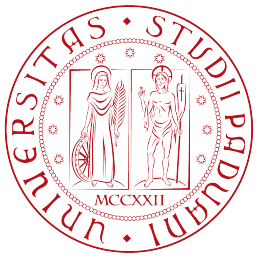

Università degli Studi di Padova

Dipartimento di Tecnica e Gestione dei sistemi industriali

Corso di Laurea Triennale
in Ingegneria Meccatronica

Sviluppo di un'applicazione industriale in Realtà Mista con HoloLens

Relatrice:
Chiar.ma Prof.ssa Monica Reggiani

Laureando:
Enrico Casarotto
1163695

Anno Accademico 2021/2022

A Oriella e Claudio che mi hanno sempre aiutato,
ai miei nonni sempre presenti,
a Michela per il sostegno e l'amore
e agli amici che mi sono sempre stati vicini.

Ringrazio la Prof.ssa Monica Reggiani e i colleghi
Piero Pettenà ed Enrico Costa per lo sviluppo di questa tesi.

Sommario

L'obiettivo di questa tesi è presentare uno studio sulla tecnologia immersiva e sui vari campi che comprende.

L'elaborato è articolato su quattro punti principali.

Il primo punto riguarda il significato di realtà mista, comparandola con quella virtuale e aumentata. Si accennerà alla storia di questa tecnologia e poi si analizzeranno gli effetti negativi sulla salute, che è una tematica molto importante da considerare.

Il secondo punto sarà un'analisi di mercato per permettere uno sguardo più ampio su ciò che è già stato sviluppato e come viene utilizzato nell'industria. Verrà fatto un elenco degli attuali dispositivi in commercio per mostrare le alternative e si proporranno alcune applicazioni future di questa tecnologia.

Il terzo punto, e ultimo argomento teorico, sarà incentrato sulle caratteristiche tecniche del visore HoloLens 2: saranno analizzate tutte le tematiche principali, la costruzione, il display, i sensori e le tecniche utilizzate per far fronte a diverse problematiche. Si passerà poi a una prova pratica in laboratorio, che consentirà di verificare l'effettiva efficienza del visore.

Infine verrà trattata la creazione di un'applicazione, tramite software Unity, utilizzabile in ambito industriale, che permette di simulare il posizionamento di diverse risorse nello spazio e acquisirne la posizione.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei codici

Capitolo 1

Introduzione

Normalmente, quando si parla di tecnologia si pensa all'ultimo computer, invece il concetto andrebbe svincolato dalla pura elettronica e apprezzato in maniera più ampia e interdisciplinare. Una tecnologia è uno strumento che ci permette di sperimentare e di ottenere nuovi risultati. Le tecnologie, infatti, provocano un cambiamento dell'atteggiamento e della concezione delle persone sul mondo. Esse sono in continua evoluzione: la ruota di legno è diventata di ferro e poi di carbonio, la stampa a caratteri mobili è diventata rotativa, poi a laser e infine in 3D.

La stesura di questa tesi è stata fatta a sei mani, in quanto l'argomento della realtà aumentata è molto vasto ed è di recente introduzione, anche se le sue origini sono abbastanza datate. Saranno così affrontate tematiche diverse per poter comprendere e analizzare ad ampio raggio il mondo che sta dietro a questa tecnologia.

L'interesse per questo argomento deriva dal fatto che permetterà grandi miglioramenti in futuro. Ciò lo rende un ambito di ricerca stimolante soprattutto in campo ingegneristico dove, spesso, sono richiesti dispositivi che permettano di agevolare le operazioni più complicate. Con le attuali potenzialità computazionali dell'informatica, la realtà aumentata è facilmente utilizzabile e permette di simulare oggetti reali in 3D con ottime prestazioni, fattore che può dare un vantaggio nel campo produttivo industriale. Alla base di questo elaborato vi è l'analisi dello sviluppo dei visori e del loro utilizzo in svariate casistiche, che possono spaziare dall'ambito domestico a quello lavorativo all'interno di un'azienda. Sono dunque le importanti prospettive e aspettative di questa tecnologia a giustificare il lavoro svolto. La continua evoluzione delle realtà digitali, come accade per molti altri settori della ricerca, impone la necessità di capirne a fondo lo stato attuale e con-

frontarlo con la sua evoluzione. Ciò permette di stimare un tasso indicativo di crescita della tecnologia, utile per proiettare aspettative sul futuro più prossimo.

Ci si focalizzerà su un particolare modello di visore, ovvero HoloLens 2 di Microsoft. È quello che offre le migliori potenzialità e tecnologie sul mercato, grazie anche alla possibilità di interagire facilmente con gli ologrammi visualizzati. È stato possibile testare il visore in laboratorio per analizzarne la struttura e le funzionalità, oltre che all'interazione con l'ambiente.

Infine, grazie al motore di gioco Unity, si è sviluppata un'applicazione per HoloLens 2 che permette di simulare un ambiente di lavoro visualizzando ologrammi di oggetti reali che possono essere spostati a piacimento e di cui è possibile salvare la posizione e ricaricarla in un successivo momento.

Capitolo 2

La realtà mista

Una tecnologia è uno strumento che ci permette di sperimentare e di ottenere nuovi risultati. Come sostiene Antonio Laudazi (fondatore di Marte5[1]) è un "*moltiplicatore di opportunità*" [1]. Le realtà virtuali, aumentate e miste sono esempi di moderne tecnologie, strettamente collegate, ed in fase di sviluppo da qualche decennio.

In questo lavoro abbiamo esplorato in particolare il campo della *realtà mista* applicata ad un visore, cercando di capirne l'evoluzione e gli utilizzi, con l'obiettivo di misurare il potenziale di questo valido strumento. Per ottenere un'inquadratura generale di questa tecnologia è necessario innanzitutto distinguerla formalmente dalle realtà virtuali e aumentate. Successivamente è essenziale analizzarne l'evoluzione fino al suo stato attuale ed esplicitare le caratteristiche che stanno frenando il suo sviluppo.

2.1 Cos'è la realtà mista?

La realtà mista (*mixed reality* o *MR*) è un'esperienza che mescola il mondo reale con quello virtuale, tramite la proiezione di informazioni digitali principalmente visive e auditive. Queste, dopo essere state generate da un processore, vengono solitamente visualizzate sul display di uno smartphone o di un visore, che in contemporanea permette la visione del mondo reale sullo sfondo. Si pensi ad esempio al navigatore Google Maps che nel 2020 ha implementato la funzione *Live View* che consiste in indicazioni stradali sotto forma di frecce sovrapposte alle immagini provenienti dalla fotocamera del cellulare [16].

[1] Agenzia specializzata in realtà virtuale e realtà aumentata.

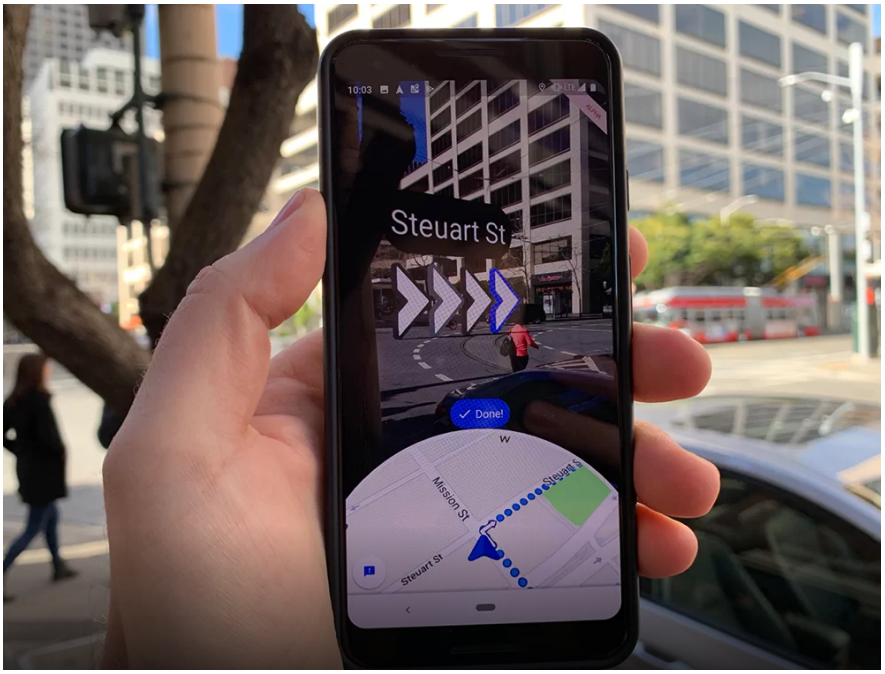

Figura 2.1: Live View Google Maps (Fonte: TechCrunch)

Si tratta dunque di una sovrapposizione in tempo reale di informazioni che possono essere testuali oppure oggetti veri e propri, ai quali ci si riferisce con il termine ologrammi. Soprattutto questi ultimi, vengono sovrapposti in maniera intelligente al mondo reale, dando l'illusione di essere presenti nella realtà. Sono oggetti integrati talmente bene con l'ambiente da poterci interagire con le mani o con un controller. Per chiarire le idee, un esempio di semplice realtà mista è quello dei filtri in real-time sullo smartphone, in cui muovendosi o toccando lo schermo essi reagiscono in qualche modo.

La sensazione che l'utente dovrebbe provare, utilizzando dispositivi di realtà mista, è quella di avere un piede nel mondo reale e un piede nel mondo virtuale. L'obiettivo, infatti, è quello di offrire all'utente un'interfaccia istintiva con le informazioni digitali, che escono dai limiti imposti da un semplice display. La recente comparsa nel mercato di visori associati alla realtà mista è un chiaro indicatore della necessità di elevata potenza di calcolo e hardware dedicato per questo settore. Mentre la prima può sembrare un limite temporaneo grazie al rapido sviluppo degli smartphone, l'hardware specifico è anche un'opportunità per immergere in maniera più naturale l'utente nell'esperienza virtuale e reale.

Il termine *realtà mista* (o *realtà mixata*), non ancora ampiamente utilizzato, venne introdotto da Paul Milgram e Fumio Kishino in un articolo del 1994 [2], in cui si delinea una classificazione dei dispositivi che secondo l'autore non appartenevano ne al dominio del mondo reale, ne al dominio della realtà virtuale.

2.2 Differenze tra realtà aumentata, mista e virtuale

È conveniente ora fare una distinzione tra i termini *aumentata*, *mista* e *virtuale*, spesso confusi oppure usati erroneamente. Nell'articolo [2], Milgram e Kishino esposero il concetto di *continuità virtuale* (dal termine inglese *virtuality continuum*) rappresentante un intervallo immaginario di tecnologie e realtà ad oggi conosciute. La rappresentazione seguente dovrebbe chiarire le idee, partendo dal mondo reale (a sinistra) e arrivando alla realtà virtuale (a destra).

Figura 2.2: Continuità virtuale (Fonte: Wikipedia)

L'ambiente reale è quello costituito solamente da oggetti reali, ovvero gli oggetti che hanno una vera e propria esistenza e che per essere visti possono essere osservati direttamente, oppure campionati e sintetizzati su un display. Gli oggetti virtuali, invece, non esistono fisicamente e per essere osservati è necessaria la loro creazione virtuale con modelli matematici, seguita dalla sintetizzazione sul display.

Realtà virtuale

La ormai celebre *realtà virtuale* (VR, dal termine *virtual reality*), estremo destro della continuità virtuale, è la tecnologia che immerge completamente (almeno per quanto riguarda i sensi della vista e dell'udito) l'utente in un ambiente artificiale e digitale. Questo può essere verosimile al mondo reale oppure completamente differente, con sue particolari leggi fisiche. La creazione dell'ecosistema tridimensionale richiede una rilevante elaborazione grafica, e per ottenere risultati soddisfacenti, anche una notevole attenzione all'interfaccia tra virtualità e sensi umani oltre che alla sincronizzazione tra il movimento del visore e il video proiettato. La realtà virtuale, infatti, è tipicamente utilizzata attraverso moderni visori ottici che ne massimizzano l'immersività e con controller per interagire con gli oggetti virtuali.

Realtà aumentata

La *realtà aumentata* (AR, dal termine *augmented reality*) è un'esperienza completamente diversa da quella virtuale. Si consideri il mondo reale, così come lo vediamo e lo percepiamo già. A questa visione si supponga di sovrapporre qualche informazione o qualche oggetto virtuale: questa è la realtà aumentata. Si tratta di una vera e propria aggiunta di dati alla realtà esistente e ancora percepibile dall'utente, che realizza una esperienza non completamente immersiva come invece accade con la realtà virtuale. Questa tecnologia si trova in una zona intermedia sulla continuità virtuale, tra il mondo reale e quello artificiale.

È inoltre interessante distinguere il termine di *realtà aumentata* (AR) con quello di *virtualità aumentata* (AV, dall'inglese *augmented virtuality*), proposto nell'articolo di Milgram [2]. Nel secondo caso, infatti, si tratta di una realtà virtuale "aumentata" con informazioni e oggetti reali sotto forma di video. È importante comprendere che esistono molte tecnologie diverse in fase di sviluppo: alcune fondate su video che viene migliorato con computer grafica, mentre altre migliorano un ambiente completamente digitale con video della realtà. Si potrebbero fare ulteriori distinzioni e classificazioni, ma non sono d'interesse per questo lavoro. Si ritiene invece importante ribadire il fatto che tutte queste tecnologie condividono la finalità di giustapporre oggetti reali con oggetti virtuali.

Con l'avanzare della tecnologia risulterà sempre più difficile distinguere la realtà di partenza che viene aumentata digitalmente, rendendo complicata la scelta tra i termini AR e AV. Oggi il termine *realtà aumentata* è usato per indicare solamente le applicazioni in cui il mondo reale viene migliorato con informazioni digitali, con le quali non si può interagire tramite oggetti reali (ad esempio con le proprie mani).

Realtà mista

La *realtà mista* indica la zona centrale della continuità virtuale, in cui esistono elementi di entrambi gli estremi (reale e virtuale). Il termine è stato introdotto nell'articolo di Milgram [2], per eliminare i problemi tassonomici tra realtà aumentata e virtualità aumentata raggruppandole in una categoria più ampia. Oggi il termine indica principalmente le tecnologie di realtà aumentata in cui è possibile l'interazione tra oggetti virtuali e reali.

Ad esempio, l'utente potrebbe vedere un ologramma e con le proprie mani "raggiungerlo" per spostarlo o ingrandirlo. Mentre nella realtà aumentata questo poteva avvenire solo con un controller oppure toccando un display, nella realtà

mista l'interazione avviene con oggetti del mondo reale. È evidente che questa tecnologia richiede una forma di riconoscimento spaziale dell'ambiente, dei gesti dell'utente e della sua voce. L'hardware specifico e l'elevata potenza di calcolo necessari per realizzare i requisiti della realtà mista, giustificano l'implementazione di questa tecnologia, al giorno d'oggi, solamente su visori dedicati.

2.3 Cenni storici

La tecnologia della realtà mista si è evoluta a partire da quella virtuale e da quella aumentata. Il primo dispositivo di interesse è il *Link Trainer*: simulatore di volo utilizzato per addestrare i piloti di molti eserciti durante la seconda guerra mondiale. Si tratta di una cella chiusa simile all'interno di un aereo e mancante la visione del mondo esterno (né reale né virtuale), costruita su un giunto cardanico comandato da pompe e valvole a vuoto. La simulazione era verosimile e prima dell'utilizzo militare venne inserita in molti parchi divertimento come giostra.

Il primo vero dispositivo ottico fu quello brevettato da Morton Heilig nel 1960: la *Telesphere Mask*. Si trattava di un visore a tubi catodici con l'aggiunta di vento (folate d'aria) e odori. Nonostante il geniale intuito dell'inventore e regista statunitense, questo dispositivo fu un enorme fallimento economico. Nel 1962 Heilig creò un secondo dispositivo chiamato *Sensorama*, che aveva le dimensioni di un jukebox e offriva immagini 3D, vibrazioni, vento, sensazioni tattili, audio stereofonico e feedback olfattivi. Durante la progettazione della macchina Heilig registrò cinque brevi film per mostrarne la capacità, ma le prospettive di successo del Sensorama furono rovinate dagli scarsi investimenti da parte delle case di produzione cinematografiche.

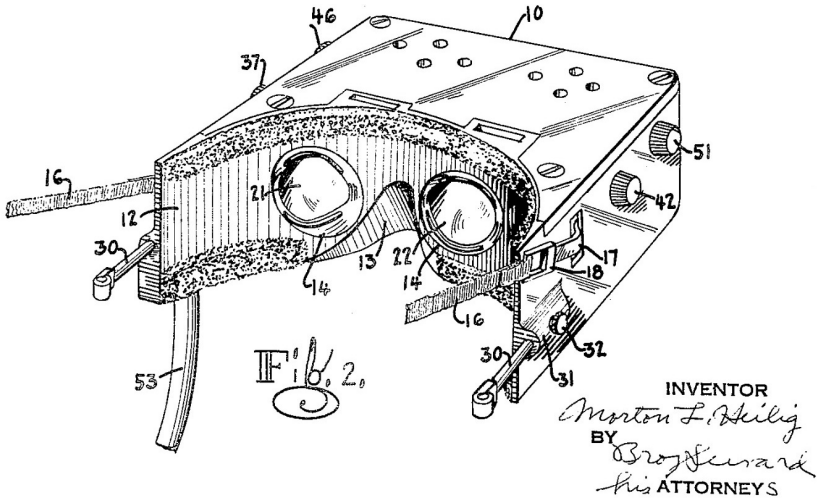

Figura 2.3: Telesphere Mark (Fonte: Google)

Nel 1968 Ivan Sutherland e Bob Sproull fecero un enorme passo avanti nello sviluppo della tecnologia e progettarono il primo visore con immagini processate in tempo reale. Il dispositivo venne chiamato *Spada di Damocle* perché, sospeso da una ingegnosa struttura, ricordava la spada appesa in uno dei racconti tramandati da Cicerone. Questo visore fu il primo a integrare il sincronismo tra movimento della testa e visione dell'ambiente, aprendo nuove frontiere alla tecnologia della realtà aumentata. La potenzialità grafica era limitata, con la possibilità di vedere solo qualche semplice modello di tipo wireframe sovrapposto alla visione dell'ambiente. Si tratta dunque del primo vero dispositivo di realtà aumentata con immagini processate in tempo reale da un computer esterno.

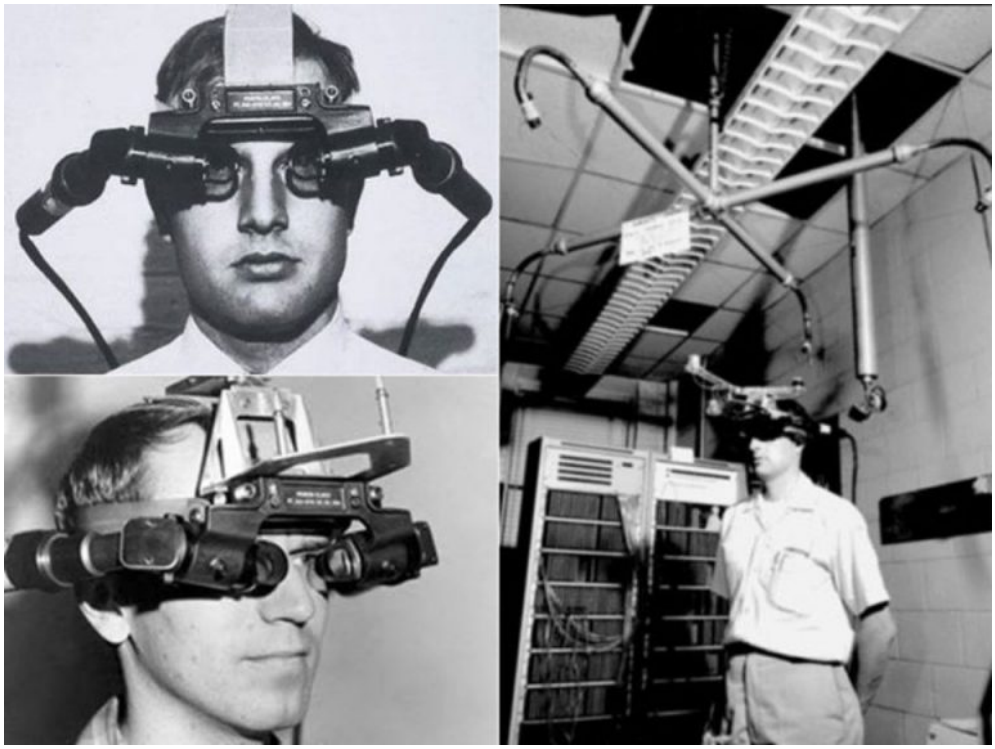

Figura 2.4: Spada di Damocle (Fonte: BBC)

Nel 1974 Myron Krueger fondò un laboratorio dedicato alla realtà artificiale, *Videoplace*, in cui sviluppò importanti tecnologie di interfaccia tra uomo e macchina. Il laboratorio era costituito da videocamere e proiettori e creava un ambiente digitale senza l'utilizzo di visori o guanti. La figura della persona veniva acquisita con una videocamera, che ne proiettava la silhouette affianco a oggetti virtuali con i quali si poteva interagire. Nonostante Krueger sviluppò questo ambiente principalmente con obiettivi artistici, le interfacce che egli sviluppò furono destinate a rimanere in molte installazioni di realtà aumentata.

