un operatore. Il quarto ha consentito all'utente di pianificare liberamente il moto del robot, ma è stato guidato in tale scelta da alcuni artefatti, posti al fine di delineare posizioni note all'interno dello spazio di lavoro del robot.

5.1 Questionari

Il presente progetto ha previsto l'impiego di due strumenti, sottoforma di questionari, atti a valutare le performance dell'applicazione: il *System Usability Survey* (SUS) e il *Nasa-TLX*. Di seguito ne saranno delineate le caratteristiche.

System Usability Survey (SUS)

La System Usability Scale (SUS) fornisce uno strumento affidabile e veloce per misurare l'usabilità di un'applicazione. È un questionario di dieci domande con cinque opzioni di risposta da "Assolutamente d'accordo" a "Assolutamente in disaccordo". Ideato da John Brooke nel 1986, permette di valutare una grande varietà di prodotti e servizi tra cui: hardware, software, dispositivi mobili, siti Web e applicazioni.

I vantaggi del SUS sono:

- è un questionario molto semplice da somministrare ai partecipanti;
- può essere utilizzato su campioni di piccole dimensioni con risultati affidabili;
- può distinguere efficacemente tra sistemi utilizzabili e inutilizzabili.

Il questionario e i punteggi sono delineati nel modello System Usability Scale (SUS) e l'interpretazione del punteggio può essere complessa. I punteggi di ogni partecipante, per ciascuna domanda, sono convertiti in un nuovo numero, sommati e quindi moltiplicati per 2,5 per trasformare i punteggi originali da un range 0-40 ad uno 0-100 (General Services Administration, 2021). La figura 5.1 fornisce un'idea generica sull'attribuzione dei punteggi di tale questionario.

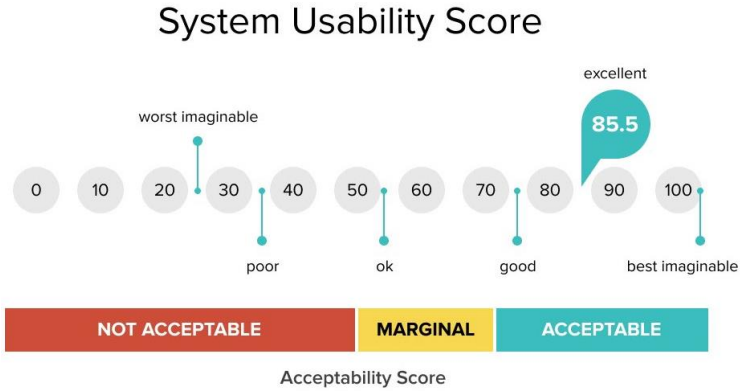


Figura 5.1: Scala di punteggio del questionario SUS

Fonte: Smyk, 2020.

Nasa-TLX

Il *Task Load Index* (TLX) è uno strumento che consente agli utenti di valutare il carico di lavoro di coloro che operano con diversi sistemi di interfaccia uomo-macchia. Sviluppato negli anni '80 come questionario su carta e matita, oggi NASA-TLX è diventato uno standard per misurare il carico di lavoro soggettivo in un'ampia gamma di applicazioni.

Grazie a una procedura di valutazione multidimensionale, NASA-TLX attribuisce un punteggio del carico di lavoro complessivo basato su una media ponderata di valutazioni su sei sotto-scale:

- Domanda mentale
- Domanda fisica
- Domanda temporale
- Prestazioni
- Sforzo
- Frustrazione

NASA-TLX è stata utilizzato con ottimi risultati per valutare il carico di lavoro in vari ambienti come le cabine di pilotaggio degli aerei; postazioni di comando, controllo e comunicazione (C3); supervisione e controllo di processo; simulazioni e prove di laboratorio (So, 2020). L'interpretazione dei risultati è mostrata, sottoforma di tabella, in figura 5.2.

Workload	Value
Low	0-9
Medium	10-29
Somewhat high	30-49
High	50-79
Very high	80-100

Figura 5.2: Interpretazione dei punteggi del NASA-TLX.

Fonte: Hancock & Meshkati, 1988.

5.2 Risultati

I risultati ottenuti sono relativi alla valutazione dell'interfaccia dell'applicazione da parte di tutti i partecipanti dell'esperimento, i quali sono riusciti a svolgere in pieno i compiti a loro assegnati.

In particolare, il primo questionario proposto è stato il System Usability Scale (SUS), i cui risultati sono riportati in *Tabella 5.1*.

	MEDIA	DEVIAZIONE STANDARD
D1	4,285714	0,755928946
D2	1,285714	0,487950036
D3	4,142857	0,377964473
D4	1,571429	0,975900073
D5	4,142857	0,690065559
D6	1,285714	0,755928946
D7	4,285714	0,755928946
D8	1,571429	0,786795792
D9	4,285714	0,755928946
D10	1,571429	0,534522484

Tabella 5.1: Risultati questionario SUS

Secondo le specifiche del questionario SUS, le domande con indice dispari hanno un'asserzione positiva, dove perciò si tenta di raggiungere un punteggio elevato; mentre quelle con indice pari presentano un'asserzione negativa, in cui l'obiettivo è ottenere un punteggio basso.

Analizzando i dati, è possibile evidenziare che gli utenti abbiano ritenuto vere le seguenti affermazioni proposte:

- *I think that I would like to use this app frequently*
- *I would imagine that most people would learn to use this app very quickly*
- *I felt very confident using this app*

Quindi, gli utenti hanno considerato l'applicazione un utile strumento per svolgere i propri compiti nei suoi contesti di utilizzo. In secondo luogo, hanno sottolineato la semplicità di apprendimento e la facile fruibilità di questo strumento.

È interessante, in tale sede, osservare anche la presenza di valori che possono essere considerati come positivi, ma nel complesso hanno ottenuto i risultati peggiori.

- *I think that I would need assistance to be able to use this app*
- *I found this app very cumbersome/awkward to use*
- *I needed to learn a lot of things before I could get going with this app*

In sintesi, questi sono gli aspetti che, grazie alla valutazione data, potrebbero essere migliorati al fine di garantire una migliore usabilità di questa tecnologia.

Riguardo alla domanda sul bisogno di assistenza per l'utilizzo dell'applicazione, è stato suscitato più disaccordo rispetto alla media: è stato, infatti, possibile evincere che le conoscenze pregresse possedute dagli utenti abbiano influito sulla fruibilità dello strumento. Coloro che avevano conoscenze proprie circa le tecnologie utilizzate hanno avuto meno difficoltà, a differenza invece di quelli che non avevano conoscenze in merito.

Nel complesso, l'applicazione ha ricevuto dagli utenti una valutazione con un punteggio finale pari a 84,64286 che, rifacendosi alla *Figura 5.1*, esprime una valutazione molto alta dell'applicazione proposta. Infatti, il valore corrispondente alla valutazione "eccellente" (85,5) è prossimo alla valutazione effettivamente ottenuta.

A differenza del questionario SUS, il questionario NASA-TLX ha come obiettivo l'ottenimento di risultati più bassi possibili. Nella *Tabella 5.2*, sono riportati i risultati ottenuti dall'applicazione per ogni macrocategoria.

	MEDIA	DEVIAZIONE STANDARD
Mental Demand	38,57143	17,72810521
Physical Demand	12,85714	10,35098339
Temporal Demand	19,28571	17,18249386
Performance	37,85714	28,41025972
Effort	30	15,8113883
Frustation	17,14286	13,183684

Tabella 5.2: Risultati questionario NASA-TLX

Analizzando gli esiti, si può notare come, in media, i valori di "domanda fisica", "domanda temporale" e "frustrazione" siano quelli che hanno ricevuto il punteggio migliore. Ciò implica che l'applicazione non impatti sull'aspetto fisico dell'individuo, tratto prevedibile poiché il

lavoro dell'utente si limita all'inserimento di target e all'utilizzo di bottoni, né quello temporale, in quanto l'utenza è riuscita a risolvere i task proposti in tempo ridotto.