Negli anni '80 si ricorda l'invenzione di Jaron Lanier in collaborazione con la VPL Research denominata *Eyephone*: si tratta di un sistema di realtà virtuale in cui l'utente poteva interagire con gli oggetti digitali tramite un guanto (*data glove*). Il fatto che questo dispositivo costasse 90.000 dollari (computer escluso) e riuscisse a generare solo 5-6 fotogrammi al secondo, spiega comprensibilmente il fallimento della VPL Research solamente un anno dopo la creazione dell'Eyephone.

Il caso della VPL Research è emblematico della ricerca sulle realtà digitali negli anni '90. Questo periodo, infatti, vide un forte rallentamento della virtualità in seguito alla comprensione di ricercatori e investitori sull'arretratezza della tecnologia impiegata. Dunque l'elettronica degli anni '90 imponeva grossi limiti di calcolo, troppo stringenti per prodotti soddisfacenti e allo stesso tempo accessibili al mercato di massa. Lo sviluppo della virtualità proseguì solamente in due settori: militare (soprattutto USA) e cinema. Secondo Antonio Laudazi [1] nell'ambito militare i motivi furono principalmente i seguenti:

- gli investimenti non erano un vincolo per la ricerca grazie al mercato ricchissimo;
- maggiore interesse nello sviluppo di qualcosa che può essere utile per salvare vite umane, uccidendone altre (quindi non un utilizzo strettamente dilettevole);
- i soldati erano già abituati a indossare un elmetto, dunque integrare quest'ultimo con dell'ulteriore hardware sarebbe stato poco invasivo.

Quest'ultimo punto è particolarmente importante in quanto uno dei freni principali alla diffusione di una nuova tecnologia è la richiesta, nei confronti degli utenti, di utilizzare nuovo hardware che normalmente non porterebbero con se.

Il cinema fu l'altro settore in cui la ricerca sulla virtualità non si fermò. Molti furono i film che trattavano il tema della realtà virtuale (da *Il tagliaerbe* di Brett Leonard a *Matrix* delle sorelle Wachowski), spesso incentrati sugli aspetti più ignoti e inquietanti della tecnologia, facendo leva sull'immaginazione e sulla paura delle persone. Negli anni 2000 venne prodotto qualche altro film sulla virtualità, e la realtà aumentata iniziò a uscire dai laboratori e ad animare alcune applicazioni. Tra queste ricordiamo il sistema di ripresa aerea *Skycam* per gli eventi sportivi negli stadi, che aveva la possibilità di sovrapporre effetti grafici (ad esempio la linea di fuorigioco nel calcio) sul terreno di gioco. In contemporanea si iniziarono

a sviluppare anche videogiochi in realtà virtuale, ottenendo però i primi risultati apprezzabili e accessibili solo dopo il 2010.

Infine, è solo negli ultimi 6 anni che la tecnologia ha permesso ad alcuni produttori di registrare cortometraggi e brevi film in VR. Questa può essere interpretata come la conseguenza del continuo tentativo, da parte dei registi, di avvicinare gli spettatori e immergerli nell'esperienza audiovisiva. Nonostante ciò, la prospettiva di vedere un film di durata normale in VR è molto bassa: la tecnologia VR deve fare prima i conti con la sensazione di nausea che provocano i visori dopo pochi minuti di utilizzo.

2.4 Effetti negativi sulla salute

Le realtà virtuale e aumentata rappresentano un importante strumento per addestramenti ad attività stressanti (cioè situazioni a elevato rischio e procedure mediche) oltre a impieghi di assistenza e controllo a distanza [3]. I dispositivi VR e AR, tuttavia, non sono ancora perfetti e molti utenti riferiscono sensazioni di nausea dopo averli utilizzati, anche solo per pochi minuti. Viste le potenzialità e le molteplici applicazioni di questa tecnologia, è di interesse comune comprendere i limiti e difetti fisiologici che attualmente ne limitano la diffusione. In questo paragrafo si analizzano gli effetti negativi sulla salute della realtà virtuale e aumentata, avviando un confronto e una riflessione sulle due tecnologie.

I visori VR sono, tra le varie tecnologie di virtualizzazione, quelli che maggiormente provocano un senso di nausea agli utenti. Altre tecnologie, come i dispositivi AR (visori e non) e altri dispositivi VR come i simulatori, possono provocare effetti negativi sulla salute degli utenti. Tuttavia questi sintomi sono tipicamente minori sia in numero che tipologia rispetto a quelli provocati dai visori VR, i quali saranno dunque il riferimento per una trattazione più generica possibile dell'argomento. La suddivisione delle cause del malessere proposta in questa tesi è quella fornita da [4].

2.4.1 Cinetosi

La cinetosi (oppure *motion sickness* in inglese) è un complesso di sintomi correlati allo spostamento reale o virtuale della persona nello spazio. Il malessere si presenta quando il cervello riceve segnali afferenti contrastanti sulla posizione e il movimento del corpo. I sintomi possono essere nausea, stanchezza, disagio generale, vertigini, mal di testa, disorientamento, pallore, sudore e nel peggio-

re dei casi vomito. La cinetosi indotta visivamente è dovuta solo alla visione del movimento, mentre quella indotta fisicamente è conseguenza del movimento fisico. La prima può essere limitata semplicemente chiudendo gli occhi, mentre la seconda no. Alcuni esempi comuni di cinetosi sono il mal di mare e mal d'auto.

Esistono diverse teorie che spiegano perché la motion sickness accade [4] e la più diffusa è la *Sensory Conflict Theory*, secondo la quale l'utente può percepire nausea quando il suo cervello riceve informazioni contrastanti sull'ambiente. Nell'ambito della realtà virtuale i segnali afferenti più significativi sono quelli visivi e quelli provenienti dal sistema vestibolare. In effetti, la diversa natura di queste informazioni (i primi provengono esclusivamente dalla simulazione mentre i secondi dal mondo reale) non può garantirne la reciproca coerenza.

La *Evolutionary Theory*, invece, spiega perché il nostro corpo genera la sensazione di nausea. Secondo questa teoria l'inconsistenza dei segnali può essere interpretata dal cervello come una forma di avvelenamento o intossicazione, che crea la sensazione di malessere ed eventualmente la risposta emetica.

La *Postural Instability Theory* approccia la questione diversamente e sostiene che alcuni degli obiettivi principali degli animali siano la stabilità posturale e l'equilibrio, senza i quali il corpo crea sensazioni di nausea. Secondo questa teoria, l'instabilità posturale precede i sintomi di malessere e ne è la causa. Si pensi, ad esempio, che quando la visione dell'utente si sposta virtualmente esso tende ad inclinarsi nella direzione del movimento in maniera istintiva, diventando meno stabile. In questo caso vengono stimolati dei muscoli a causa del video che si sta visualizzando e non della vera e propria necessità del corpo. La Postural Instability Theory sostiene che siano questi piccoli movimenti a generare instabilità e successivamente malessere. Esistono altre interessanti teorie per le quali si rimanda alla lettura di [4].

In generale, si può riassumere la percezione del corpo umano con lo schema di Figura 2.5.

Senza fare riferimento alla realtà virtuale, è lo stato della persona nel mondo a fornire le informazioni uditive, visive, vestibolari, propriocettive e tattili al cervello. Essendo ogni senso umano in qualche modo limitato, è necessario trasmettere segnali di diversa natura al cervello per generare una stima univoca e precisa dello stato della persona nell'ambiente. Infatti, la mente, grazie alle informazioni sensoriali, e a una serie di "strumenti" come le aspettative e la memoria, genera una stima della persona nell'ambiente e aggiorna il suo modello mentale

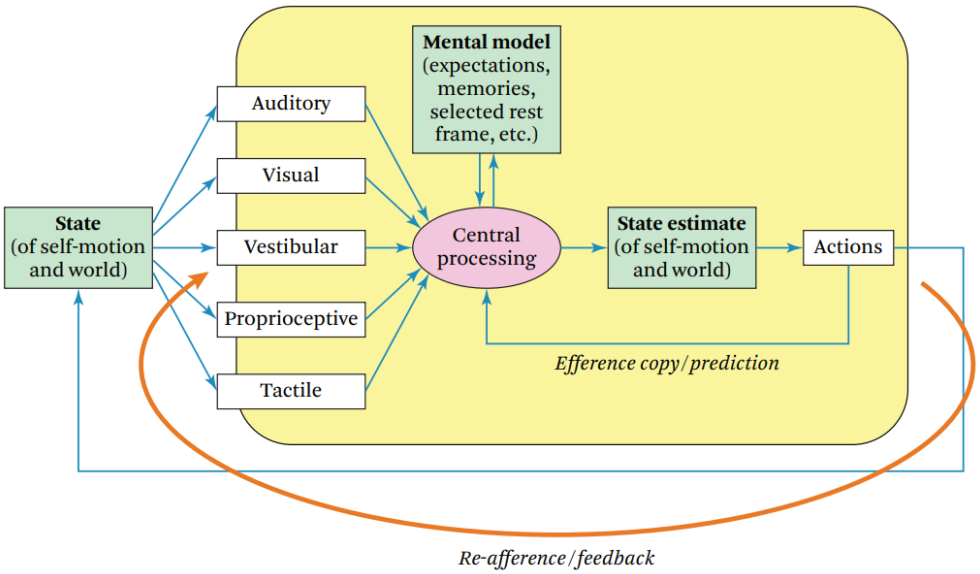

Figura 2.5: Modello unificato della percezione del movimento e del malessere (Fonte: Jason Jerald, *The VR Book* [4])

di come funziona il mondo. In base alla stima la persona effettua delle azioni, e contemporaneamente viene innescato un interessante meccanismo indispensabile per la concezione della realtà stabile[2].

Quando, ad esempio, vengono generati stimoli per un movimento della testa, copie di questi stimoli efferenti sono inviate alla zona del cervello responsabile della creazione del modello mentale. Nel frattempo questa particolare zona crea un insieme di aspettative sui segnali afferenti attesi in corrispondenza delle azioni volontarie che il corpo sta effettuando. Nel caso di rotazione della testa, il cervello potrebbe aspettarsi un impulso vestibolare di rotazione e di vedere la visione traslare in una determinata direzione. Questa retroazione diretta tra azione e cervello, senza passare per gli organi sensoriali, può generare malessere nel caso di informazioni afferenti e aspettative contrastanti. Ad esempio, questo meccanismo spiega perché è molto più raro che gli autisti delle automobili soffrano di cinetosi rispetto ai passeggeri della stessa vettura.

Dovrebbe risultare chiaro che, nel caso in cui qualcosa dovesse andare storto nel modello di Figura 2.5 a causa dell'inserimento di dispositivi artificiali, la sensazione di benessere della persona potrebbe essere compromessa.

[2]La realtà "visivamente stabile" è fondata sulla relazione tra azione dell'individuo e aspettative di cambiamento delle informazioni sensoriali coerentemente con il movimento effettuato.

2.4.2 Disturbi ottici

Tra le altre cause di malessere dovuto all'utilizzo di visori VR è importante segnalare quella relativa al *conflitto di accomodazione-vergenza* (traduzione dall'inglese di *accomodation-vergence conflict*), rappresentata in Figura 2.6. L'accomodazione è la funzione del cristallino dell'occhio che permette di mettere a fuoco oggetti a diverse distanze. La vergenza, invece, è la simultanea rotazione degli occhi in direzioni opposte per "centrare" l'oggetto che si sta mettendo a fuoco. Ad esempio, quando una persona guarda un oggetto posto a pochi centimetri dal suo volto, la vergenza farà ruotare i suoi occhi in maniera opposta e tenderà a farli incrociare leggermente. L'accomodazione e la vergenza sono due attività che normalmente vengono effettuate in maniera sincrona e spontanea. Tuttavia, quando si utilizzano dei visori per la realtà virtuale, il cervello è costretto a mantenere una messa a fuoco fissa (perché lo schermo è a distanza fissa dagli occhi) e a far convergere gli occhi sui vari oggetti[3]. L'artificiale relazione tra convergenza e accomodazione durante l'utilizzo di visori VR comporta sensazioni di fatica agli occhi, ed è un problema che la ricerca nell'ambito delle fotocamere plenottiche [17] potrebbe risolvere nei prossimi anni.

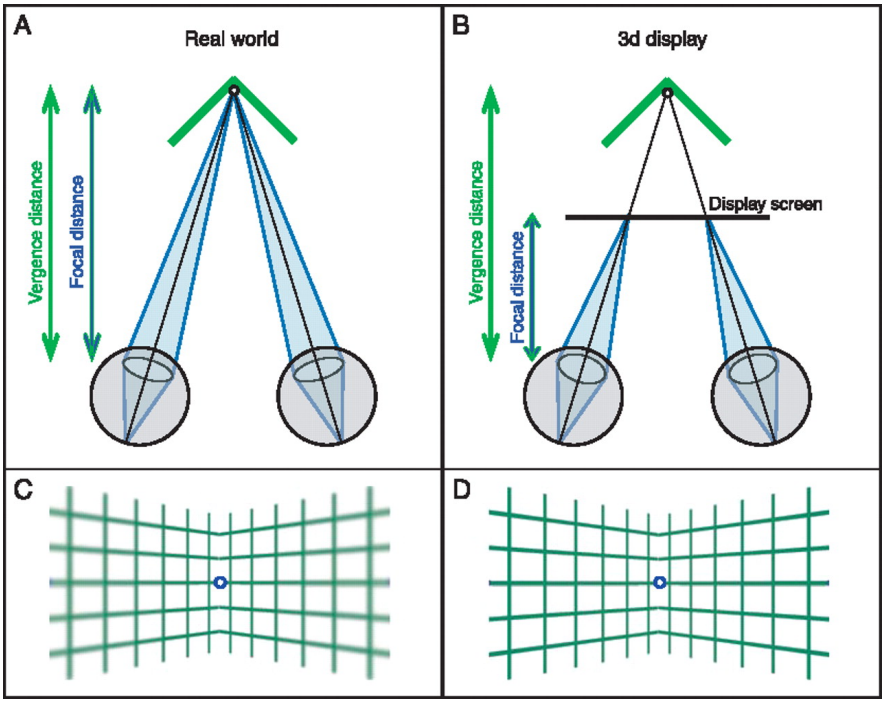

Figura 2.6: Conflitto di accomodazione-vergenza (Fonte: David M. Hoffman, *Vergence-accommodation conflicts hinder visual performance and cause visual fatigue* [5])

[3]Si ricorda che nei visori moderni la vista rimane binoculare.

Un ulteriore disturbo ottico dei visori è quello provocato dallo "sfarfallio" dello schermo. Come conseguenza dei compromessi tra risoluzione e tempi di latenza, dovuto allo stato attuale della tecnologia, lo sfarfallio può diventare più evidente. Questo fastidioso fenomeno porta all'affaticamento degli occhi, alla nausea, al mal di testa e al panico.

Infine, è importante sottolineare che esistono problemi che sussistono anche dopo l'utilizzo dei visori VR. Gli effetti postumi comprendono disorientamento, instabilità percettiva del mondo, flashback e colpiscono circa il 10% delle persone per una o due ore dopo l'utilizzo del dispositivo [4]. È quindi importante che le persone che utilizzano i visori per la prima volta vengano avvisate degli effetti postumi, invitandole a evitare di guidare per almeno 30-45 minuti dall'attività.

2.4.3 Altri difetti da tenere in considerazione

La tecnologia VR presenta altri problemi salutari per gli utilizzatori. In particolare, ogni forma di latenza del dispositivo è responsabile di un peggioramento dell'esperienza e di una più alta probabilità di nausea. Le fonti di latenza possono essere di natura diversa [4]:

- ritardi dai sensori di movimento;
- ritardi software di applicazione;
- ritardi di rendering;
- ritardi del display;
- ritardi di sincronizzazione.

Si suppone che i ritardi possano essere migliorati con l'avanzare della tecnologia.

Infine, esistono problematiche relative l'ergonomia dell'utilizzo dei visori. Le più significative sono [4]:

- la fatica fisica (dovuta al peso e al centro di massa del dispositivo);
- la comodità del visore;
- il rischio di infortunio (scontro con oggetti reali non visibili con il visore);
- il rischio di infiammazioni (anche dovute all'utilizzo di controller);

- il rischio di danneggiamento dell'udito;
- problemi di igiene quando più persone utilizzano lo stesso visore.

L'obiettivo della comunità di ricercatori e investitori nel settore VR è quello di affrontare questi problemi, dal più banale al più significativo, per migliorare la diffusione di questa tecnologia.

2.4.4 Confronto tra VR e AR

I visori di realtà aumentata condividono molte caratteristiche con quelli VR. Infatti tutte le problematiche relative all'ergonomia (tranne il rischio di infiammazioni delle mani e il rischio di infortunio) sono condivise tra le due tecnologie. Secondo uno studio della Oxford Brookes University [6], effettuato su un visore *HoloLens*, i test sulla realtà aumentata non hanno provocato sintomi di malessere a quasi nessuno dei partecipanti. Questo risultato potrebbe essere giustificato dalla presenza del mondo reale nello sfondo. Esso, infatti, fornisce all'utente segnali autentici per l'orientamento nello spazio, eliminando così tutte le imperfezioni di un mondo virtuale. L'utente ha a disposizione riferimenti reali grazie ai quali i sintomi di nausea e malessere dovrebbero essere attenuati. Nonostante ciò, alcuni sintomi legati al nervo oculomotore continuarono a manifestarsi tra i partecipanti dell'esperimento. In particolare si tratta di affaticamento degli occhi e mal di testa[4]. Le conclusioni dello studio sostengono che le interazioni di realtà aumentata hanno meno probabilità di generare le sensazioni di nausea gastrointestinale tipiche dei dispositivi VR. I disturbi visivi invece, come l'affaticamento degli occhi, possono manifestarsi in ogni caso.

[4]Il mal di testa, nella valutazione del SSQ (Simulator Sickness Questonnaire) è considerato come conseguenza di stimoli visivi.

Capitolo 3

Applicazioni e analisi di mercato

In questo capitolo si parlerà di realtà aumentata in quanto è il termine più comune con cui ci si riferisce a questa tecnologia e spesso viene confuso con le altre due realtà immersive. Nulla toglie alla realtà mista che, come spiegato nel capitolo precedente, raggruppa sotto un unico termine la realtà virtuale e la realtà aumentata, ovvero la tecnologia che permette di interagire con gli ologrammi.

Si propone un approfondimento sulla tematica dell'industria relativa alla realtà aumentata e dei suoi ambiti applicativi, passando per un elenco dei dispositivi in commercio e infine proponendo dei possibili scenari futuri di applicazione di questa tecnologia.

3.1 L'industria della realtà aumentata

La realtà aumentata (AR) è una tecnologia che sta prendendo sempre più piede nella vita di tutti i giorni. Lo si può capire dalle numerose app per smartphone che consentono di aggiungere funzioni al dispositivo personale. Tra esse spiccano la possibilità di misurare le dimensioni di oggetti o stanze come l'app ufficiale Measure [18] di Apple per iPhone, la capacità di visualizzare un'anteprima del prodotto che si andrà ad acquistare come un mobile di Ikea attraverso l'app Ikea Place [19], un nuovo paio di occhiali [20] oppure l'ultimo modello di scarpe alla moda [21], creando quindi un nuovo modo di fare shopping.

Non ultimi per importanza, i videogiochi in AR che nel settore mobile sono sempre più diffusi e la cui punta di diamante è rappresentata dall'enorme successo del gioco Pokémon GO [22].

Anche gli head-up display nelle automobili sono una forma di realtà aumentata. Grazie a un proiettore posto nel cruscotto è possibile visualizzare sul para-

brezza, direttamente davanti agli occhi del guidatore, delle informazioni quali la velocità e le linee della carreggiata [23].

In ambito industriale la situazione è invece un po' diversa. Se il settore commerciale ad uso civile è principalmente dominato da software di Augmented Reality scaricabili sul proprio smartphone oppure dai visori di realtà virtuale per i videogiochi, il settore industriale è rappresentato soprattutto dai visori in realtà aumentata o in realtà mista. I visori AR, VR, o MR sono dispositivi indossabili simili a un ibrido tra un paio di occhiali e un caschetto, con la possibilità di interagire attraverso dei joystick o attraverso il movimento delle proprie mani.

Questa tecnologia ha trovato molte applicazioni perché aiuta i lavoratori a svolgere soprattutto i compiti più difficili; infatti, le aziende negli ultimi anni hanno investito sempre più fondi sulla tecnologia immersiva.

Gli ambiti di applicazione della realtà aumentata sono molto vari. Il suo utilizzo spazia dall'industria allo studio dei prototipi, dai cantieri edili alle sale operatorie degli ospedali, dalla didattica del futuro al settore militare fino allo shopping virtuale.

Un esempio di applicazione industriale è l'adozione di questa tecnologia da parte di Boeing e Airbus[5] che forniscono ai propri lavoratori i visori, come HoloLens di Microsoft, per poter consultare documentazione, siti web e addirittura gli schemi degli impianti dei loro prodotti, il tutto direttamente sul luogo di assemblaggio e lasciando le mani dell'operatore libere [24, 25].

Sempre sullo stesso filone, anche thyssenkrupp ha fornito ai suoi operai i visori HoloLens per agevolare e velocizzare le operazioni di manutenzione nei suoi ascensori, come ad esempio nel One World Trade Center a New York [26].

Anche la NASA ha impiegato HoloLens per addestrare i propri astronauti e aiutarli a svolgere compiti complessi, in cui c'è la necessità di avere le mani libere per operare e allo stesso tempo di leggere della documentazione. Il progetto si chiama *Sidekick* e nel 2015 i visori di Microsoft sono stati portati a bordo della ISS per fornire assistenza agli astronauti [27].

Sempre la stessa NASA sta studiando un software per il monitoraggio del traffico aereo generato dai droni, il tutto utilizzando visori in realtà aumentata attualmente in commercio [28].

[5]Note multinazionali leader nella produzione di aerei, razzi e satelliti.

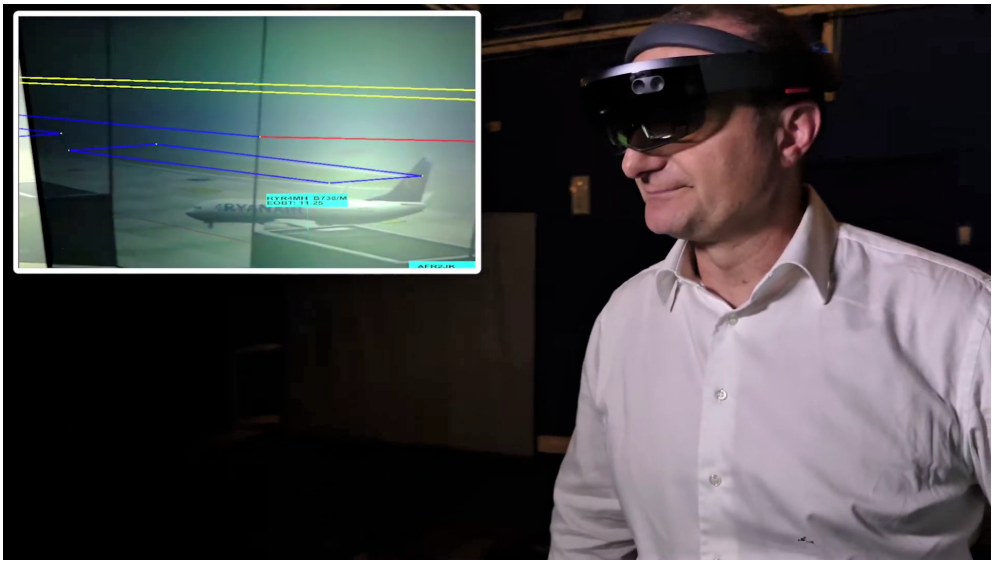

Figura 3.1: Progetto RETINA (Fonte: EUROCONTROL)

Restando nel settore aerospaziale, un progetto molto interessante sviluppato dall'Università di Bologna [29] in collaborazione con EUROCONTROL e all'interno del più ampio progetto SESAR (Single European Sky ATM Research), è chiamato *Project RETINA* (Resilient Synthetic Vision for Advanced Control Tower Air Navigation Service Provision) [30]. Lo scopo è quello di migliorare il controllo del traffico aereo negli aeroporti assistendo i controllori di volo a svolgere questo compito. Una delle problematiche di questo lavoro si ha quando le condizioni atmosferiche non permettono una buona visibilità delle piste dalla torre di controllo. I visori di realtà aumentata come HoloLens vengono in aiuto in queste situazioni. Con essi l'operatore può stare davanti alle vetrate monitorando la situazione delle piste nella realtà, al posto di guardare i monitor dei computer, e con il visore può visualizzare le informazioni sugli aerei, sui voli e sulle piste di atterraggio.

Un altro progetto, però questa volta più ambizioso, è stato pensato da Microsoft che vuole utilizzare la tecnologia dei suoi visori HoloLens per poter mappare tutto il mondo [31]. In particolare, è stato depositato un brevetto [32] dove l'azienda spiega la volontà di scansionare gli edifici e i luoghi visitati dagli utenti degli HoloLens grazie all'utilizzo dei sensori montati, che sono accurati, e del cloud Azure molto efficiente. Si verrebbe così a creare un'infrastruttura online composta da una quantità molto elevata di modelli 3D scansionati. Questo può essere uno scenario abbastanza preoccupante se si pensa in termini di privacy,

anche se per il momento il problema non si pone perché il principale utilizzo dei visori HoloLens è confinato all'ambito aziendale.

Un ruolo fondamentale dell'utilizzo dei visori in realtà aumentata si ha nel campo dello studio dei prototipi. Infatti, per le aziende la fase di prototipazione di un nuovo prodotto è molto importante per capire se il risultato finale sarà quello desiderato. I prototipi in genere sono molto costosi sia in termini di tempo sia di denaro. Con l'attuale potenza computazionale dei computer è possibile fare rendering 3D anche in alta risoluzione, però il prodotto resta confinato all'interno di uno schermo bidimensionale. Quindi spesso vengono realizzati dei mockup con materiali poco costosi come legno o plastica, anche grazie all'uso delle stampanti 3D, ed è necessario attendere finché non vengano creati internamente all'azienda o addirittura da aziende terze. È proprio qui che i visori esprimono tutte le loro potenzialità, perché grazie ad essi è possibile visualizzare in 3D il prototipo con i materiali e i colori scelti. Inoltre è possibile apporre modifiche in real-time abbattendo di fatto costi e tempistiche.

Anche la realtà virtuale è adatta a questo utilizzo perché svolge le stesse funzioni, però entrando in un mondo virtuale si perdono i riferimenti delle dimensioni spaziali reali che sono un parametro molto importante nello studio dei prototipi.

Un esempio di come la realtà aumentata aiuti nella fase di progettazione è dato dal sistema *LayAR* (Layout Augmented Reality) di Audi. I pianificatori logistici della casa automobilistica possono, attraverso l'uso di visori, visualizzare un prototipo delle loro nuove strutture logistiche da inserire nei capannoni [33].

La Augmented Reality viene utilizzata anche nei cantieri edili dove architetti e operai hanno bisogno di sapere le dimensioni degli edifici o delle stanze [34]. Al posto di tenere in mano ingombranti fogli degli schemi oppure dovendo utilizzare un tavolo per appoggiarli, grazie ai visori in realtà aumentata è possibile visualizzare queste informazioni direttamente sul luogo di interesse lasciando le mani dell'operatore libere. È possibile inoltre apporre modifiche in tempo reale ai parametri riducendo i tempi di progettazione. Un esempio di visore adibito per questa funzione è la versione speciale di *HoloLens 2 Trimble XR10* [35] che integra un elmetto protettivo al quale è agganciato il visore. Il risultato è un unico casco molto facile da indossare e con tutta la tecnologia della realtà mista.

La realtà aumentata trova impiego anche in campo medico, soprattutto nelle

sale operatorie, dove è importante che i medici abbiano le mani libere per operare e nel contempo visualizzare informazioni sul paziente [36]. La stessa Microsoft, nella presentazione di HoloLens, ha sfruttato questo esempio per descrivere l'eterogeneità di applicazione del suo prodotto. Esistono inoltre altri progetti, supportati da visori diversi da HoloLens e creati per essere utilizzati in campo medico, tra cui *NextAR* [37] [38] e *ProjectDR* [39]; oltre al progetto *AccuVein* [40] che consente, con dispositivi proprietari, di visualizzare il sistema vascolare di un paziente sopra la superficie della pelle, di fatto poco invasivo.

Con i visori, che sono dotati di telecamere, è inoltre possibile far operare i chirurghi da remoto collaborando con un medico locale anche lui provvisto di visore, attraverso una semplice videochiamata. Questo implicherebbe un notevole risparmio di tempo in quanto il chirurgo specializzato non dovrebbe recarsi fisicamente nel luogo dell'operazione. Ciò permette, se quest'ultima è urgente, di salvare un maggior numero di vite perché si riescono a ridurre le tempistiche di preparazione.

I visori permettono anche di ottenere ottimi risultati nel campo terapeutico e riabilitativo, perché la simulazione di situazioni reali attraverso elementi tridimensionali stimola i sensi del paziente. Contemporaneamente è possibile monitorare i parametri della persona grazie agli appositi sensori collegabili. Un esempio di questa applicazione è stata creata dall'utente Roguish [41] che ha pubblicato un plugin per HoloLens con il quale è possibile monitorare la frequenza cardiaca attraverso un sensore Bluetooth.

Sono stati fatti studi anche su come la tecnologia immersiva può aiutare le persone con disabilità motorie o soggette al morbo di Alzheimer.

In particolare, la collaborazione tra l'Università delle Forze Armate ESPE e l'Università di Castilla, ha dato vita al progetto *Holonote* [7] ovvero un editor di testo per HoloLens studiato appositamente per persone con disabilità motorie. L'applicazione ha la funzione di semplice editor di testo ma quello che cambia sono i metodi di input: al posto di utilizzare mouse e tastiera o touch screen, l'inserimento del testo avviene tramite sguardo, riconoscimento vocale o riconoscimento dei gesti. Questo faciliterebbe l'usabilità e l'accessibilità soprattutto da parte di persone che hanno difficoltà a coordinare braccia e/o mani. Secondo lo studio, il progetto è stato accolto positivamente dagli utilizzatori e ha anche stimolato l'apprendimento implicito.

Invece il progetto *MemHolo* [8], del Politecnico di Milano, consiste nel far utilizzare il visore di Microsoft a persone con forme lievi di Alzheimer dove si

può ancora mitigare l'effetto del declino mentale. L'applicazione mostra degli ologrammi di oggetti reali e gli utenti devo completare dei compiti semplici forniti dai supervisori. Questa funzione permette di allenare la memoria spaziale e quella a breve termine. Tra i risultati dello studio, si può leggere che i soggetti dei test, principalmente persone anziane, hanno distinto facilmente gli oggetti reali dagli oggetti virtuali, con lo stupore dei terapisti i quali pensavano ci potessero essere difficoltà. Anche questo progetto è stato recepito positivamente dall'utenza e ne è stato rinnovato l'utilizzo.

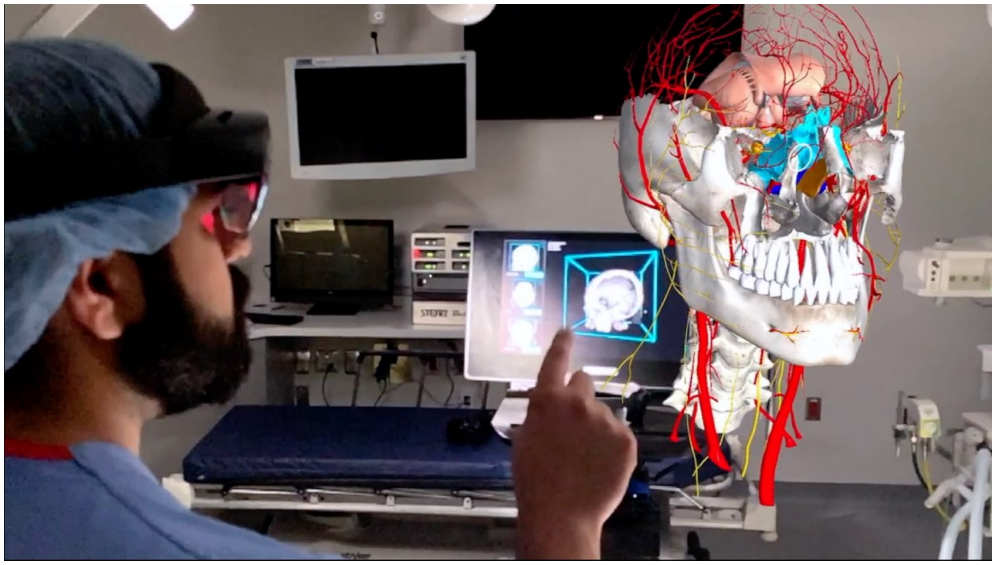

Figura 3.2: AnatomyX di Medivis (Fonte: Microsoft)

Un altro settore molto importante, dove può essere sfruttata al massimo la potenzialità della realtà aumentata, è quello della didattica perché gli studenti possono interagire con gli ologrammi. Uno dei limiti principali dell'istruzione scolastica sono i libri che, per quanto accurati e ben scritti, contengono le nozioni solo sotto forma di testo o al più sotto forma di immagini. Spesso questo limite non permette agli studenti, soprattutto quelli più giovani, di comprendere appieno l'argomento studiato. Per HoloLens esistono già delle applicazioni che permettono di interagire, ad esempio, con un motore a combustione e tutti i suoi componenti oppure con un modello di corpo umano dove si vedono tutti gli organi e lo studente può capirne la forma [42].

In un recente studio [9] è stato dimostrato che gli studenti sono più coinvolti nell'insegnamento perché non sono ricevitori passivi ma attivi, grazie all'interazione che permette la Mixed Reality. Inoltre anche i docenti hanno percepito questa tecnologia positivamente e la ritengono molto utile per migliorare l'insegnamento.

Uno scenario interessante che può coinvolgere anche il mondo della didattica, è quello presentato da Microsoft con il progetto *Holoportation* [43]. In sostanza si può dire che l'azienda americana vuole sostituire le videoconferenze con gli ologrammi delle persone reali. Grazie a un set di telecamere, per il momento soluzione poco pratica, è possibile scansionare una persona e trasmettere il flusso video ai visori HoloLens dei colleghi che possono trovarsi in aree geografiche differenti. Con la realtà aumentata dei visori, è possibile quindi visualizzare l'ologramma della persona in tempo reale come se fosse una video-riunione tradizionale. Questa applicazione permette maggiore immersività nelle conversazioni perché l'ologramma trasmette anche i movimenti della persona, quindi non è un'immagine statica. Inoltre con il sistema *Mesh* [44], sempre di Microsoft e grazie all'ausilio del suo cloud Azure, è possibile anche collaborare manipolando in tempo reale gli ologrammi che risultano condivisi tra i vari visori; proprio come se ci si trovasse realmente con i colleghi sul posto di lavoro e con gli oggetti davanti. Il progetto presentato è solo uno dei possibili utilizzi, perché potrebbe coinvolgere anche didattica a distanza, allenamenti sportivi, telemedicina, convegni e concerti.

La realtà aumentata viene utilizzata anche in ambito militare. In particolare Microsoft ha firmato un accordo con l'esercito americano per fornire dei visori HoloLens da utilizzare in addestramento. I soldati possono così visualizzare strumenti utili come una mappa e una bussola. I visori forniti non sono quelli standard acquistabili ma l'azienda ne ha prodotto una versione speciale che va ad inserirsi nel programma IVAS (Integrated Visual Augmentation System) [45].

Infine, un'applicazione lato consumer è quella di Natuzzi che fornisce in alcuni negozi i visori HoloLens. Questo permette ai propri clienti di visualizzare ogni singolo prodotto del catalogo per vedere se si adatta meglio al proprio salone e gli permette anche di personalizzarlo sul momento con i materiali e colori che preferisce [46].

L'industria ha dimostrato un interesse sempre maggiore verso la tecnologia immersiva, perché permette di migliorare i processi produttivi e aumentare il business. Collaborando da remoto, attraverso i visori, si riducono drasticamente i tempi di esecuzione oltre ad abbattere le spese per le trasferte, ciò incide anche nella riduzione dell'inquinamento pensando ad esempio ai viaggi di lavoro per mezzo aereo [47].

3.2 Dispositivi in commercio

Come si può dedurre dal precedente paragrafo, il visore HoloLens è quello che attualmente trova maggiori applicazioni nell'industria, ma non è l'unico in commercio. Il suo successo è dovuto principalmente alla sua caratteristica di essere un visore per la realtà mista, quindi l'utente può interagire con gli ologrammi utilizzando oggetti reali, come le sue mani. Inoltre è stato presentato subito come un dispositivo da utilizzare in ambito industriale, mentre altri competitor hanno provato a vendere i loro visori anche per un uso civile, ad esempio i Google Glass. Questi ultimi non si sono affermati e alcuni progetti concorrenti sono stati addirittura cancellati: il motivo risiede soprattutto nella tecnologia ancora un po' acerba e nel costo piuttosto elevato per un dispositivo consumer [48].

Attualmente in commercio si trovano molti dispositivi per la realtà aumentata, ma non tutti possono svolgere le stesse funzioni.

Il più famoso è il visore *HoloLens* di Microsoft che per definizione è un visore di realtà mista. È stata prodotta la prima versione nel 2016 e successivamente una seconda nel 2019, la quale ha portato con se molte migliorie tecniche (vedasi paragrafo 4.5 per un approfondimento). Della seconda versione ne sono stati prodotti diversi modelli tra cui uno standard, uno per uso militare, un altro per uso industriale in ambienti sterili e/o deflagranti che è più isolato e rispetta delle norme di sicurezza specifiche, e infine quello per un uso nei cantieri dotato di elmetto protettivo, qui raffigurato in Figura 3.5.

Il costo è abbastanza elevato ma in ambito industriale ciò può essere poco rilevante se permette di migliorare la produttività.

Figura 3.3: HoloLens 1 (Fonte: Microsoft)

Figura 3.4: HoloLens 2 (Fonte: Microsoft)

Figura 3.5: Trimble XR10 con HoloLens 2 (Fonte: Microsoft)

Anche Google ha provato a fare un visore per la realtà aumentata ma ha avuto poco successo. Il prodotto della casa americana prende il nome di *Google Glass* e assomiglia più a un paio di occhiali che a un vero e proprio visore. L'idea nasce con lo scopo di mostrare informazioni come meteo, orologio e notifiche sul campo visivo dell'utente andando a sostituire lo smartphone.

Recentemente ne è stata prodotta una seconda versione più indirizzata all'uso industriale che a quello consumer: Enterprise Edition 2 è dotato di una paratia laterale per proteggere gli occhi dei lavoratori.

Figura 3.6: Google Glass Enterprise Edition 2 (Fonte: Google)

Cercando di cogliere l'onda dei Google Glass, l'azienda Vuzix ha prodotto i suoi occhiali smart *Vuzix Blade* che hanno le stesse funzioni e scopi dei sopracitati. Esiste anche un'altra versione dei loro occhiali concepita per un uso industriale. I *Vuzix M400* perdono in estetica ma integrano maggiore praticità e comodità per un utilizzo anche con il casco protettivo.

Figura 3.7: Vuzix Blade (Fonte: Vuzix)

Figura 3.8: Vuzix M400 (Fonte: Vuzix)

Un prodotto che invece ha fatto molto scalpore è il visore *Magic Leap One*. La startup americana è riuscita a convincere gli investitori, tra cui Google a AT&T, grazie a un'ottima campagna pubblicitaria [49]. Purtroppo però il prodotto ha deluso le aspettative e, anche a causa del suo costo elevato, il visore ha avuto meno vendite del previsto. Successivamente la società è stata venduta [50].

Figura 3.9: Magic Leap One (Fonte: Magic Leap)

Altri dispositivi meno conosciuti sono *Kiber 3* e *ThirdEye X2*, per quanto riguarda i visori AR che trovano applicazione nel campo industriale.