I valori ottenuti dalla "frustrazione" sono risultati molto positivi perché dimostrano che a ridotta frustrazione equivale una facilità di utilizzo e immediatezza dell'applicazione: aspetto non scontato con una tecnologia, come gli HoloLens 2, poco utilizzata e poco accessibile.

Valori leggermente più alti sono stati evidenziati per le categorie di "domanda mentale", "sforzo" e "prestazione", mettendo in luce un impegno maggiore da parte del campione. In particolare, i risultati ottenuti in merito alle prime due categorie sono stati omogenei nella valutazione dell'applicazione; quindi, la maggior parte dei soggetti ha attribuito punteggi simili. Discorso a parte riguarda le prestazioni dell'applicazione perché, a discapito del valore medio, si nota un'elevata deviazione standard. Ciò sembrerebbe indicare un'importante eterogeneità nella valutazione della performance.

Questo è attribuibile alla differenza all'interno del campione di un unico soggetto che, per questa categoria, ha attribuito un valore che si discosta molto dalla media. È facile ipotizzare che, in presenza di un campione più ampio, tale valore potrebbe variare in conseguenza alla numerosità.

Nel complesso, l'applicazione ha raggiunto una valutazione pari a 31,76 che, facendo riferimento alla *Tabella 5.2*, denota un carico di lavoro leggermente sopra la media ma comunque accettabile. È utile ricordare che la scala di valutazione è strutturata in modo tale che a valori minori corrispondano risultati migliori.

Una volta conclusi i test, è stato chiesto agli utenti di fornire commenti individuali sull'applicazione stessa per evidenziarne criticità o pregi.

È emerso che, tra i fattori positivi, l'applicazione risulta ben fatta e facile da usare; mentre è stato sottolineato che la funzione riguardante la macchina a stati tende a rendere l'applicazione più difficile e meno immediata da utilizzare. Inoltre, è stato evidenziato che, seppur non sia di diretta competenza, il tracking ottenuto tramite l'utilizzo di Vuforia sia poco stabile e non del tutto preciso. Infine, molti utenti hanno evidenziato una problematica, interna all'HoloLens 2, dovuta all'eccessiva sfocatura delle immagini.

Capitolo 6

Conclusioni e sviluppi futuri

L'elaborato ha cercato di fornire, se pur in modo parziale, un contributo sia bibliografico sia applicativo alla Human Robot Interaction (HRI). Si è tentato di raggiungere tale obiettivo mediante l'ideazione di un programma ad hoc che risultasse di facile impiego per le aziende.

L'applicazione progettata, grazie ai test effettuati presso il Politecnico di Torino, si è dimostrata semplice, intuitiva e facile da utilizzare. Alla luce di tali caratteristiche funzionali, si potrebbe prospettare un suo utilizzo in ambito industriale per aumentare la sicurezza della forza lavoro.

Al fine di proporre una tesi completa ed esaustiva, è stata condotta un'indagine bibliografica mediante l'utilizzo delle principali banche dati quali Ebsco, Google Scholar, Pubmed e ResearchGate.

Queste sono state adoperate al fine di approfondire la tematica, analizzandone i vari fattori quali per esempio pregi, difetti e ambiti applicativi. Inoltre, ci si è concentrati sul tema, complesso e fondamentale, della tecnologia HoloLens 2.

Tale tecnologia è sicuramente poco conosciuta dalla massa e, al contempo, ancora poco impiegata. L'analisi della letteratura e l'utilizzo pratico dell'HMD hanno, infatti, evidenziato quanto ancora occorra approfondire lo studio e l'applicazione tecnologica per intensificare le conoscenze a riguardo.

Questa considerazione è coerente con l'aspettativa iniziale dell'elaborato, secondo la quale lo studio e l'approfondimento di un tema permetta di individuare nuove chiave di lettura e di comprensione.