Figura 3.10: Kiber 3 (Fonte: Kiber)

Figura 3.11: ThirdEye X2 (Fonte: ThirdEye)

Per lo stesso settore c'è anche *DAQRI Smart Helmet*: un casco dotato di visore per realtà aumentata che viene utilizzato da costruttori, ingegneri e progettisti [51]. Invece il visore *Meta 2* per Mixed Reality è molto simile a HoloLens e trova svariate applicazioni perché permette l'interazione con gli elementi virtuali; il suo svantaggio è quello di doversi collegare ad un computer.

Figura 3.12: DAQRI Smart Helmet (Fonte: Daqri)

Figura 3.13: Meta 2 (Fonte: WearVR)

Per quanto riguarda gli smart glasses AR, più comodi e pratici da utilizzare, esistono *Epson Moverio Bt30c* e il *Toshiba AR100* su base Vuzix M400,

Figura 3.14: Moverio Bt30c (Fonte: Epson)

Figura 3.15: Toshiba AR100 (Fonte: dynabook)

mentre gli occhiali smart *Solos* e *Everysight Raptor* sono destinati al pubblico sportivo che vuole mantenere sott'occhio i parametri più importanti del loro allenamento.

Figura 3.16: Solos smart glasses (Fonte: Solos)

Figura 3.17: Everysight Raptor (Fonte: Everysight)

Purtroppo alcuni di questi progetti sono falliti assieme all'azienda produttrice [52, 53]. Il motivo di questo declino può essere attribuito al costo dei visori perché il prezzo medio si aggira attorno al migliaio di dollari, cifra poco sostenibile per un uso casuale o di svago, ma che invece può essere ammortizzata se affrontata come investimento da un'azienda per aumentare la sua produttività.

Un altro fattore importante che ha condizionato il successo nel mercato di massa dei dispositivi per la realtà aumentata è la comodità e la praticità. Infatti, questi device sono scomodi dopo molte ore di utilizzo e ingombranti con, in alcuni casi, dei cavi da collegare ad un computer. Per questo motivo i visori per la realtà virtuale, anche se più grandi e pesanti di quelli per la realtà aumentata, hanno avuto una maggiore diffusione sul mercato consumer, soprattutto nell'ambito dei videogiochi.

Per elencare alcuni dei più noti visori VR troviamo: *HP Reverb G2*, *Playstation VR*, *Valve Index*, *Pimax 5K*, la serie di *HTC VIVE Pro* e *VIVE Focus* oltre al nuovo *HTC VIVE Flow* e infine i vari *Oculus Quest*, *Rift* e *Go* di Facebook.

Il loro successo è dovuto al minor costo perché sfruttano una tecnologia più semplice rispetto ai visori per la realtà aumentata, cioè due piccoli schermi posti molto vicino agli occhi che permettono di immergersi in un mondo virtuale. Il principale svantaggio, su alcuni modelli, è la necessità di collegarli ad un computer o a una console e di dover disporre di joystick per i comandi.

Infine esiste anche un visore ibrido che integra sia la realtà aumentata sia la realtà virtuale. *Lynx R-1* è il visore della omonima startup francese che permette di vedere in realtà virtuale una parte della stanza in cui ci si trova e poi, spostandosi, vedere un'altra parte della stanza in realtà aumentata, quest'ultima visualizzabile mediante le apposite fotocamere esterne che trasmettono le immagini agli schermi interni. Inoltre è possibile interagire con gli ologrammi visualizzati attraverso l'uso delle mani perché è dotato del tracciamento dei movimenti, quindi si può dire che sia anche un visore per la realtà mista [54]. Per ora è un progetto in crowdfunding che fa del costo basso il suo punto di forza.

Figura 3.18: Lynx R-1 (Fonte: Lynx)

3.3 Possibili scenari futuri

In commercio esistono diverse tipologie di visori per la realtà aumentata o virtuale. Purtroppo, il loro attuale limite sta nella potenza di calcolo perché devono renderizzare immagini tridimensionali in tempo reale. In genere sono forniti di una propria CPU e GPU con anche una batteria che li rende dispositivi stand-alone rispetto al computer.

Gli attuali processori montati nei visori rappresentano il collo di bottiglia delle prestazioni di questi device; oppure, se sono molto prestazionali, non garantiscono una buona autonomia perché molto energivori. In futuro, però, è possibile che la tecnologia del 5G venga implementata nei visori. Essa garantisce basse latenze

ed elevate velocità di trasmissione dei dati, anche in luoghi difficili da raggiungere con le precedenti generazioni di rete mobile [55]. Questo permetterebbe la connessione con centri di calcolo cloud molto più performanti di una normale CPU e ciò garantirebbe migliori prestazioni sia grafiche sia in termini di fluidità dell'esperienza, senza gli attuali rallentamenti. Commissionare la potenza di calcolo al cloud, al posto di mantenerla circoscritta all'interno del visore, permetterebbe anche di ottenere dispositivi meno ingombranti e più leggeri, quindi più comodi.

Fin dagli albori, vedasi i Google Glass, i visori sono stati pensati come occhiali tecnologici da poter indossare quotidianamente. Il 5G rappresenterebbe quindi una svolta per questa tecnologia, perché ne favorirebbe la diffusione commerciale grazie alla possibilità di integrare tutta l'elettronica in occhiali smart eleganti, sottili e leggeri [56].

Si pensa che in futuro si potranno eliminare i manuali cartacei e i faldoni, che sovrastano gli ambienti di lavoro, grazie all'utilizzo della tecnologia immersiva, contribuendo così a ridurre l'uso del cartaceo in favore del digitale. Inoltre sarà possibile, ad esempio, costruire macchinari senza pannello di controllo, perché grazie alle tecnologie IoT e ai visori in realtà aumentata, tutte le informazioni e i parametri della macchina verranno visualizzate direttamente davanti agli occhi dell'operatore. Questa tecnologia potrebbe aiutare molto i lavoratori e permetterebbe così di proiettare le aziende verso la cosiddetta industria 4.0 [57].

Nei prossimi anni la realtà aumentata verrà usata sempre più spesso in ambito medico-chirurgico perché offre un aiuto notevole ai medici e permette di effettuare operazioni anche in luoghi difficili da raggiungere [58]. Inoltre, se verranno sviluppate applicazioni apposite, come visto dagli studi nel paragrafo 3.1, i visori potranno essere uno strumento molto utile per le persone con disabilità, come difficoltà motorie o problemi celebrali tra cui l'Alzheimer; permetterebbe loro di completare attività relativamente complesse ma compiendo gesti semplici.

Per concludere, si può immaginare che i visori di realtà aumentata e virtuale un giorno potranno sostituire gli smartphone se non addirittura superarli tecnologicamente. L'obbiettivo primario, ora, è quello di renderli dei dispositivi sempre più indipendenti in modo che possano essere utilizzati maggiormente e con più facilità.

Esistono anche altri progetti per la realtà aumentata non legati a dei visori. È il caso delle lenti a contatto smart della startup Mojo Vision, tramite le quali è possibile visualizzare informazioni digitali. Hanno ottenuto finanziamenti dalla

Stanford University, da Google e Motorola, ma per ora sono dei prototipi non ancora messi in commercio. Forse tra qualche anno la tecnologia sarà pronta e allora cambierà l'approccio verso la realtà aumentata [59].

Infine, i dispositivi immersivi un giorno potrebbero essere controllati anche con il pensiero [60]. Il progetto di Facebook è già in corso ed è stato pubblicato un documento [10] dove si è annunciato che il team è riuscito a decodificare un piccolo insieme di parole e frasi usando l'attività celebrale.

Capitolo 4

Il visore HoloLens 2

Il visore a realtà aumentata HoloLens 2 viene definito dalla stessa casa produttrice come: "un computer olografico indipendente" [61] che fornisce un'esperienza immersiva all'utente. L'obiettivo di gran parte delle applicazioni sviluppate per questo device è assistere l'utilizzatore in mansioni complicate e garantire un'interazione tra il mondo fisico e il mondo digitale. Sarà interessante analizzare dapprima le sue specifiche e le potenzialità di utilizzo e successivamente scendere nei particolari tecnici quali la sensoristica di bordo, il display e l'interazione con l'ambiente circostante. Infine sarà importante la prova in laboratorio, per poter testare i suoi limiti e l'effettiva applicabilità nell'uso quotidiano o in ambito lavorativo.

4.1 Specifiche e potenzialità

Tra le funzionalità più interessanti dichiarate dalla casa produttrice spiccano [62]:

- possibilità di ancorare in modo preciso e persistente gli ologrammi nel mondo reale;
- tracciamento delle mani, è possibile interagire con gli ologrammi modificandoli, spostandoli ingrandendoli ecc. . . ;
- dispone di un sistema di tracciamento dello sguardo che gli consente di capire l'intento dell'utente e adattare gli ologrammi di conseguenza;
- fornisce la possibilità di interagire tramite comandi vocali;
- indipendente, senza fili o supporti esterni;

- possibilità di condividere il tempo reale ciò che si vede;

I test delle funzionalità sopra citate sono descritti nel paragrafo 4.6.

4.1.1 Un sistema indipendente

Per garantire un'esperienza d'uso ottimale, il visore HoloLens 2 è stato dotato di tutto il necessario per eseguire programmi e applicazioni in maniera indipendente, ovvero non necessita di supporti esterni per funzionare. Il processore integrato è un Qualcomm Snapdragon 850.

Negli ultimi anni questo processore, insieme ad altri con caratteristiche simili, viene impiegato anche nella costruzione di computer portatili, consentendo bassi consumi e quindi ridotti sistemi di raffreddamento. Tuttavia è necessario scendere a compromessi a livello di prestazioni, infatti nel caso del loro utilizzo nel mondo PC sono state introdotte delle ottimizzazioni software per permettere la fruizione di programmi di uso comune.

La stessa Qualcomm poi dichiara nelle specifiche la presenza di un supporto per l'intelligenza artificiale integrato nei suoi processori: Qualcomm Artificial Intelligence (AI) Engine, ciò lo rende particolarmente adatto ad essere impiegato in sistemi quali visori e smartphone [63].

Lo Snapdragon 850 è incentrato sull'archietttura ARM (Advanced RISC Machine), sigla che indica una famiglia di microprocessori RISC (Reduced Instruction Set Computing) a 32-bit e 64-bit utilizzata in maniera massiccia in una moltitudine di sistemi embedded grazie alle caratteristiche prima sottolineate. Tra i vantaggi dell'architettura RISC rientrano [64]:

- istruzioni poco numerose e semplici;
- le istruzioni non variano in base al tipo di dato trattato, ciò favorisce la lineare operatività del processore[6];
- elevata velocità di esecuzione delle istruzioni.

Per ulteriori specifiche si rimanda al sito del produttore [61].

[6]Non impiega più cicli macchina per processare la stessa istruzione ma con un tipo di dato diverso

4.1.2 Il sistema operativo

Il visore monta Windows Holographic OS che è una versione di Windows 10 riadattata per fornire migliori prestazioni riguardo ad esperienze di realtà mista [65]. Microsoft negli ultimi anni sta avviando un processo di "liberalizzazione" del proprio sistema operativo, rendendolo disponibile anche su realtà di terze parti che vogliano sviluppare software o hardware nell'ambito della realtà aumentata, mista e virtuale.

4.2 Il display

Una delle componenti principali del visore è il display, che sfrutta una particolare tecnologia al laser per funzionare e permettere di visualizzare ologrammi chiari e comprensibili anche in caso di testo da leggere. La tecnologia che si nasconde dietro la creazione di questo display si chiama MEMS, acronimo che sta per Micro Electro-Mechanical Systems e indica che la dimensione media dei componenti utilizzati nel processo si aggira attorno al micrometro. Per ogni occhio dunque è presente un trio di laser RGB (uno rosso, uno verde e uno blu) che, per generare un singolo fotone[7], viene acceso in modo da produrre lo spettro di colore desiderato. Il fascio di luce colpisce poi due specchi:

- Fast scan mirror: così chiamato perché viene colpito dal fascio luminoso ad una frequenza di 12 kHz. Il suo scopo è di ricevere il fotone ed espanderlo sulla parte orizzontale del display.
- Slow scan mirror: che lavora alla frequenza di 120 Hz e il suo compito è di espandere ciò che riceve dal fast scan mirror sulla parte verticale del display.

Tutto questo contribuirà a creare, sulle lenti del visore, un ologramma: quello che visualizzerà l'utente finale.

La Figura 4.2 rappresenta invece un ingrandimento al microscopio del Fast scan mirror. Il componente nero centrale oscilla attorno al proprio asse, identificato dalla presenza delle barrette che lo congiungono al resto del telaio, in modo da diffondere il fotone ricevuto dai laser.

[7] Il fotone è da intendersi come quanto di energia della radiazione elettromagnetica, storicamente chiamato anche quanto di luce

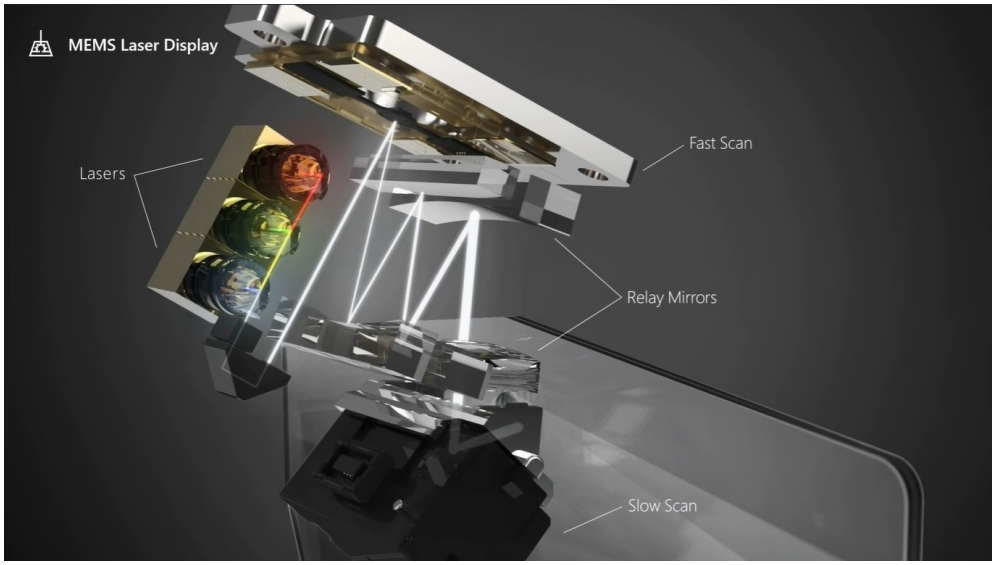

Figura 4.1: Struttura generale del sistema Display (Fonte: Microsoft)

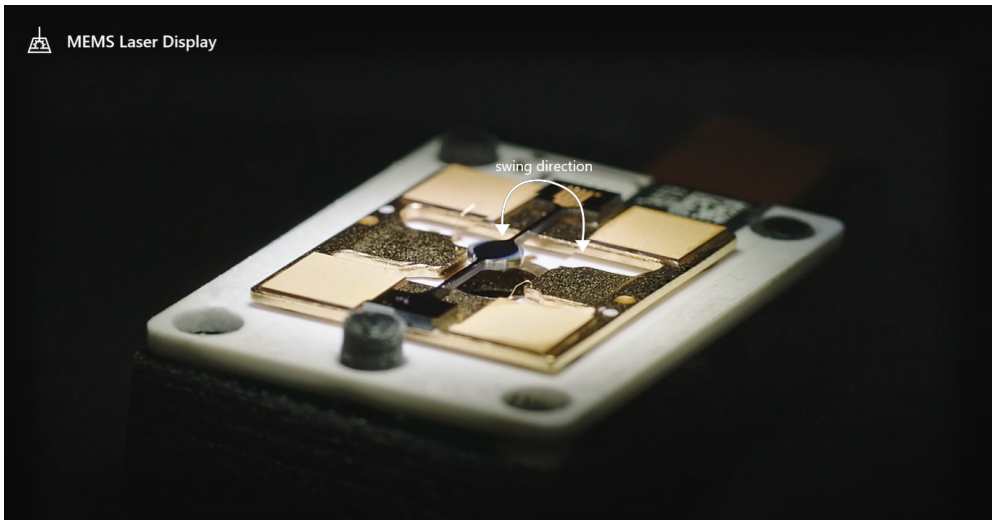

Figura 4.2: Struttura del fast scan mirror (Fonte: Microsoft)

Il movimento di oscillazione del fast scan (Figura 4.2) è creato dalla frequenza di alimentazione applicata ai capi del componente stesso. Il movimento così ottenuto è di estrema precisione e, seppur delicato, consente la sua potenziale applicazione all'interno di un prodotto commerciale sottoposto a forti stress.

4.2.1 Problemi nello sviluppo e soluzioni

La complessità di costruzione di questo metodo è giustificata da un limite dell'occhio umano che in optometria è definito come punto prossimo [66], ossia la minima distanza alla quale può trovarsi un oggetto per poterlo distinguere in

modo chiaro. Tale limite si attesta attorno ai 15/20 cm. È comprensibile che ottenere una distanza simile risulta impossibile nel caso in oggetto, poiché il visore è posto direttamente sulla testa dell'utente sotto forma di occhiali e deve essere poco voluminoso. È risultato però da studi ed esperimenti che, se i fotoni provenienti dai laser vengono proiettati nell'occhio allo stesso modo in cui esso percepisce il mondo reale, tale problema risulta risolto. Questo però richiede un sistema atto allo scopo [67].

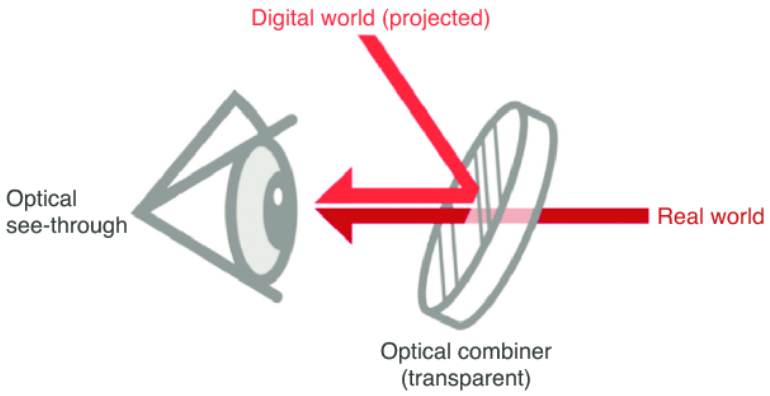

Figura 4.3: Direzione dei due fasci di luce entranti nell'occhio (Fonte: Yao Zhou, *Advances in the design of optical see-through displays* [11])

Nel visore HoloLens 2 il sistema adottato è la tecnologia ottica chiamata *waveguide*, ovvero un apparato fisico (meglio descritto nel paragrafo che segue) che guida le onde elettromagnetiche nello spettro ottico. Ciò permette di piegare e indirizzare la luce con precisione in modo da ottenere il risultato in Figura 4.3. In particolare è stata sfruttata una branca precisa di questa tecnologia, definita come SRG (Surface Relief Gratings).

4.2.2 Optical waveguide e SRG

Dalla Figura 4.4 è possibile individuare sulla lente l'area adibita alla guida delle onde elettromagnetiche, identificata all'interno dei perimetri rossi. Nel dettaglio, questa parte è realizzata tramite quello che viene chiamato diffraction grating o anche reticolo di diffrazione, ovvero una grata costituita da una serie di strutture lineari molto fini ripetute con periodicità dell'ordine delle lunghezze d'onda trattate.