È evidente la complessità di una tematica quale la Realtà Mista ma spesso si perde di vista l'obiettivo di questi studi. Oltre al bisogno di comprensione teorica, c'è l'esigenza di fornire applicazione di tali tecnologie che diventino utili ed efficaci strumenti in diversi ambiti applicativi. Ecco perché lo studio e la ricerca continui rispondono a specifici bisogni personali e professionali, nella speranza che queste ricerche possano, un giorno, semplificare la vita delle persone indipendentemente dalle specifiche condizioni in cui esse si trovano.

Il presente studio integra la letteratura già esistente riguardante la Realtà Mista e il suo impiego in ambito industriale, poiché diversi studi precedenti avevano già fornito materiale in questo senso ma, in tale sede, si è tentato di coniugarlo alla robotica per creare un discorso completo e omogeneo.

Tuttavia, è importante considerare che questa tesi si è concentrata esclusivamente sulla definizione di traiettorie per manipolatori robotici mediante interfacce in Realtà Mista. Una volta evidenziata la base teorica di queste tematiche, considerando anche altri fattori secondari che potrebbero entrare in gioco durante il percorso che conduce agli esiti dei test, le riflessioni in merito potrebbero variare.

Il tema è assai complesso e la medesima tecnologia è in continuo mutamento e sviluppo; ivi per cui ogni dichiarazione espressa in tale sede appare vera ora e in tale contesto ma, al contempo, rimane suscettibile di possibili variazioni.

Si è tentato, quindi, di discuterlo in modo completo ma comunque semplice: una volta illustrati i risultati ottenuti tramite i test e le conclusioni a cui è stato possibile giungere, occorre ora proporre alcune possibili prospettive future mirate a migliorare questo progetto.

Una raccomandazione per ulteriori ricerche future potrebbe essere quella di realizzare uno studio simile approfondendo differenze legate alle caratteristiche del software, al rapporto tra tecnologia e contesto; oppure sarebbe curioso effettuare una ricerca analoga che però possa indagare quantitativamente l'impiego della Realtà Mista concentrandosi nel dettaglio su un ambito di applicazione tra quelli discussi nella parte introduttiva dell'elaborato.

Sarebbe, poi, interessante individuare una soluzione che possa ridurre i tempi di attesa tra l'invio dei punti target e l'inizio del movimento del robot virtuale. Per via delle numerose connessioni

presenti nel sistema, infatti, la latenza dell'applicazione tende a essere molto alta e, in caso di scarsa connessione, è possibile che i dati siano scartati. Inoltre, l'utilizzo prolungato degli HoloLens 2 tende a creare nausea e malessere all'operatore. Diminuire i tempi di simulazione significherebbe quindi ridurre anche l'utilizzo dell'HMD.

Un ulteriore miglioramento che si potrebbe apportare al software consiste nella possibilità di modificare il path compiuto dal manipolatore dopo la simulazione. Al momento, dopo aver visualizzato la traiettoria in ambiente virtuale, l'utente può inviare la serie di punti al manipolatore reale o iniziare una nuova simulazione, ma non è consentita la modifica in tempo reale.

Altro aspetto da prendere in considerazione è la possibilità di manipolare un singolo braccio robotico, anziché l'intera catena cinematica, in modo da condurre traiettorie diverse da quelle inizialmente calcolate dall'applicazione. Può essere utile in ambienti in cui sono presenti ostacoli o sia necessario uno specifico movimento.

Al momento, il passaggio della punta operativa da un target al successivo avviene tramite un movimento point to point (PTP). In tal senso, può risultare utile implementare una metodologia che permetta di seguire altri tipi di traiettorie, per esempio la traiettoria ad arco.

Infine, l'applicazione, per come è stata sviluppata, è utilizzabile esclusivamente con il robot e.Do. Un eventuale sviluppo futuro potrebbe essere, quindi, la possibilità di scegliere il manipolatore da utilizzare tramite un'interfaccia virtuale o la scansione di un codice QR.