Questo sistema agisce come un prisma con l'effetto collaterale di dividere la luce in diverse componenti in relazione alla loro lunghezza d'onda e di influenzare

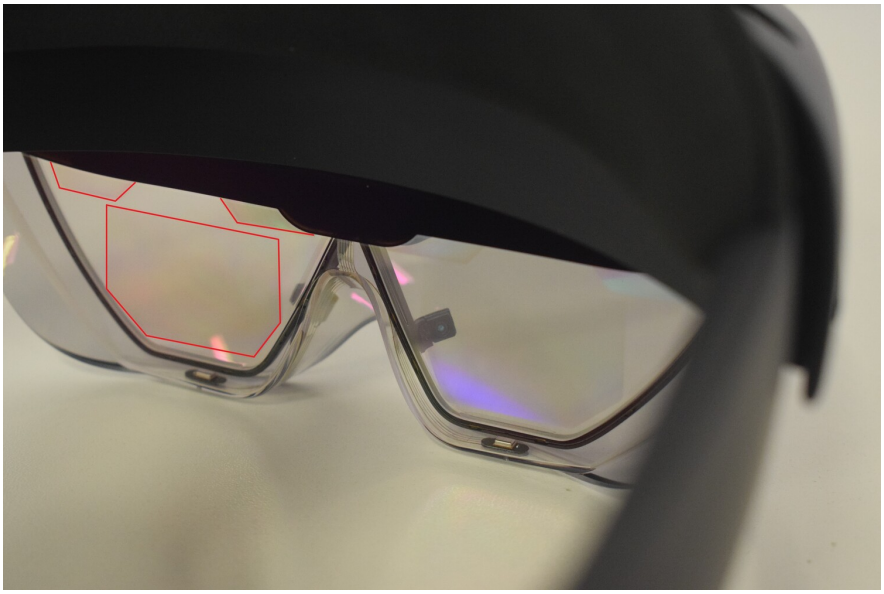

Figura 4.4: Struttura della lente

la polarizzazione[8] della stessa. Per evitare questo, il brevetto [68] indica che è stata modificata la struttura del reticolo, in modo da ottenere una convergenza dei diversi spettri di luce, come mostrato nelle Figure 4.5 e 4.6[9]. Per meglio visualizzare cosa comporta questo nella pratica, si riportano due foto rappresentative del reticolo in questione al microscopio (Figura 4.7) [67].

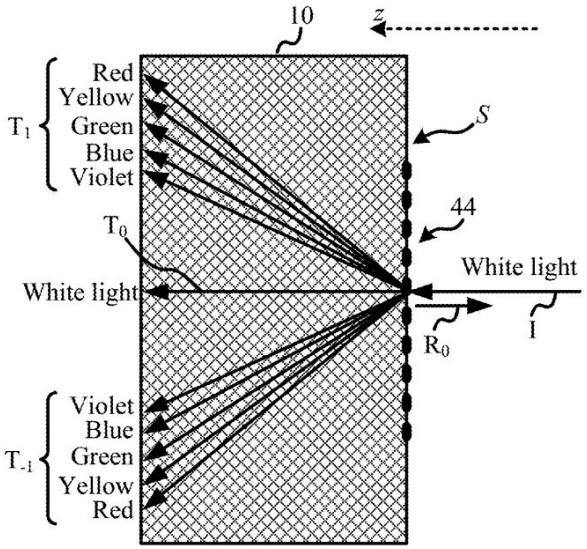

Figura 4.5: Incidenza della luce su una superficie piana (Fonte: USPTO [68])

[8]La polarizzazione è una proprietà che si applica alle onde trasversali che specifica l'orientamento geometrico delle oscillazioni.

[9]Le immagini rappresentano l'effetto di una superficie rispettivamente "binaria dritta", "binaria inclinata" e "triangolare a sbalzo" all'incidenza di un fascio di luce rappresentato dalla freccia superiore.

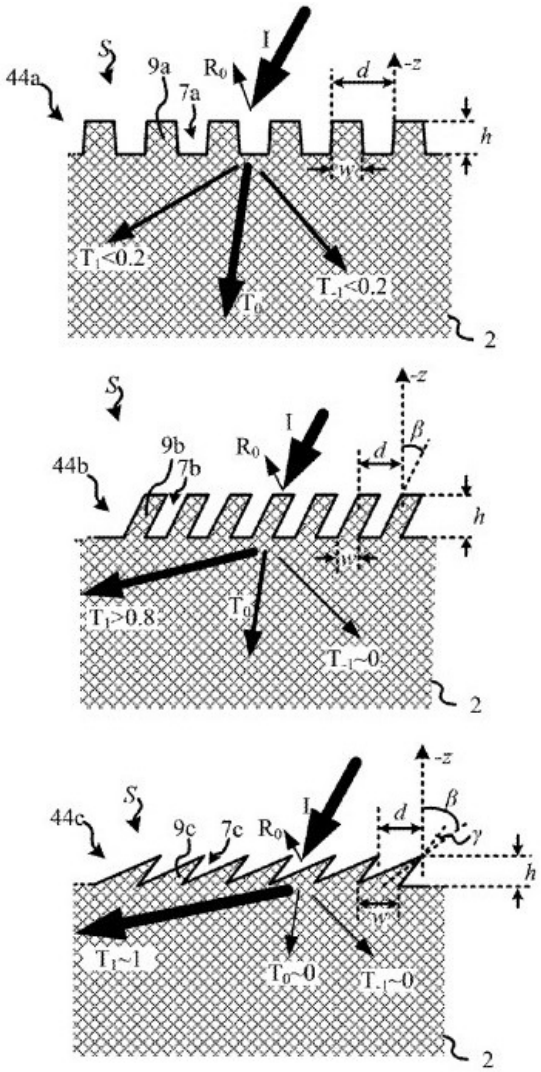

Figura 4.6: Incidenza della luce sulle diverse superfici (Fonte: USPTO [68])

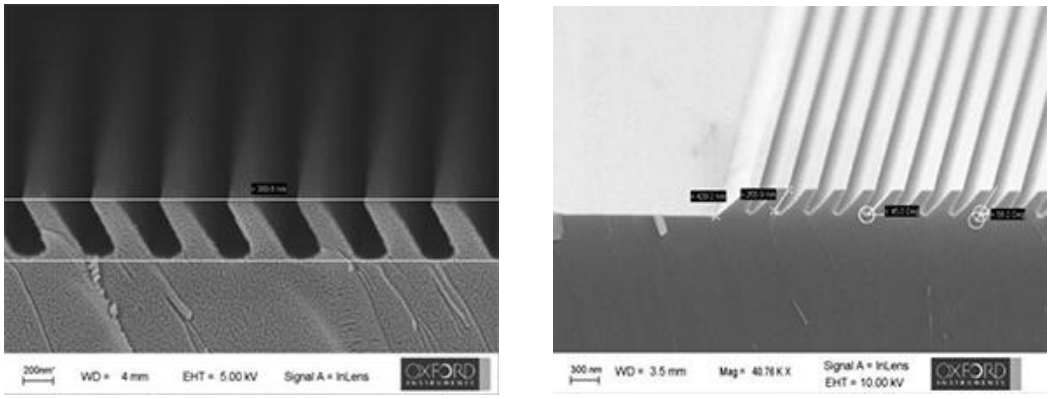

Figura 4.7: Foto al microscopio del reticolo di rifrazione (Fonte: Oxford Instruments)

Prima di proseguire è necessario definire anche il fenomeno del TIR: Total Internal Reflection, il principio secondo il quale una volta che la luce è entrata in una superficie di vetro o plastica trasparente con un certo angolo, essa verrà totalmente riflessa senza perdite.

Si può quindi ora definire tutto il processo:

- i fasci di luce/onde generate dai laser rimbalzano sui due specchi e arrivano al reticolo di diffrazione (paragrafo 4.2);
- il reticolo permette di curvare queste onde in modo da immetterle nella lente con un angolo che garantisce la TIR;
- il fascio di luce rimbalza fino ad arrivare all'occhio e qui viene estratto grazie a un secondo reticolo che nella Figura 4.8 è indicato con il numero "14/16", visualizzando l'immagine finale.

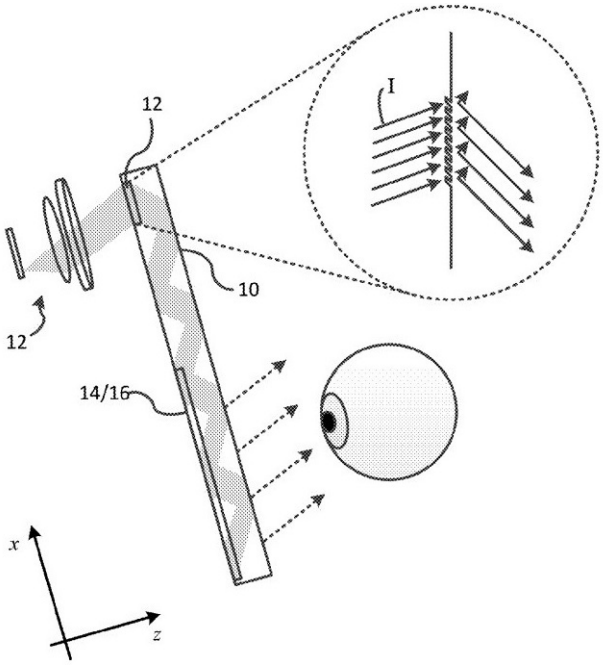

Figura 4.8: Funzionamento complessivo del display schematizzato (Fonte: USPTO [68])

4.3 Sensoristica di bordo

Un dispositivo come l'HoloLens 2 richiede un massiccio impiego di sensori per permettere tutte le funzionalità promesse. Nel dettaglio le specifiche dichiarate per i sensori sono descritte nella Tabella 4.1.

Funzione	Componente
Tracciamento della testa	4 videocamere a luce visibile
Tracciamento dello sguardo	2 videocamere a infrarossi
Misure di profondità	1-MP sensore Time-of-Flight
Unità di misura inerziale	Accelerometro, giroscopio, magnetometro
Videocamera	8-MP con risoluzione 1080p30

Tabella 4.1: Principali sensori montati a bordo del visore

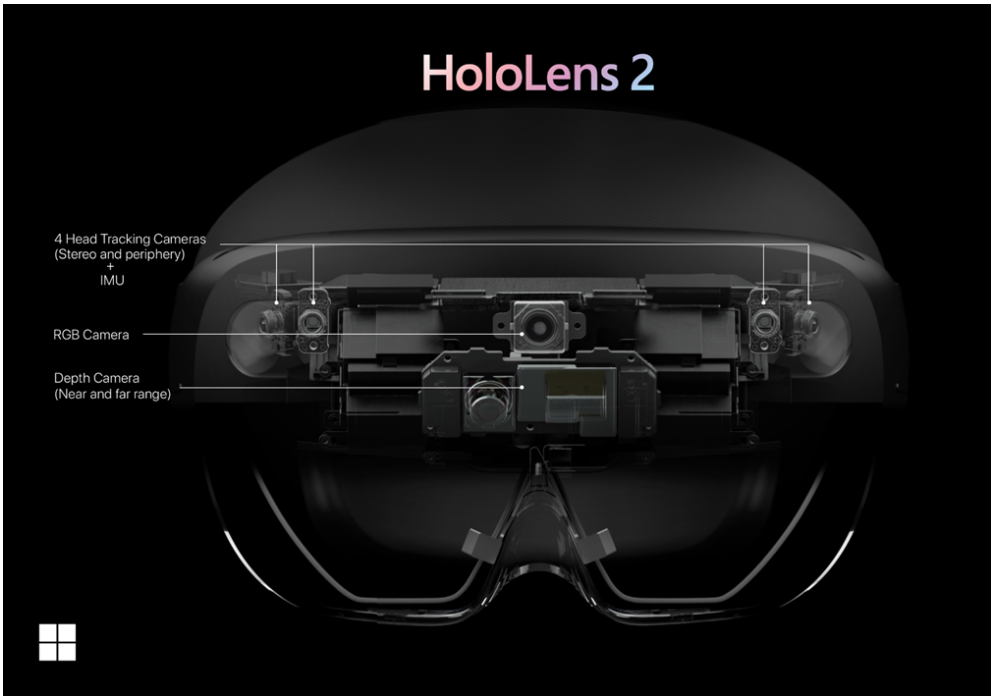

Figura 4.9: Disposizione dei sensori sul visore (Fonte: Microsoft)

Oltre alle 4 videocamere che permettono al dispositivo di percepire l'ambiente circostante, le tecnologie più interessanti sono quelle dell'eye tracking e della percezione della profondità, che verranno analizzate nei successivi paragrafi.

4.3.1 Eye tracking

Il processo di tracciamento dello sguardo o *eye tracking* permette al visore di riconoscere dove si sta focalizzando l'attenzione dell'utente e reagire di conseguenza. In termini generali si può dire che una luce infrarossa viene indirizzata verso il centro dell'occhio; essa sarà poi riflessa prima dalla pupilla e poi dalla cornea. Interpolando questi due punti tramite una videocamera a infrarossi è possibile

individuare un vettore che indica in modo preciso la direzione dello sguardo: in base a dove l'utente guarda, la distanza relativa tra i due punti varia. Questo metodo di tracciamento è anche chiamato *Pupil Center Corneal Reflection* (PCCR) [69, 12]

Figura 4.10: Schema di funzionamento PCCR (Fonte: Brain Support)

Il tutto è sostenuto da un iniziale calibrazione del sistema che permette di migliorarne le prestazioni.

I motivi per cui viene utilizzata una luce a infrarossi sono principalmente due:

1. la radiazione infrarossa ha una banda di frequenza dello spettro elettromagnetico inferiore a quella della luce visibile, quindi non arreca disturbo all'occhio umano;

2. non genera riflessi indesiderati che potrebbero inficiare la misurazione.

4.3.2 Misure di profondità

Le misure di profondità sono essenziali nell'ambito dell'orientamento spaziale. Nel caso in questione vengono effettuate tramite un cosiddetto "sensore a tempo di volo" o *Time-of-Flight depth sensor*. Anche questa tipologia di sensori sfrutta la luce a infrarossi, e il relativo riflesso, per determinare la distanza tra se stesso e l'ambiente circostante. La logica di funzionamento è illustrata in Figura 4.11.

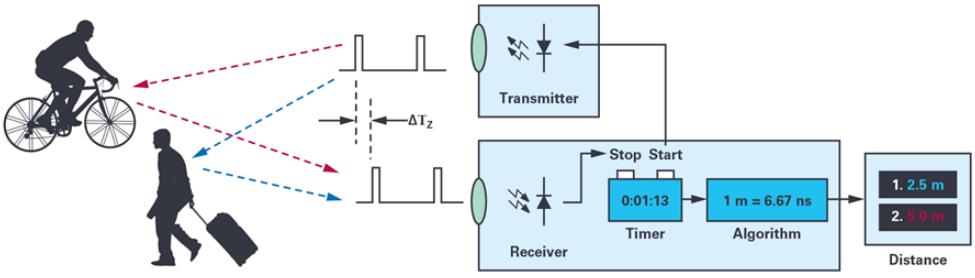

Figura 4.11: Schema di funzionamento di un sensore ToF (Fonte: Colm Slattery, *ADI ToF Depth Sensing Technology* [13])

A livello funzionale si possono distinguere due tecniche [13]:

- *Pulse method*: che è tra le due la più banale e consiste semplicemente nell'inviare un impulso di luce infrarossa e misurare il tempo che intercorre tra l'emissione della luce e la ricezione, da parte del sensore, del riflesso.
- *Continuous wave method*: nella quale viene emesso un segnale portante luminoso modulato in ampiezza, l'informazione sulla distanza viene estratta dal segnale ricevuto, ossia il riflesso, confrontando la sua fase con quella del segnale emesso.

Nella Figura 4.12 è possibile vedere l'effettivo posizionamento del sensore sul visore e nella Figura 4.13 la sua struttura esterna.

Figura 4.12: Posizionamento del sensore di profondità (Fonte: Microsoft)

Figura 4.13: Struttura esterna del sensore di profondità (Fonte: Microsoft)

Questo hardware è utilizzato sopratutto per le operazioni di *spatial mapping*, che "offre una rappresentazione dettagliata delle superfici reali nell'ambiente intorno al HoloLens, consentendo agli sviluppatori di creare un'esperienza di realtà mista convincente" [70].

Per avere un'idea generale, si analizzano ora le due già citate tecniche di funzionamento di questi sensori in ambito commerciale:

Pulsed method

Vengono emessi rapidi impulsi luminosi ad infrarossi che saranno riflessi dall'ambiente circostante e torneranno al sensore. Sensore che è equipaggiato con: interruttori elettronici che permettono di attivare l'acquisizione della luce solo per brevi finestre temporali, elementi fotosensibili per convertire la luce in corrente (tipicamente fotodiodi) ed elementi di memoria (tipicamente condensatori). Una volta ricevuto il segnale di ritorno è possibile ricavare la distanza sfruttando la Formula 4.1

$$d = \frac{1}{2} \cdot c \cdot \Delta t \left(\frac{Q_1}{Q_1 + Q_2} \right) \tag{4.1}$$

dove Q_1 e Q_2 rappresentano le quantità di cariche elettriche immagazzinate in due elementi di memoria e Δt è la durata dell'emissione dell'impulso luminoso. Tipicamente si approssima la velocità della luce a $c = 300.000.000\ m/s$.

In Figura 4.14 è poi rappresentato il diagramma temporale che descrive il funzionamento.

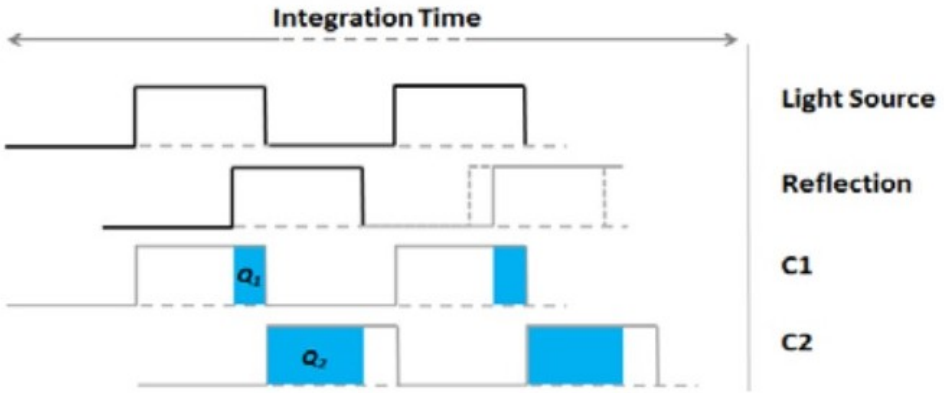

Figura 4.14: Principio di funzionamento del pulse method (Fonte: Larry Li, *Time-of-Flight camera - An introduction* [14])

Continuous wave method

Viene utilizzato lo stesso sensore, in questo caso però per calcolare la distanza viene sfruttato lo sfasamento come è rappresentato in Figura 4.15. Di conseguenza la formula diventa:

$$d = \frac{c}{2}\frac{\Delta\varphi}{2\pi f} \tag{4.2}$$

dove con f viene rappresentata la frequenza della modulante del segnale inviato e con φ lo sfasamento tra il segnale inviato e quello riflesso.

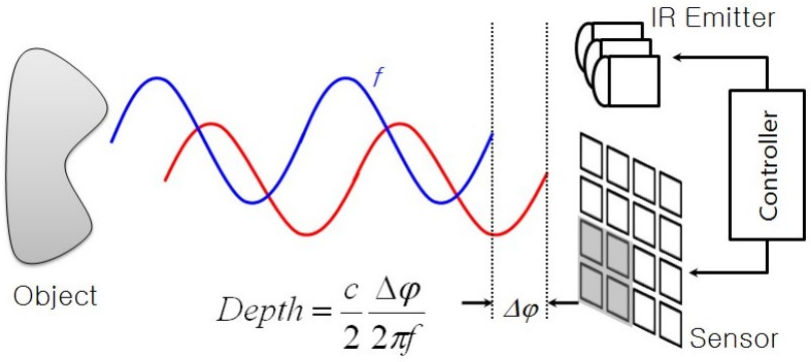

Figura 4.15: Principio di funzionamento del continuous wave method (Fonte: Miles Hansard, *Time-of-Flight Cameras: Principles, Methods, and Applications* [15])

4.4 Interazione con l'ambiente e spatial mapping

L'aspetto di interazione ambientale prevede una digitalizzazione dell'area circostante ed è indispensabile per poter garantire un'esperienza d'uso ottimale del visore. In questo modo sarà possibile proiettare ologrammi che verranno percepiti dall'utente alla stregua degli oggetti fisici. Tipicamente il processo sopracitato viene chiamato *mappatura spaziale* e si vuole trattare in questo paragrafo, in maniera generale, come viene fatto e gestito dalle varie applicazioni.

4.4.1 L'importanza della mappatura spaziale

La mappatura spaziale rende possibile la collocazione su superfici fisiche degli oggetti virtuali e la loro permanenza, cioè l'ancoraggio degli ologrammi nel mondo reale, che potranno essere dunque nascosti da oggetti frapposti tra loro e l'utente.

Lo scopo è sostanzialmente quello di non avere situazioni in cui gli ologrammi sono compenetrati con altre superfici.