Ringraziamenti

Giunto alla fine di questo elaborato, mi sembra doveroso ringraziare le persone che mi hanno accompagnato e sostenuto durante il percorso.

Ringrazio, in principio, il mio relatore Andrea Sanna per avermi dato la possibilità di lavorare a un progetto interessante e innovativo, che mi ha concesso di poter utilizzare degli strumenti all'avanguardia nel mondo IT e approfondire alcune tematiche contemporanee come la Realtà Mista e l'HRI.

Ringrazio i miei correlatori Federico Manuri e Francesco De Pace, per l'aiuto e la disponibilità sia durante la fase di sviluppo del progetto, sia durante la fase di test; e per gli utili consigli che mi hanno fornito.

Ringrazio i miei genitori: Silverio e Federica per il sostegno durante questi anni di studio, sia nei momenti migliori, sia nei momenti più difficili della mia carriera. Non ce l'avrei fatta senza il vostro aiuto.

Ringrazio le mie sorelle, Laura ed Enza, per la loro disponibilità, i loro consigli e il loro interesse durante questi anni universitari.

Ringrazio i miei tre nipoti, Alessandro, Emma e Federica, per la felicità che mi donano ogni giorno, per loro contagiosa allegria e per essere uno dei regali più belli che la vita mi ha fatto.

Ringrazio Franco e Fulvia, per essermi sempre stati vicini in questi anni, per essere stati presenti in ogni momento e per avermi spronato a dare sempre il massimo in ogni situazione.

Ringrazio Simona, per tutto l'aiuto, il sostegno, i consigli e la pazienza profusi in questi mesi di scrittura della tesi e per la costante presenza nei momenti di bisogno.

Ringrazio tutti i miei amici che, tra una chiacchierata, una passeggiata o una serata passata insieme, mi hanno permesso di passare bei momenti e mi hanno distratto dalle preoccupazioni di questo periodo difficile.

BIBLIOGRAFIA

AA. VV. 2014. About ROS. In *Ros*.

AA. VV. 2021. Introduzione a ROS. In *Macheronte*.

ABB. 2018. Manuale dell'operatore Guida introduttiva - IRC5 e RobotStudio. In *Robotics*.

Advanced Knowledge & Technology. 2017. I linguaggi di programmazione: cosa sono e quali sono i migliori. In *AKT*.

Aurea s.r.l. 2020. Realtà Mista. In *Aurea Lab*.

Billard A., Calinon S., Dillmann R., Schaal S. 2018. Robot Programming by Demonstration. In *Robotics*.

Bischoff M. 2021. ROS#. In *GitHub*.

Chacko S. M., Kapila V. 2019. *An Augmented Reality Interface for Human-Robot Interaction in Unconstrained Environments*. China: International Conference on Intelligent Robots and Systems.

Chapagain S. 2018. *Application development with vuforia and unity 3D*. Centria University Of Applied Sciences.

Comau SpA. 2021. e.DO People Learn Robotics. In *Edo.cloud*.

Comau SpA. 2021. e.DO™ robot compatto, modulare e open source. In *Comau*.

Comau SpA. 2021. Questo è e.DO. In *Edo.cloud*.

Conterno N. 2019. Sviluppo di una linea di confezionamento con robot collaborativi. Università degli studi di Padova.

Croccolino L. 2019. *La piattaforma ROS per lo sviluppo di applicazioni per la robotica: panoramica e caso di studio*. Università di Bologna.

Dillmann R. 2004. Teaching and learning of robot tasks via observation of human performance. In *Robotics and Autonomous Systems*; 47: 109–116.

Gabbrielli M., Martini S. 2011. *Linguaggi di programmazione: principi e paradigmi*. Milano: McGraw-Hill.

General Services Administration (GSA). 2021. System Usability Scale (SUS). In *Usability*.

Generali M. 2020. Realtà aumentata, virtuale e mista: definizione e utilizzi. In *Tecnopolo*.