4.4.2 Panoramica di funzionamento

Due sono gli oggetti o classi[10] principali a livello software sui quali si fonda la mappatura spaziale: lo *Spatial Surface Observer* e la *Spatial Surface*. L'applicazione in esecuzione su HoloLens 2 richiede allo Spatial Surface Observer di mappare un'area dello spazio, quest'ultimo provvede inviando all'applicazione un set di 'Spatial Surfaces' o superfici spaziali mappate. Ogni superficie fornita descrive una piccola porzione del mondo reale ed è rappresentata come una 'mesh'[11] o rete di triangoli ancorata allo spazio fisico, come si può vedere in Figura 4.16.

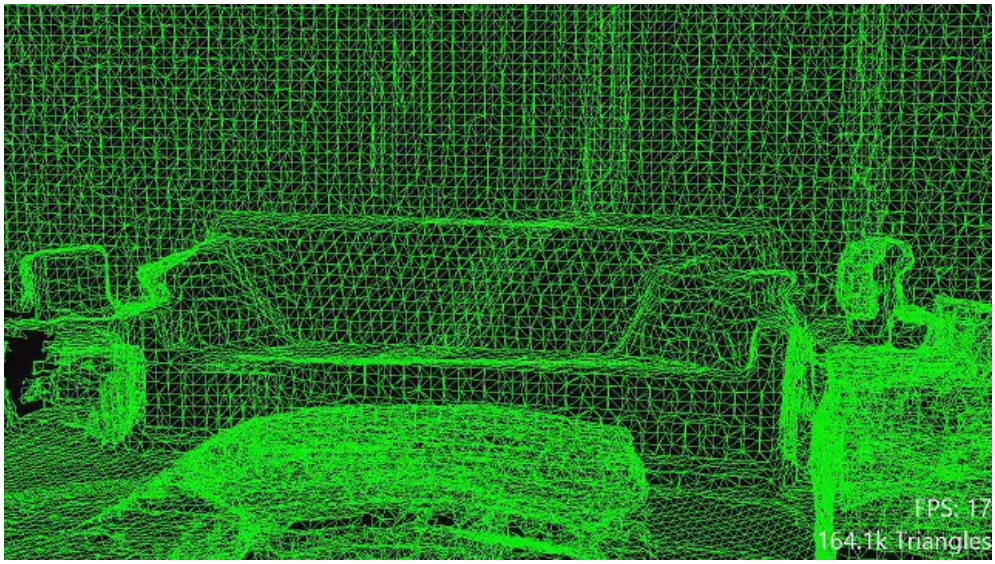

Figura 4.16: Mesh derivata dalla mappatura spaziale di una stanza (Fonte: Microsoft)

4.4.3 Lo Spatial Surface Observer

Il primo punto per una buona mappatura spaziale è l'utilizzo del Surface Observer, la logica di funzionamento di questo oggetto è così composta:

- Crea un *surface observer object*

[10] Questi due elementi sono delle classi definite in C++ o C# che contengono ed elaborano i dati sulla mappatura spaziale.

[11] Non fa riferimento all'applicativo Microsoft trattato nel paragrafo 3.1

  - È necessario specificare quali aree si vuole mappare, questo è possibile tramite quelli che vengono chiamati *volumi spaziali* ovvero delle forme che definiscono un'area nello spazio.
  - Tali volumi spaziali vengono ancorati al mondo fisico in modo da identificare un'area fissa.
  - Aggiornare l'area da mappare man mano che la visuale dell'utente si sposta.
  - I volumi spaziali possono essere modificati in qualsiasi momento, dipendentemente allo stato dell'applicazione o dell'utente.

- Utilizzare il metodo di polling o quello di notification per aggiornare le informazioni sulle superfici
  - Il polling opera un aggiornamento continuo delle superfici e viene impiegato nel caso in cui si abbiano ologrammi che seguono l'utente, il visore deve aggiornare continuamente la loro posizione in relazione allo spostamento.
  - Il notification invece è applicato nel caso di ologrammi statici, ovvero avviene una memorizzazione delle caratteristiche spaziali di dove l'oggetto è collocato e vengono aggiornate solamente se tale oggetto viene modificato.

- Gestione dei cambiamenti dei dati spaziali di diversi ambienti

Il procedimento sopra indicato ha come risultato finale la generazione della *mesh*.

4.4.4 Mesh caching

Il processo indicato nel paragrafo 4.4.3 è computazionalmente dispendioso per il processore, per questo si attua il caching delle informazioni, in modo da velocizzare il tutto.

È lasciata all'applicazione la gestione della tecnica di caching che deve ottimizzare l'utilizzo della memoria. Tra le scelte fondamentali da fare per uno sviluppatore spiccano:

- la gestione della generazione delle mesh delle nuove superfici;
- la modifica delle mesh quando lo spazio cambia;

- l'eliminazione delle mesh di superfici non più presenti.

L'ultimo punto è il più critico nel caso in cui il visore venga utilizzato in un ambiente che presenta più stanze, infatti il rischio di eliminare delle mesh che poi potrebbero nuovamente essere utili è alto.

4.4.5 Rendering spaziale

Affrontato l'argomento della creazione delle mesh, è ora fondamentale vederne un'applicazione pratica nel rendering degli ologrammi, ovvero la loro costruzione e gestione dello spazio. I principali utilizzi di una mesh sono dunque:

- visualizzazione delle superfici: ad esempio per visualizzare l'ombra di un ologramma su una superficie fisica;
- l'occlusione degli ologrammi dietro oggetti reali;
- modificare la forma e la posizione degli ologrammi in relazione allo spazio circostante.

4.5 Confronto con HoloLens 1

Si propone ora un confronto tra le specifiche delle due versioni del visore Microsoft HoloLens per discutere quali miglioramenti sono stati apportati.

HoloLens 1 monta una CPU Intel Atom con architettura a 32 bit, mentre HoloLens 2 ha una CPU Qualcomm Snapdragon a 64 bit con maggiore potenza di calcolo che permette quindi anche l'esecuzione di algoritmi di AI.

In HoloLens 2 si passa dalla classica memoria flash eMMC da 64GB di HoloLens 1 alla più veloce USF 2.1, sempre da 64GB. La RAM viene incrementata da 2GB a 4GB ed è stato inserito il nuovo standard di connettore USB. La connettività Bluetooth passa dalla versione 4.1 alla più moderna ed efficiente versione 5.0.

Entrambe le generazioni di HoloLens sono dotate del coprocessore HPU (Holographic Processing Unit) costruito appositamente da Microsoft. Il suo compito è quello di riconoscere mani, occhi, superfici e stanze. Permette inoltre il tracciamento 6DoF[12]. Questo coprocessore in HoloLens 1 aveva limitate prestazioni,

[12] "6 Degree of Freedom" permette di riconoscere i movimenti su 6 assi: 3 per la direzione e 3 per la rotazione

mentre la sua nuova versione inserita in HoloLens 2 migliora il campo di visione, i calcoli computazionali e la velocità della RAM.

La batteria più capiente garantisce un autonomia di 3 ore, rispetto alle 2,5 del modello precedente.

Il tracciamento dello sguardo è una peculiarità di HoloLens 2, assente in HoloLens 1. Questo aggiornamento permette, come già descritto nel paragrafo 4.3.1, di rendere l'interazione con il mondo digitale molto più fluida e naturale, anche grazie alla calibrazione IPD (InterPupillary Distance).

Il riconoscimento dell'iride, possibile grazie all'introduzione del tracciamento dello sguardo, permette a chi utilizza il visore un rapido login in alternativa all'inserimento di una classica password con molti caratteri.

La sensoristica di HoloLens 2 gli permette di riconoscere fino a 25 punti della mano, ciò non era possibile con la precedente versione di visore.

È stata migliorata la gesture di apertura del menu start di Windows che, in HoloLens 1, viene attivato tramite l'apertura della mano mentre in HoloLens 2 attraverso il tocco del polso.

Un'importante modifica è stata fatta al campo di visione (FoV) che passa da 34° ai 52° in HoloLens 2.

Microsoft ha incrementato la risoluzione del display ad un equivalente di 2K per occhio, rispetto ai 720p del modello precedente. Nonostante questo incremento di angolo di visione e di dettaglio, la densità di pixel rimane la stessa: 47 pixel per grado.

Un problema che rimane irrisolto nel display, rispetto alla prima versione di HoloLens, è la risoluzione verticale. Un ologramma, a meno che non sia posto esattamente nel campo visivo dell'utilizzatore, può risultare tagliato verticalmente quindi è necessario spostare la testa per visualizzarlo completamente.

La miglioria più visibile è stata fatta sull'ergonomia. La prima generazione ha tutto il peso concentrato nella parte anteriore che contiene l'unità logica, i sensori e la batteria. Ciò lo rende scomodo e doloroso da indossare per più di una dozzina minuti. Invece HoloLens 2 è molto più bilanciato, perché l'unità logica e la batteria sono state spostate sul retro. Anche se le due versioni hanno lo stesso ingombro e un peso simile, solo 13 grammi di differenza, la seconda generazione risulta molto più comoda.

Per far aderire in modo efficiente il visore alla testa dell'utente, è stata inserita posteriormente una ghiera che permette di stringere le fasce laterali. Sono state introdotte anche numerose imbottiture frontali e occipitali per consentire un comfort ottimale e duraturo.

Infine, per agevolare le azioni nel mondo reale senza l'ingombro delle lenti, queste ultime si possono sollevare evitando quindi di dover rimuovere HoloLens 2 dalla testa [71, 61].

La Tabella 4.2 riassume le specifiche tecniche delle due versioni di visore.

Specifiche	**HoloLens 1**	**HoloLens 2**
Processore	Intel Atom x5-Z8100P	Qualcomm Snapdragon 850
Architettura	32 bit	64 bit
Memoria	64 GB eMMC	64 GB USF 2.1
RAM	2 GB	4 GB
USB	micro USB 2.0	USB Type-C
Bluetooth	4.1	5.0
HPU	v1.0	v2.0
Durata batteria	2,5 ore	3 ore
Tracciamento mani	Una mano	Entrambe le mani
Gesture avanzate	No	Si
Sicurezza biometrica (scansione iride)	No	Si
Campo di visione (FOV)	34°	52°
Risoluzione display	1268x720 (per occhio)	1440x936 (per occhio)
Peso	579 grammi	566 grammi
Tracciamento occhi	No	Si
Sollevamento lenti	No	Si
Camera	2.4 MP, HD video	8MP, 1080p video
Audio	Altoparlanti, 3.5mm jack	Altoparlanti con spatial audio
Prezzo	$3000	$3500

Tabella 4.2: Comparazione specifiche delle due versioni di HoloLens

4.6 Prova pratica in laboratorio

La prova pratica effettuata in laboratorio è stata molto utile per delineare i limiti e le capacità del visore. Appena indossato e regolato adeguatamente in base alla forma della propria testa, si avvia la configurazione come un qualsiasi computer con Windows. Una volta terminata la configurazione, il visore richiede una calibrazione dovuta alla soggettività della conformazione del viso di ciascun individuo.

4.6.1 Calibrazione

Viene offerta la calibrazione per due parametri principali:

- calibrazione della percezione: per permettere all'utente di percepire gli ologrammi in maniera corretta;
- calibrazione del colore: può essere necessaria anche in base al tipo di luce ambientale.

Dopo vari test, dove si è cambiato l'utilizzatore, si è notato che senza il procedimento di calibrazione della percezione (si può eventualmente ignorare la calibrazione del colore) l'esperienza non è ottimale e la percezione degli ologrammi risulta distorta. Questo si riflette in una difficoltà ad interagire con gli elementi digitali (selezionarli, spostarli ecc...) e una sfocatura degli stessi, rendendone tedioso l'utilizzo.

In Figura 4.17 e 4.18 è implementata la regolazione. Dopo averla avviata il visore chiederà all'utente di mantenere la testa ferma e seguire con lo sguardo delle figure geometriche che verranno visualizzate sul display.

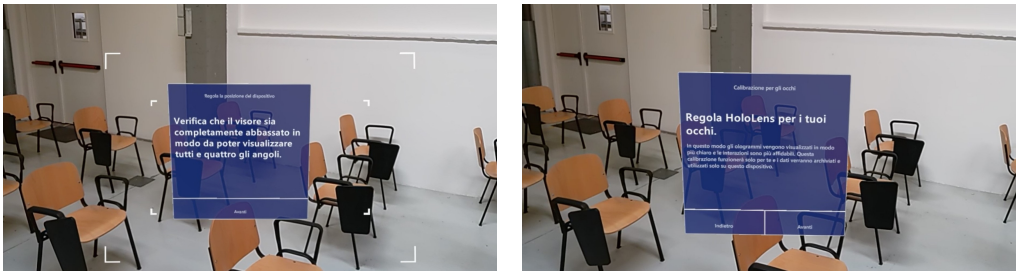

Figura 4.17: Schermate di avvio della calibrazione

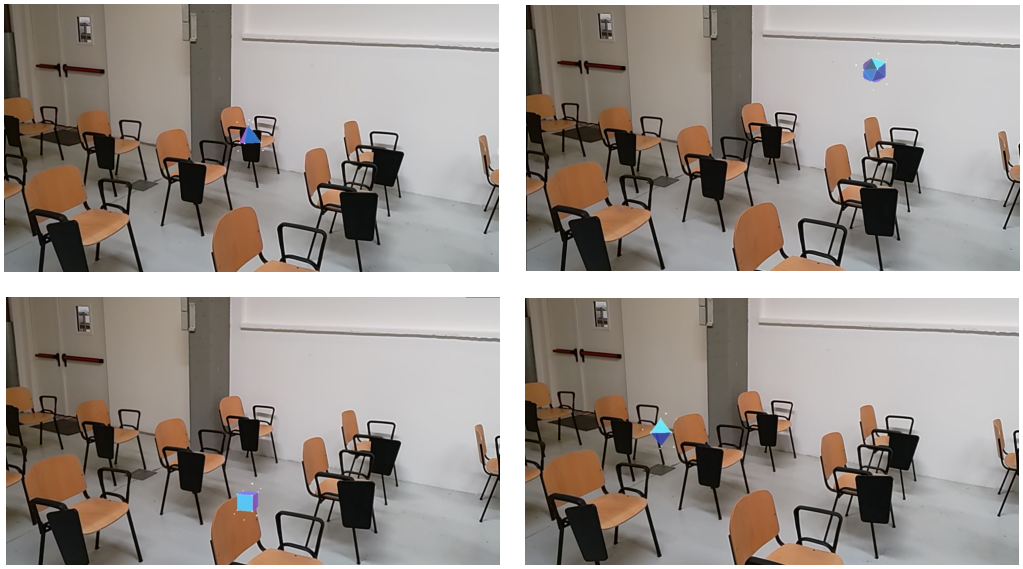

Figura 4.18: Procedura di calibrazione

4.6.2 Il campo visivo

Il campo di visione dell'HoloLens 2, nonostante notevolmente migliorato rispetto a quello del primo modello, risulta comunque ristretto rispetto alle aspettative, in quanto il cervello umano è abituato ad un angolo di visione di 200° [72]. Gli ologrammi sono apprezzabili solamente se inquadrati nel pieno campo visivo dell'utente, la visione periferica degli stessi risulta limitata se non nulla, come rappresentato nella Figura 4.19. Si è notato inoltre che muovendo la testa, ciò che si vede viene riadattato per renderlo percepibile in modo naturale nello spazio, dando una prospettiva diversa a seconda di dove punta lo sguardo.

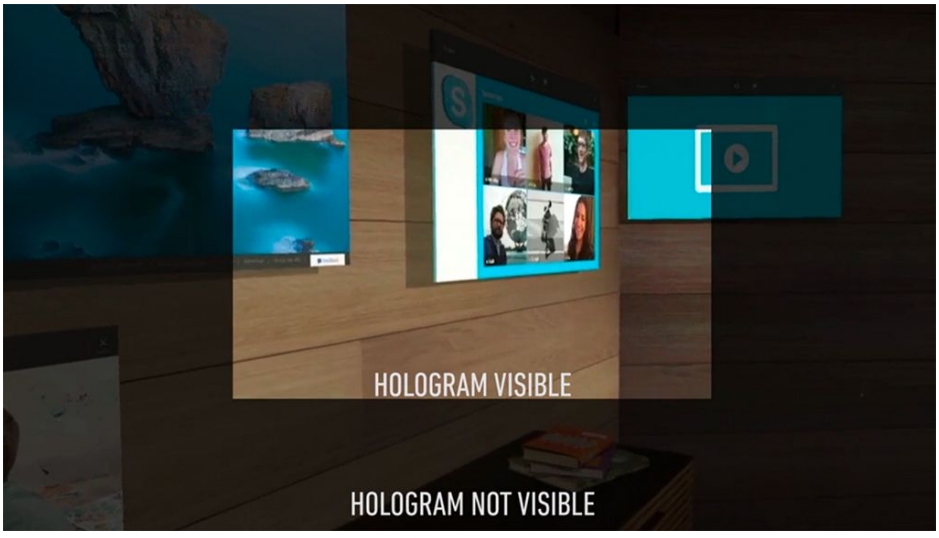

Figura 4.19: Campo di visione dell'HoloLens 2 (Fonte: The Verge)

4.6.3 Visibilità e persistenza degli ologrammi

Se ben centrati nel campo visivo dell'utente, gli ologrammi risultano di ottima qualità e perfettamente nitidi. La luce ambientale, soprattutto se frontale, ne inficia la nitidezza rendendoli sbiaditi e accorciando la distanza da cui possono essere visti, arrivando a valori pratici di 4-5 metri in un ambiente chiuso, mentre dimezza all'esterno. Come limite estremo, se si ingrandiscono molto gli ologrammi, la distanza di visione nitida aumenta fino a 7-8 metri.

Per quanto riguarda la persistenza degli ologrammi, le prestazioni sono veramente eccellenti. Il visore è in grado di mappare l'area circostante e mantenere gli oggetti fissi nella loro posizione. Anche cambiando ambiente, che sia una stanza chiusa oppure uno spazio esterno, mantiene le sue caratteristiche ed è possibile interagire con ciò che è stato precedentemente posizionato.

Nel caso in cui delle componenti strutturali dell'ambiente fisico si sovrappongano a degli elementi digitali ancorati nello spazio, fa sì che quest'ultimi vengano nascosti dai primi, ma aggirando l'ostacolo fisico l'ologramma risulta nella posizione corretta, come in Figura 4.20. Tutte le considerazioni precedenti valgono anche in assenza di connessione di rete internet e con scarsa luminosità.

È da segnalare il fatto che gli oggetti digitali non rispettano i piani inclinati, ovvero si appoggiano sul piano (anche se inclinato) ma il loro asse verticale resta ortogonale al terreno.

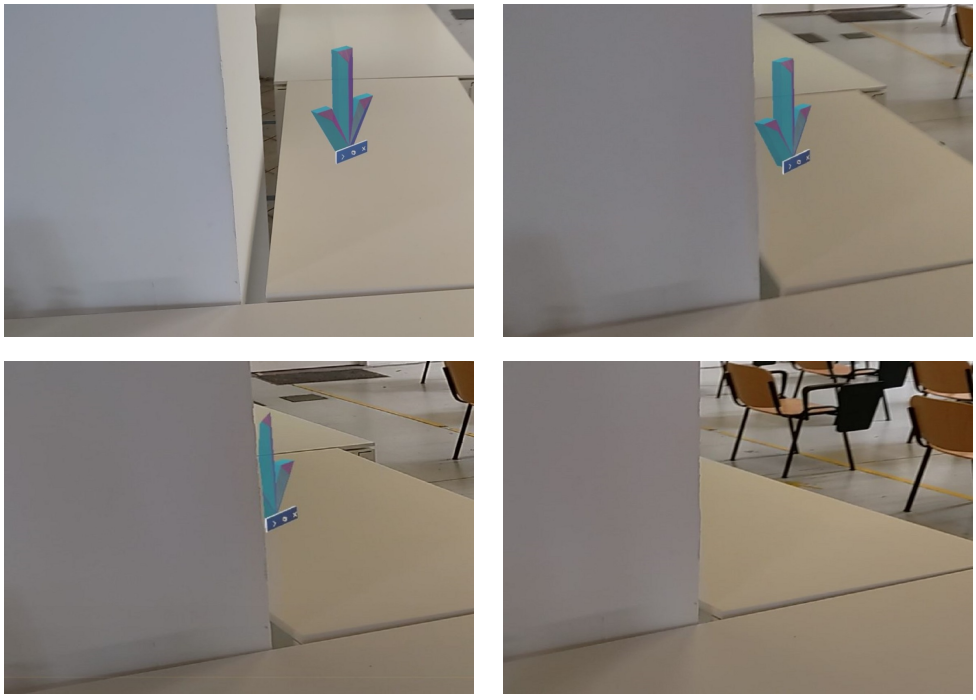

Figura 4.20: L'ologramma sparisce dietro l'angolo del muro

4.6.4 Interazione con gli ologrammi

L'interazione con gli oggetti digitali posizionati nel mondo reale avviene tramite l'uso delle mani e alcune gesture, le principali sono:

- tocco con l'indice, per selezionare;
- pinch, ovvero unire il pollice e l'indice per poter afferrare, allargare, spostare.

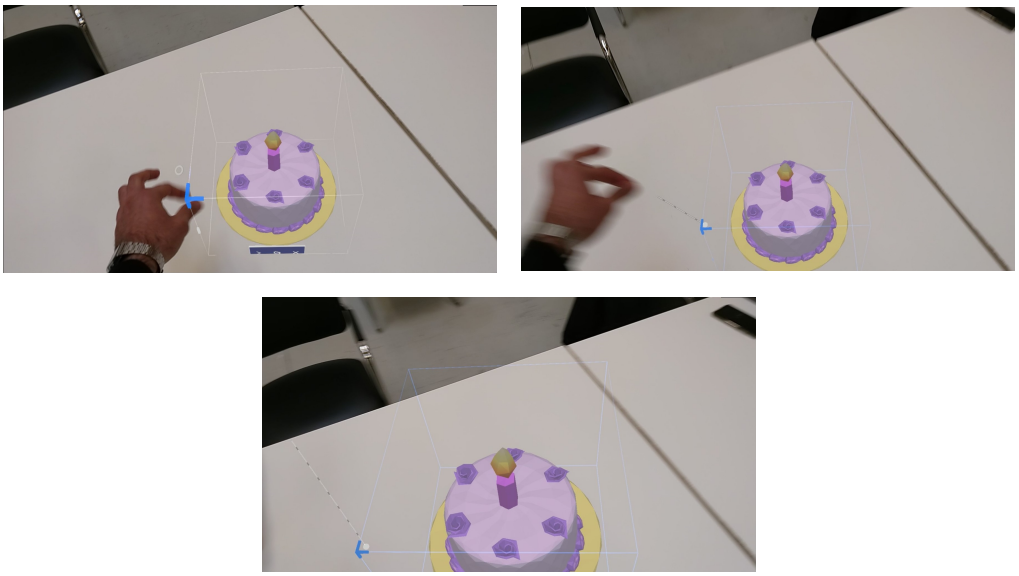

Figura 4.21: Il pinch serve per aumentare la dimensione degli ologrammi

Se adeguatamente calibrato, il tocco risulta preciso e permette di interagire in modo agevole con il mondo digitale. È possibile modificare gli ologrammi anche a distanza, data la presenza di due rette che identificano la posizione delle mani nello spazio, come si può vedere in Figura 4.22.

Un'altra possibilità di interazione tra utente ed oggetti è quella dei comandi vocali. Avendo i microfoni posizionati sotto le lenti, come si può notare il Figura 4.23, HoloLens 2 riesce a captare distintamente la voce di chi lo indossa, a discapito di rumori esterni anche piuttosto consistenti.

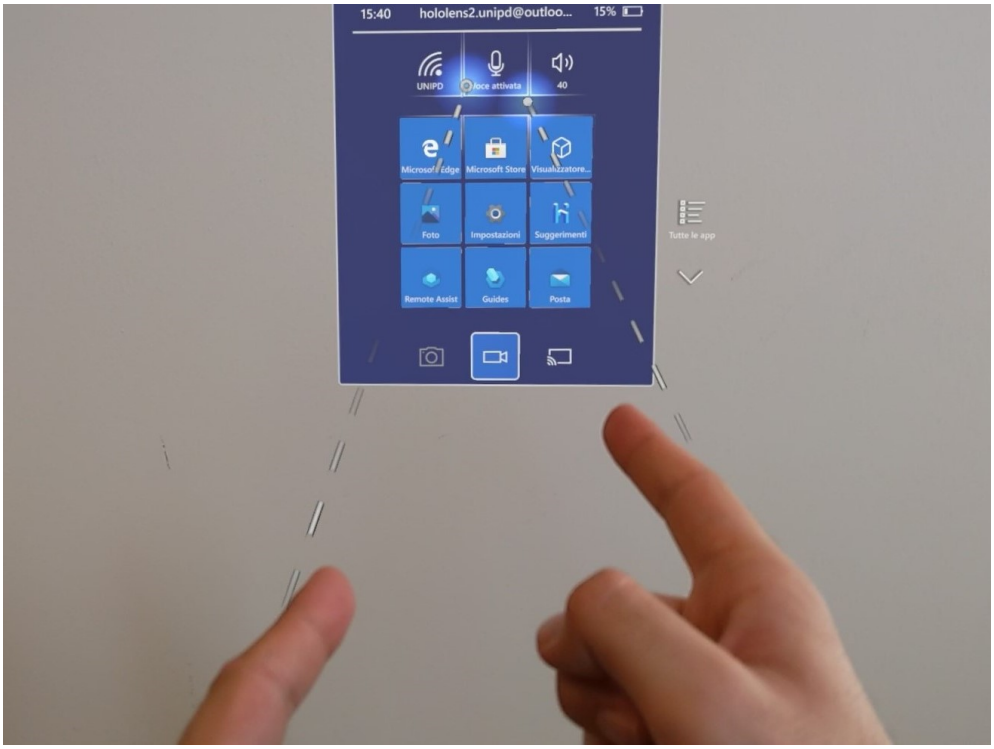

Figura 4.22: Nella selezione a distanza, due fasci luminosi rappresentano la posizione delle dita nello spazio

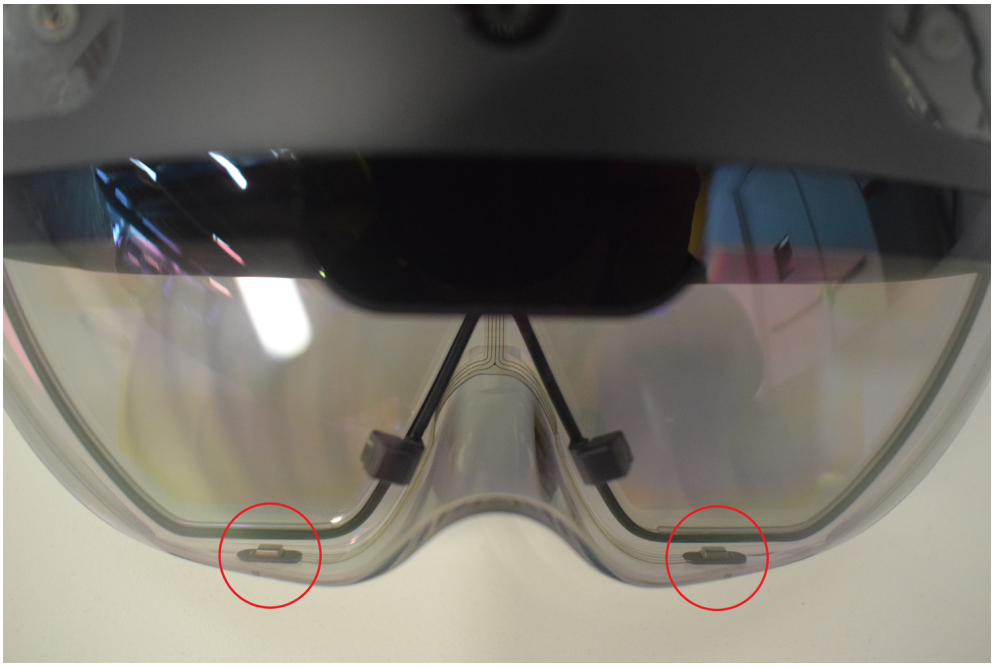

Figura 4.23: Microfoni posizionati sotto le lenti

Capitolo 5

Realizzazione di un'applicazione per HoloLens 2

Saper sviluppare un'applicazione per realtà aumentata permette di sfruttare appieno le potenzialità di HoloLens 2.

In questo capitolo si vuole offrire un esempio di sviluppo di un applicativo, partendo dall'analisi del contesto in cui verrà sfruttato, per poi analizzare gli strumenti e la procedura di creazione. È però solo una realizzazione preliminare che potrà essere sviluppata e integrata con altri strumenti.

5.1 Contesto di utilizzo

Ci si vuole focalizzare su un contesto lavorativo in ambito industriale. Qui tutti gli aspetti del lavoro devono essere studiati in maniera scientifica, per stabilire leggi, regole e formule che permettano di ottimizzare la produzione. Vale a dire che ogni parte del processo produttivo, che sia una linea di fabbricazione o di assemblaggio, deve essere analizzata per poter definire il modo più veloce ed efficiente per compiere ogni operazione necessaria ad ottenere il prodotto finito. In ultimo, seguendo le regole appena tracciate, viene eseguita una standardizzazione della produzione.

Le operazioni fondamentali per la stesura di politiche di produzione funzionali sono:

- lo studio dei tempi, che sfrutta diverse metodologie tra cui: cronometraggio dei tempi, *sistemi a tempi predeterminati* [73];

- lo studio dell'ergonomia, per permettere all'operatore di lavorare in modo agevole anche per diverse ore.

Per questo e altri tipi di analisi a livello logistico, come il *Parts Feeding*[13], è necessario poter simulare una postazione di lavoro, con la flessibilità di spostamento dei vari materiali in modo da trovare la configurazione adatta.

Da qui si vuole partire per lo sviluppo dell'applicazione in oggetto. Ovvero dare la possibilità di ricreare un banco da lavoro con diverse UDC (Unità Di Carico[14]) sotto forma di ologrammi, e acquisire la posizione delle stesse in modo da poter replicare il tutto con le unità fisiche.

Pensando a un contesto più ampio, si potrebbe integrare tutto questo con un metodo di analisi dei movimenti dell'operatore. Una *tuta inerziale*, per esempio, permetterebbe di approfondire notevolmente tutte le analisi ergonomiche e i cronometraggi dei tempi.

5.2 Programmi utilizzati

I software utilizzati per lo sviluppo dell'applicativo sono:

- Unity per la programmazione grafica e il relativo script di acquisizione della posizione;
- Visual Studio per ottenere l'applicazione sotto forma di pacchetto installabile sul sistema operativo Windows Holographic.

Lo script è stato scritto con linguaggio C# e permette appunto di salvare le coordinate degli ologrammi in un file di testo, posizionato in una determinata directory all'interno di HoloLens. L'origine del sistema di riferimento è un cubetto rosso che appare al centro della stanza (Figura 5.1).

5.3 Sviluppo con software Unity

L'ambiente di sviluppo si presenta con la schermata in Figura 5.1.

[13]Metodo con cui i componenti vengono portati e posizionati nella postazione di lavoro
[14]Contenitori dentro i quali vengono posizionati i diversi componenti

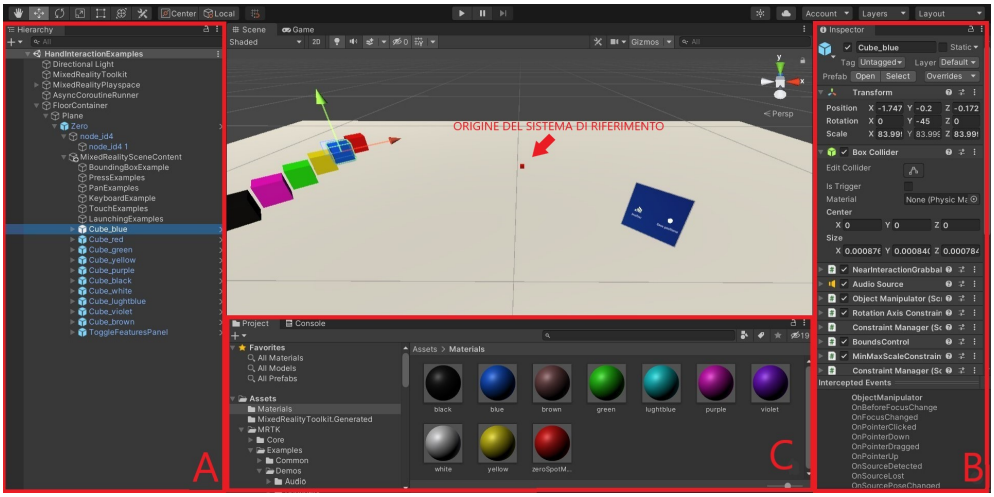

Figura 5.1: IDE di Unity

Le diverse aree evidenziate con le lettere A, B e C sono le più importanti nella creazione di un'applicazione.

Hanno i seguenti scopi:

A. **Hierarchy**: offre una panoramica degli oggetti presenti nel progetto e permette di accedere ai relativi attributi.

B. **Inspector**: mostra tutte le *componenti* associate ad un determinato oggetto, ogni componente ha un ben preciso effetto sul comportamento dell'oggetto.

C. **Project**: elenca i file presenti nella cartella di progetto.

Al centro è visualizzata invece una simulazione dell'ambiente.

Per creare un'applicazione con le caratteristiche elencate finora, è necessario eseguire due passaggi fondamentali: creare gli oggetti e scrivere lo script che ne acquisisce la posizione.

5.3.1 Creazione degli oggetti

Per simulare le UDC, sono stati creati dieci cubi di diversi colori (Figura 5.2). Questi sono manipolabili e possono essere traslati nello spazio, ruotati e ingranditi a piacimento, in modo da essere più flessibili possibile.

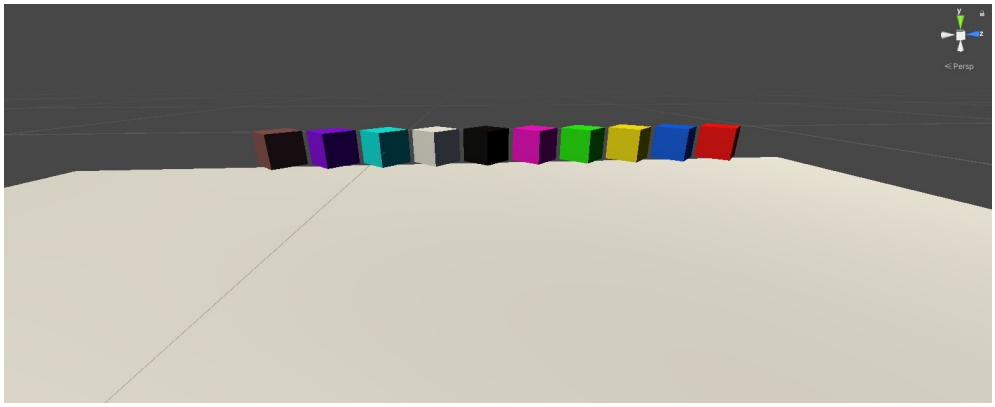

Figura 5.2: Cubi per simulare UDC

Ad ogni cubo è stato assegnato un *materiale*, che ne determina il colore, e diverse componenti che si possono dividere in quattro categorie: script, audio source, box collider, transform.

Tutte queste componenti sono visibili in Figura 5.1 nel riquadro a destra, contrassegnato con la lettera B.

Script

Come si può notare dalla Figura 5.3, sono presenti vari script che permettono all'oggetto di essere manipolato, facendo si che ad ogni tocco della mano dell'utente corrisponda una reazione. Ad esempio l'*Object Manipulator* permette la traslazione e la rotazione del cubo.

Si segnala che tutti gli script relativi al movimento dell'oggetto sono stati importati direttamente dalla libreria MRTK (Mixed Reality ToolKit) fornita da Microsoft per lo sviluppo di software per realtà mista.

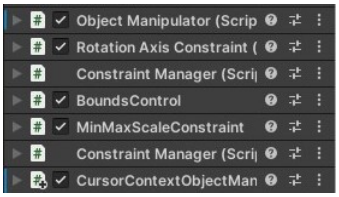

Figura 5.3: Script relativi al movimento degli oggetti

Audio source, box collider e transform

Per quanto riguarda *audio source*, è semplicemente il suono che viene riprodotto quando viene effettuata una manipolazione del cubo, come la traslazione o rotazione.

Il *box collider* permette agli oggetti di collidere tra di loro, in modo da poter definire una risposta alla collisione tramite uno script.

Transform è un attributo di default legato agli oggetti 3D che riporta la loro posizione nello spazio.

5.3.2 Realizzazione dello script

Per tutto questo paragrafo si farà riferimento al Codice 5.1.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using System.IO;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System;
9
10 public class Position : MonoBehaviour
11 {
12    //Definizione degli oggetti
13    public GameObject Cube_blue;
14    public GameObject Cube_red;
15    public GameObject Cube_green;
16    public GameObject Cube_yellow;
17    public GameObject Cube_purple;
18    public GameObject Cube_black;
19    public GameObject Cube_white;
20    public GameObject Cube_lightblue;
21    public GameObject Cube_violet;
22    public GameObject Cube_brown;
23
24    //Funzione di salvataggio della posizione
```

```
25      public void SavePosition()
26      {
27          //Acquisizione della posizione di ogni cubo
28          Vector3 posCube_blue = Cube_blue.transform.localPosition;
29          Debug.Log(posCube_blue);
30
31          Vector3 posCube_red = Cube_red.transform.localPosition;
32          Debug.Log(posCube_red);
33
34          Vector3 posCube_green = Cube_green.transform.localPosition;
35          Debug.Log(posCube_green);
36
37          Vector3 posCube_yellow = Cube_yellow.transform.localPosition;
38          Debug.Log(posCube_yellow);
39
40          Vector3 posCube_purple = Cube_purple.transform.localPosition;
41          Debug.Log(posCube_purple);
42
43          Vector3 posCube_black = Cube_black.transform.localPosition;
44          Debug.Log(posCube_black);
45
46          Vector3 posCube_white = Cube_white.transform.localPosition;
47          Debug.Log(posCube_white);
48
49          Vector3 posCube_lightblue = Cube_lightblue.transform.
                localPosition;
50          Debug.Log(posCube_lightblue);
51
52          Vector3 posCube_violet = Cube_violet.transform.localPosition;
53          Debug.Log(posCube_violet);
54
55          Vector3 posCube_brown = Cube_brown.transform.localPosition;
56          Debug.Log(posCube_brown);
57
58          //Conversione in un'unica stringa delle coordinate
59          List<string> positions = new List<string>
60          {
61              "Blue cube coordinates:", posCube_blue.ToString(),
62              "Brown cube coordinates:", posCube_brown.ToString(),
63              "Red cube coordinates:", posCube_red.ToString(),
64              "Green cube coordinates:", posCube_green.ToString(),
65              "Yellow cube coordinates:", posCube_yellow.ToString(),
66              "Purple cube coordinates:", posCube_purple.ToString(),
67              "Black cube coordinates:", posCube_black.ToString(),
```