GlobalData Technology. 2019. Early signs are promising but augmented reality still has some way to go. In *Verdict*.

Gori T. 2016. Come si addestra l'esercito con la realtà virtuale e aumentata in Italia. In *Vice*.

Groover M. P. 2008. *Automation, Production Systems, and Computer-Integrated Manufacturing*. Londra: Pearson Education.

Haas J. 2021. A History of the Unity Game Engine An Interactive Qualifying Project. Worcester Polytechnic Institute.

Hancock P. A., Meshkati N. 1988. Human Mental Workload. In *Elsevier Science Publishers*.

Hejlsberg A., Wiltamuth S., Golde P. 2006. The C# Programming Language. Boston: Addison-Wesley Professional.

ICT Value Consulting. 2018. Intelligenza Artificiale – tecnologie e ambiti applicativi. In *Intelligenza Artificiale – Report*.

Insight Technology Solutions S.R.L. 2021. Microsoft HoloLens 2. In *Insight*.

Intel Corporation. 2021. Demistificare il panorama della realtà virtuale. In *Intel*.

Istituto della Enciclopedia Italiana. 2017. *Treccani. Dizionario della lingua italiana*. Firenze: Giunti.

KG Partners srl, 2021. Mixed Reality. In *KG Partners*.

Laschi C. 2015. *Architetture di supervisione e controllo di robot*. Università di Pisa.

Lotsaris K., Gkournelos C., Fousekis N., Kousi N., Makris S. 2021. AR based robot programming using teaching by demonstration techniques. In *Procedia CIRP*; 97: 459-463.

Lubanovic B. 2015. Introducing Python. In *Academia*.

Macina V. 2020. Realtà virtuale, aumentata e mista: differenze e applicazioni. In *Industry*.

Maldera L. 2020. ROBOT in Azione: quali sono gli ambiti di applicazione …? In *Emme.A*.

Manring L., Pederson J., Potts D., Boardman B., Mascarenas D., Harden T., Cattaneo A. 2020. Augmented Reality for Interactive Robot Control. In *Special Topics in Structural Dynamics & Experimental Techniques*; Vol. 5. Conference Proceedings of the Society for Experimental Mechanics Series.

Melotti E. 2016. Perché usare Python. In *html*.

Michalosa G., Karagiannisa P., Makrisa S., Tokçalarb Ö., Chryssolourisa G. 2016. Augmented Reality (AR) Applications for Supporting Human-robot Interactive Cooperation. In *Procedia CIRP*; 41: 370-375.

Microsoft. 2021. Informazioni HoloLens 2. In *Microsoft*.

Microsoft. 2021. Presentazione del linguaggio C#. In *Microsoft*.

Naini A. 2019. Visual Studio 2019 – 10 New Features and Improvements. In *Geekflare*.

Ong S. K., Yew A. W. W., Thanigaivel N. K., Nee A. Y. C. 2020. Augmented reality-assisted robot programming system for industrial applications. In *Robotics and Computer-Integrated Manufacturing*; 61.

Open Source Robotics Foundation. 2017. Package Summary. In *Ros*.

Open Source Robotics Foundation. 2020. Writing a Simple Publisher and Subscriber (Python). In *Ros*.

Ostanin M, Mikhel S., Evlampiev A,, Skvortsova V., Klimchik A. 2020. Human-robot interaction for robotic manipulator programming in Mixed Reality

Ostanin M., Klimchik A. 2018. Interactive Robot Programing Using Mixed Reality. In *IFAC*; 51:(22) 50-55.

Ostanin M., Klimchik A., Yagfarov R. 2019. Interactive Robots Control Using Mixed Reality. In *IFAC*; 52.

Porcari V. D. 2018. *La conservazione del patrimonio artistico, architettonico, archeologico e paesaggistico*. Matera: XIV Congresso Internazionale Di Riabilitazione Del Patrimonio.

PTC Inc. 2021. Image Targets. In Vuforia.

Quintero C. P., Li S.; Pan M. K. X. J., Chan W. P., Van der Loos H. F. M., Croft E. 2018. *Robot Programming Through Augmented Trajectories in Augmented Reality*. New Jersey: IEEE.

Remondino M. 2018. Applicazioni delle tecnologie immersive nell'industria e Realtà Aumentata come innovazione di processo nella Logistica: stato dell'arte ed implicazioni manageriali. In *Electronic Journal of Management*.

Riba C. 2018. *BIM e VR per le strutture sanitarie tra gestione e riabilitazione. Caso Studio: l'Ospedale S.S. Trinità di Fossano*. Politecnico di Torino.

Robotnik. 2014. "MoveIt!" a standard for manipulation mobile. In *Robotnik*.

Rossi D., Meschini A., Feriozzi R., Olivieri A. 2018. Cose dell'altro mondo. La realtà virtuale immersiva per il patrimonio culturale. In *Ambienti digitali per l'educazione all'arte e al patrimonio*; 240-256. Milano: Franco Angeli.

Rossi S. 2019. Perché usare C#? In *Redchar*.

Siciliano B. 2018. Introduzione alla Robotica. In *Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G. 2009. Robotics. Modelling, Planning and Control. Berlino: Springer*.

Smyk A. 2020. The System Usability Scale & How It's Used in UX. In *Adobe*.

So P. 2020. NASA TLX: task load index. In *National Aeronautics and Space Administration* (NASA).

Sucan I. A., Chitta S. 2021. Move Group Python Interface. In *MoveIt*.

Sucan I. A., Chitta S. 2021. Moving robots into the future. In *MoveIt*.

Technogym S.p.A. 2021. Come la Mixed Reality cambierà il tuo allenamento. In *Technogym*.

Tucci F. 2014. *Realtà virtuale e realtà aumentata nel campo di battaglia*. Milano: Il caffè Geopolitico.

Tzafestas S. G. 2020. Introduction to Intelligent Robotic Systems. *In Tzafestas S. G. 2020. Intelligent Robotic Systems. Florida: CRC Press*.

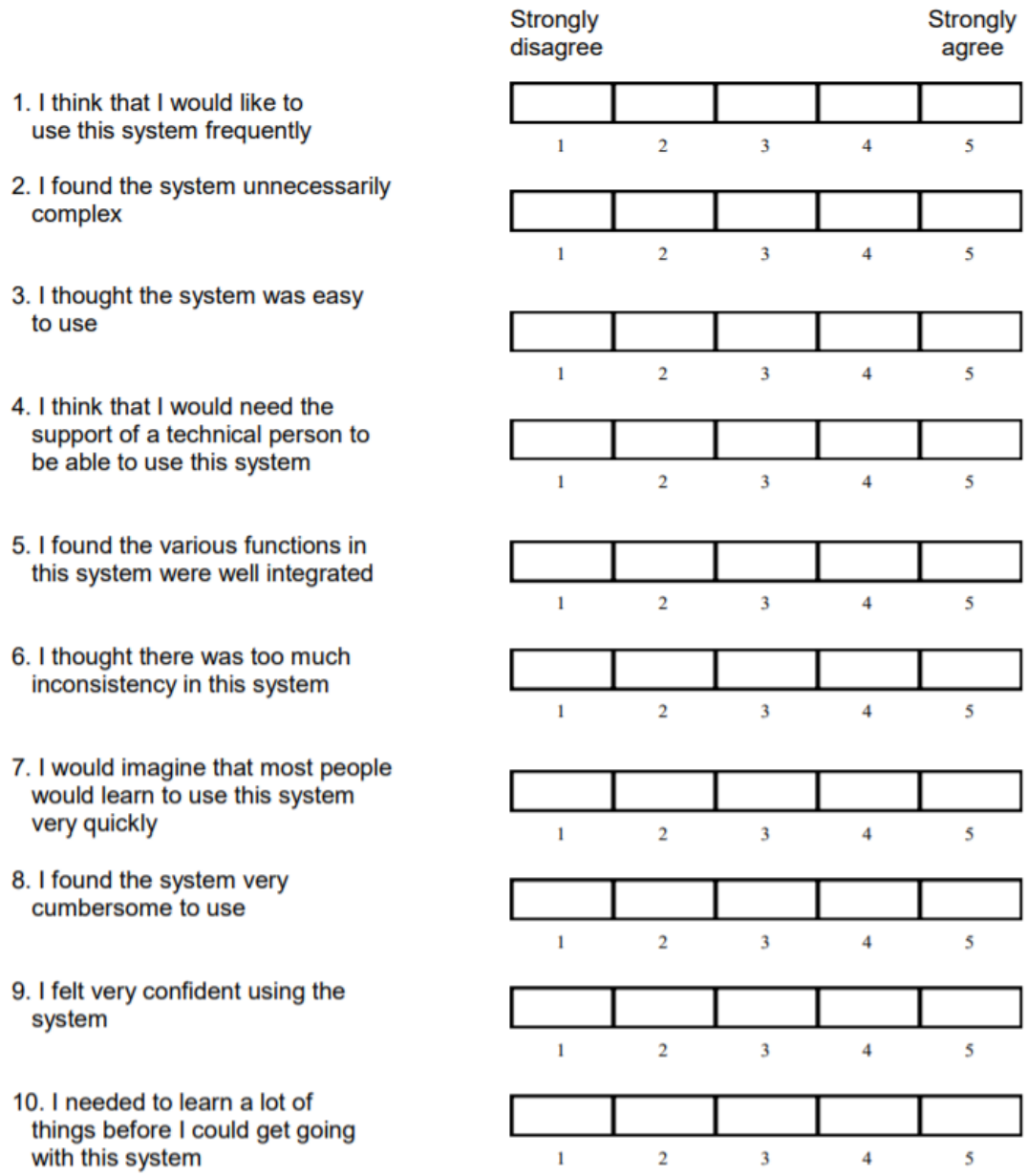
Ungureanu D., Bogo F., Galliani S., Sama P., Duan X., Meekhof C., Stuhmer J., Cashman T. J., Tekin B., Schonberger J. L., Olszta P., Pollefeys M. 2020. HoloLens 2 Research Mode as a Tool for Computer Vision Research. In *arXiv*; 11239.

Young S. N., Peschel J. M. 2020. *Review of Human–Machine Interfaces for Small Unmanned Systems With Robotic Manipulators*. Parigi: IEEE Transactions on Human-Machine Systems; 50: 2.

Zimmermann S. A., Berninger T. F. C., Derkx J., Rixen D. J. 2020. *Dynamic modeling of robotic manipulators for accuracy evaluation*. Parigi: IEEE International Conference on Robotics and Automation (ICRA).

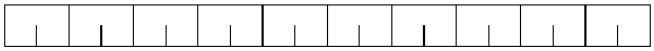
APPENDICE

Questionario SUS



NASA-TLX

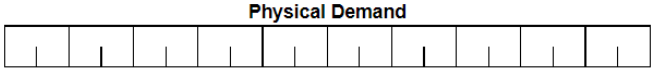
Mental Demand



Low High

How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?

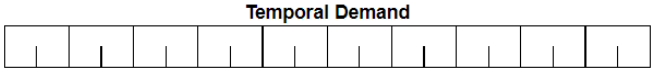
Physical Demand



Low High

How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

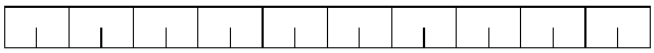
Temporal Demand



Low High

How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

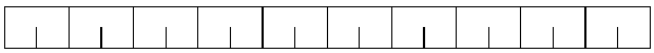
Performance



Good Poor

How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

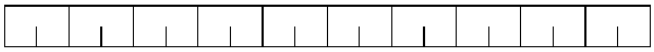
Effort



Low High

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration



Low High

How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?