```
68            "White cube coordinates:", posCube_white.ToString(),
69            "Lightblue cube coordinates:", posCube_lightblue.ToString()
                  ,
70            "Violet cube coordinates:", posCube_violet.ToString(),
71        };
72
73        //Definizione del percorso del file di destinazione
74        string path = Path.Combine(Application.persistentDataPath, "
              MyFile.txt");
75
76        //Scrittura della stringa precedentemente definita su file
77        File.WriteAllLines(path, positions);
78    }
79 }
```

Codice 5.1: Position.cs

Come prima cosa vengono dichiarati gli oggetti. Il tipo di variabile è *GameObject* e ha attributo *public*. Quest'ultimo definisce se ciò che segue può essere visto da agenti esterni allo script stesso, come per esempio l'IDE Unity.

Successivamente si definisce la funzione *SavePosition*, anche questa con attributo public e di tipo *void* dato che non dovrà restituire nessun valore. Un GameObject è assimilabile ad una *class* e ha degli attributi ai quali è possibile accedere scrivendo "nome_oggetto.attributo". Così viene fatto in questo caso, in più viene aggiunta l'istruzione *transform* che serve appositamente per acquisire la posizione.

LocalPosition è l'attributo che contiene le coordinate x, y e z dei diversi cubi. Quest'ultime possono variare a seconda del sistema di riferimento che si sceglie. Nel caso in oggetto i cubi sono stati creati come *figli* del piano chiamato *Plane*, ciò implica che le loro coordinate sono relative al centro del piano identificato da un cubetto rosso, chiamato *Zero* (Figura 5.4), che essendo mobile può essere posizionato dove si ha necessità. La scelta di utilizzare un segnale per identificare il punto di riferimento è stata fatta per permettere una più semplice comprensione delle posizioni una volta che si andrà ad utilizzare l'applicazione sul visore.

Infine per stampare i dati acquisiti sul file, si riportano le posizioni di tutti i cubi salvate in dei vettori (Vector3) in un'unica stringa, tramite un cast con funzione *ToString()*; poi si definisce il *path*, ovvero il percorso del file di destinazione.

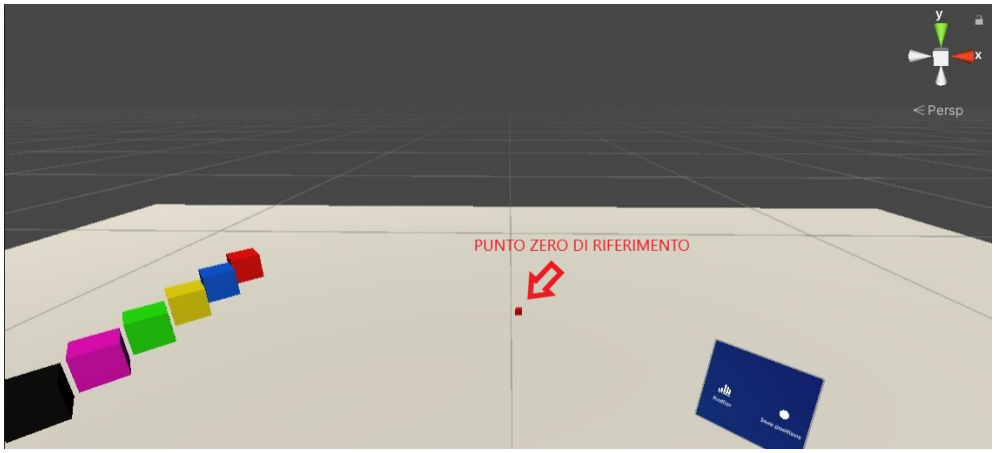

Figura 5.4: Punto Zero a cui fanno riferimento le coordinate degli oggetti

Successivamente si scrivono le coordinate tramite l'istruzione *WriteAllLines*.

È possibile verificare il funzionamento dell'applicazione direttamente sull'IDE Unity. Una volta avviata la simulazione e spostati i cubi, l'utente potrà avviare l'acquisizione della posizione premendo l'apposito tasto con dicitura *Save Positions*, che si presenta sotto forma di ologramma.

Il risultato di scrittura sul file è visibile nel Codice 5.2.

5.4 Prova dell'applicazione in laboratorio

Una volta compilata e installata su HoloLens 2, l'applicazione si avvia correttamente e vengono visualizzati tutti gli elementi.

Inizialmente si posiziona il cubetto che identifica il riferimento, come in Figura 5.5. In questo caso la scelta è ricaduta sul bordo del banco da lavoro, ma può variare a seconda delle esigenze.

È possibile posizionare gli ologrammi a 360° attorno all'operatore e questi manterranno la loro collocazione. Le foto in Figura 5.6 rappresentano quanto descritto, gli scaffali sono posti alle spalle dell'utente, mentre il banco da lavoro di fronte. Girandosi quindi è possibile vedere tutti gli oggetti, anche se essi spariscono momentaneamente quando sono all'esterno del campo di visione.

Figura 5.5: Posizionamento del riferimento

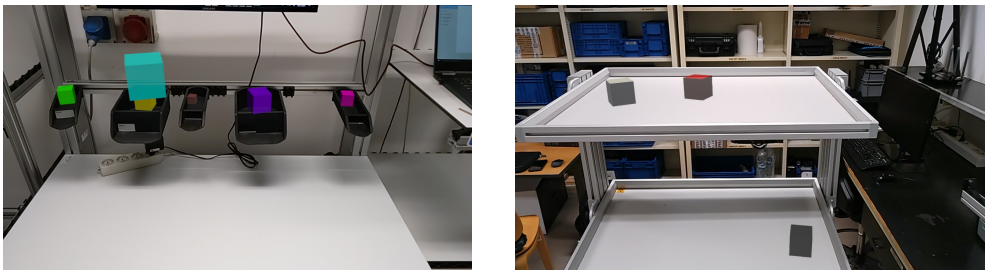

Figura 5.6: Visione dell'operatore

Anche il salvataggio della posizione funziona correttamente; premendo il pulsante come rappresentato in Figura 5.7, il risultato di scrittura su file è analogo a quanto ottenuto in simulazione (Codice 5.2).

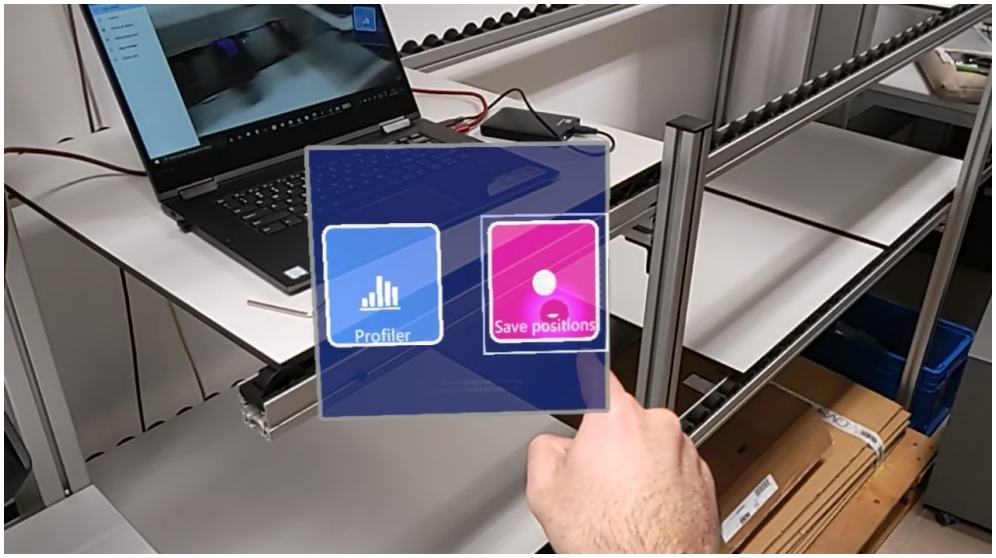

Figura 5.7: Pulsante di salvataggio della posizione

```
Blue cube coordinates:
(-1.7, -0.2, -0.2)
Brown cube coordinates:
(-1.2, -1.0, -0.7)
Red cube coordinates:
(-1.7, -0.2, -0.1)
Green cube coordinates:
(-0.6, -1.2, 0.2)
Yellow cube coordinates:
(-1.7, -0.2, -0.3)
Purple cube coordinates:
(-1.7, -0.2, -0.5)
Black cube coordinates:
(-1.7, -0.2, -0.6)
White cube coordinates:
(-1.7, -0.2, -0.7)
Lightblue cube coordinates:
(-2.5, -1.0, -1.6)
Violet cube coordinates:
(0.2, -0.9, -2.3)
```

Codice 5.2: MyFile.txt

Capitolo 6

Estensione dell'applicazione

L'obiettivo dell'applicazione è fornire un servizio di posizionamento e salvataggio delle posizioni di oggetti virtuali in un ambiente di lavoro reale. L'utente deve poter scegliere con precisione la collocazione di questi ologrammi per riprodurre lo spazio di lavoro che ritiene più corretto. Il progetto, infatti, è pensato per lavorare insieme ad una tuta a sensori inerziali, che monitora gli indici ergonomici dell'utilizzatore in tempo reale. Un'analisi di questi dati permetterebbe di rilevare eventuali movimenti dannosi a lungo termine per l'utente e di rivalutare il posizionamento degli oggetti.

In questo lavoro di tesi si è sviluppata l'applicazione per il visore HoloLens 2 che consente di spostare e salvare la posizione degli oggetti. I modelli scelti sono da considerarsi puramente indicativi, e si rimanda all'appendice A per istruzioni su come importare i propri oggetti.

6.1 Funzionamento dell'applicazione

L'applicazione prevede due set di oggetti diversi, che possono essere selezionati attraverso un menu principale. Quando viene selezionato un set, si possono far apparire, ruotare e cambiare le dimensioni degli oggetti da posizionare. I modelli sono quindi inizialmente disattivati (e invisibili) ed è l'utente a decidere quali di questi fare comparire. Infine la presenza di alcuni pulsanti permette di salvare le informazioni, caricarle da un file, resettare le posizioni degli oggetti selezionati o tornare al menu principale.

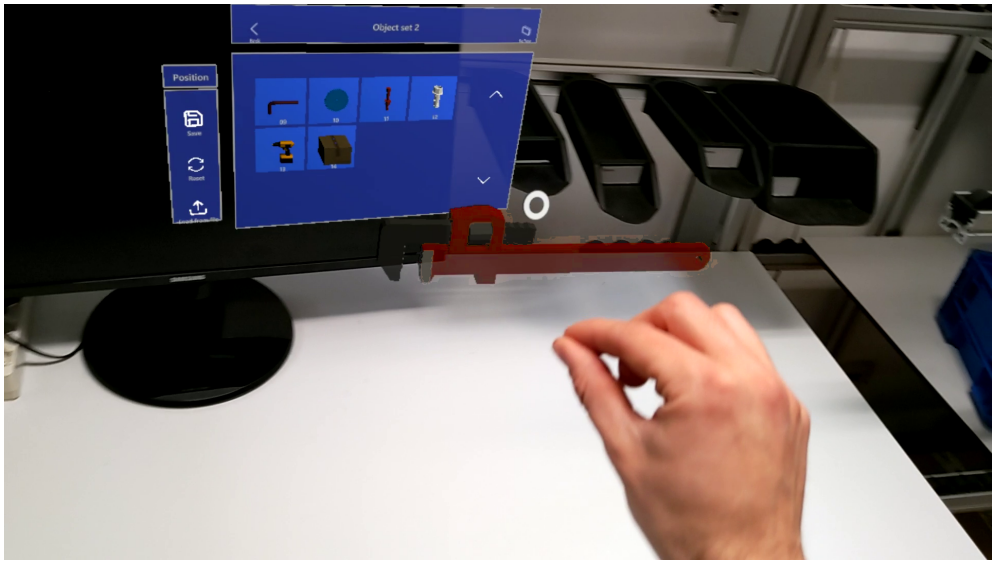

Figura 6.1: Vista dell'applicazione da HoloLens

6.2 Unity

Il programma in Unity è strutturato in due diverse scene. All'avvio l'utente vede la scena del menu principale, dove è presente un pannello con vari pulsanti per selezionare quale set di oggetti vedere. Le altre scene rappresentano ognuna un set di oggetti diverso. Gli oggetti sono modelli tridimensionali ai quali si associano proprietà sottoforma di script e componenti interni di Unity. Lo sviluppo del progetto è possibile grazie al pacchetto MRTK (Mixed Reality Toolkit) [74], che fornisce una serie di componenti base per accelerare lo sviluppo delle applicazioni di realtà mista su Unity. Nel kit di Microsoft sono presenti codici per le interazioni spaziali con gli oggetti e alcune interfacce utente. In questo lavoro di tesi si è utilizzato MRTK e si è preso spunto dalle sue scene di esempio. I pulsanti e le finestre, ad esempio, sono stati forniti dal pacchetto e sono stati adattati agli obiettivi del programma.

I componenti Unity applicati agli oggetti con il quale si deve interagire sono:

- `Transform`: contiene i valori di posizione, rotazione e scala rispettivamente dei tre assi dello spazio.
- `Mesh Filter`: contiene la mesh dell'oggetto, in particolare la forma.
- `Mesh Renderer`: contiene la mesh dell'oggetto, in particolare la texture.
- `Rigidbody`: è il componente relativo alla fisica dell'oggetto. Permette di impostare diversi parametri tra cui la massa dell'oggetto, la velocità di

trascinamento e la velocità angolare, oltre ad attivare/disattivare la gravità e la cinetica.

- `Box Collider`: permette di impostare sull'oggetto un volume che serve per identificare l'effettiva dimensione e quindi permette di afferrarlo per spostarlo.
- `Bounds Control`: visualizza un quadrato attorno agli oggetti per facilitarne la trasformazione.
- `Object Manipulator`: rende gli oggetti spostabili, scalabili e ruotabili utilizzando una o due mani.
- `Near Interaction Grabbable`: permette l'interazione ravvicinata con gli oggetti.
- `Rotation Axis Constraint`: componente per limitare la rotazione di un modello attorno ad uno o più assi.
- `Min Max Scale Constraint`: permette di impostare il valore minimo e massimo di scala dell'oggetto anche quando viene ridimensionato durante l'uso dell'applicazione.
- `Constraint Manager`: permette di impostare vincoli alla trasformazione dei GameObject, tra i quali `Rotation Axis Constraint` e `Min Max Scale Constraint`.
- `Audio Source`: permette di riprodurre suoni quando si interagisce con l'oggetto.

Questi componenti sono visibili applicati a un oggetto nella parte destra della figura d'esempio A.1.

Per questo progetto si è ritenuto opportuno abilitare alcune (e non tutte) delle proprietà fisiche degli oggetti. Si è deciso di rendere tutti gli oggetti manipolabili con la propria massa e volume ma non soggetti alla gravità, perciò fluttuanti davanti all'utente. Questo permette di spostare gli ologrammi con facilità e posizionarli sopra a ripiani o tavoli grazie alla scansione dell'ambiente fatta da HoloLens, come riportato nel capitolo 4.4. Inoltre è stata attivata l'occlusione, cioè la possibilità di "nascondere" gli ologrammi degli oggetti dietro a mobili o sotto ai tavoli. In questo modo si da un effetto più realistico alla scena che vede l'utente [70].

6.3 Script

Durante lo sviluppo di questo progetto sono stati scritti i seguenti script in C#:

- `CoordinatesScript`
- `SwitchScenarioScript`
- `ExitScript`
- `SaveObject`

Ogni scena del progetto, corrispondente ad un set di oggetti, contiene due GameObject chiamati `CoordinatesManager` e `SwitchScenario`, dedicati rispettivamente alla gestione dei dati da salvare e al cambio scena, ovvero al cambio di menu. Ad essi, infatti, sono associati solamente gli script `CoordinatesScript` (codici da 6.1 a 6.9) e `SwitchScenarioScript` (codice 6.11).

CoordinatesScript

È una classe pubblica che contiene la maggior parte del codice del programma e gestisce il salvataggio, il reset e il caricamento della posizione, rotazione e scala di ogni oggetto. Le funzioni di salvataggio e di caricamento dei dati degli oggetti avvengono tramite un file in formato `.json`. CoordinatesScript è pensato per essere associato ad un GameObject, da cui la proprietà `MonoBehaviour`.

All'inizio del programma si includono i namespace necessari tramite la keywork `using`. Successivamente si definisce la classe e al suo interno si dichiarano alcune variabili condivise da più funzioni.

```
18 using System.Collections;
19 using System.Collections.Generic;
20 using UnityEngine;
21 using UnityEngine.UI;
22 using System.IO;
23 using UnityEngine.SceneManagement;
24
25 public class CoordinatesScript : MonoBehaviour
26 {
27    /* dir and filePath contain json directory and file path (
          including file name), respectively.
28     * json is the string file written by the unity json write methods
          .
```

```
29      * jsonArray is the string array read by the unity json read
            methods.
30      */
31     static string dir;
32     static string filePath;
33     private string json;
34     private string[] jsonArray;
35
36     /* totalObjects contains every GameObject in its initial position
            when Start() is executed.
37      * initialData contains initial relevant information for every
            GameObject.
38      * scene is the current active scene.
39      */
40     private GameObject[] totalObjects;
41     private SaveObject[] initialData;
42     Scene scene;
```

Codice 6.1: CoordinatesScript.cs

Nella funzione `Start()`[15] si salvano i dati iniziali di ogni oggetto nel vettore di tipo `SaveObject` (definito a pagina 75) chiamato `initialData` e si scrivono le stringhe contenenti i percorsi per il salvataggio e la lettura di file. Infine, si controlla l'esistenza della cartella dei file ed eventualmente la si crea.

```
45 public void Start()
46 {
47     /* Finds all GameObjects at the beginning of the scene that have
           a Rigidbody component attached
48        * and saves them in an array of GameObjects
49        */
50     Rigidbody[] rigidbodyArrayObject = FindObjectsOfType<Rigidbody>(
           true);
51     totalObjects = new GameObject[rigidbodyArrayObject.Length];
52
53     for (int i = 0; i < rigidbodyArrayObject.Length; i++)
54        totalObjects[i] = rigidbodyArrayObject[i].gameObject;
55
56     // Convert initial GameObjects to SaveObjects
57     initialData = new SaveObject[totalObjects.Length];
58     for(int i = 0; i < totalObjects.Length; i++)
59        initialData[i] = GOToSO(totalObjects[i]);
60
```

[15] Viene eseguita all'avvio di ogni scena.

```
61      // Writing directory and file path as strings
62      dir = Application.streamingAssetsPath + "/SavedData/";
63      scene = SceneManager.GetActiveScene();
64      filePath = Application.streamingAssetsPath + "/SavedData/
            DataObjectSet" + scene.buildIndex + ".json";
65
66      // Check if the directory dir exists, otherwise create it
67      if (!Directory.Exists(dir))
68          Directory.CreateDirectory(dir);
69  }
```

Codice 6.2: CoordinatesScript.cs

La funzione `SaveToJson()` gestisce il salvataggio su file di tutte le posizioni, rotazioni e dimensioni dei GameObject attivi. Le informazioni vengono convertite nella classe serializzabile `SaveObject` e il formato del file è `.json`, in cui ogni riga rappresenta un GameObject. `SaveToJson()` viene assegnata al pulsante Save di ogni scena.

```
75  public void SaveToJson()
76  {
77      File.WriteAllText(filePath, string.Empty);
78      GameObject[] activeObjects = GameObject.FindGameObjectsWithTag("
            model");
79
80      SaveObject so;
81      foreach (GameObject go in activeObjects)
82      {
83          so = GOToSO(go);
84          json = JsonUtility.ToJson(so);
85          File.AppendAllText(filePath, json + "\n");
86      }
87  }
```

Codice 6.3: CoordinatesScript.cs

La funzione `Reset()` sposta tutti gli oggetti attivi alla posizione iniziale, ovvero quella in cui compaiono quando vengono selezionati per la prima volta. Questa funzione è assegnata all'omonimo pulsante di ogni scena.

```
94  public void Reset()
95  {
96      GameObject[] activeObjects = GameObject.FindGameObjectsWithTag("
            model");
97
```

```
98      if(activeObjects.Length != 0)
99      {
100         GameObject[] resetObjects = new GameObject[initialData.Length
                ];
101
102         for (int i = 0; i < initialData.Length; i++)
103            resetObjects[i] = SOToGO(initialData[i]);
104
105         UpdateExistingObjects(resetObjects, activeObjects);
106      }
107      else
108      {
109         Debug.LogError("In Reset(): there are no active objects to
                reset.");
110      }
111   }
```

Codice 6.4: CoordinatesScript.cs

La funzione `LoadFromJson()` controlla l'esistenza di un file `json` da leggere e, quando questo esiste, ne estrapola le informazioni e le carica nella scena. La lettura avviene riga per riga e dunque richiede che il file sia specificatamente nel formato corretto. Per migliorare il progetto e renderlo più flessibile, in futuro si potrebbero aggiungere controlli sul formato del file prima della lettura, in modo da accettare anche documenti `json` leggermente diversi. `LoadFromJson()` è associata al pulsante Load di ogni scena.

```
118   public void LoadFromJson()
119   {
120      if(File.Exists(filePath) && new FileInfo(filePath).Length != 0)
121      {
122         /* Separates every line from json file and saves them in
                jsonArray
123            * Then converts from json to SaveObject each line and
                   finally from
124            * SaveObject to GameObject, all in one line.
125            */
126         jsonArray = File.ReadAllLines(filePath);
127         GameObject[] newGameObj = new GameObject[jsonArray.Length];
128
129         for (int i = 0; i < jsonArray.Length; i++)
130            newGameObj[i] = SOToGO(JsonUtility.FromJson<SaveObject>(
                   jsonArray[i]));
131
```

```
132         UpdateExistingObjects(newGameObj, totalObjects);
133     }
134     else
135     {
136         if(File.Exists(filePath))
137             Debug.LogError("Save file is empty.");
138         else
139             Debug.LogError("Save file does not exist.");
140     }
141 }
```

Codice 6.5: CoordinatesScript.cs

La funzione `UpdateExistingObjects()` viene chiamata all'interno di `Reset()` e `LoadFromJson()` e consente di aggiornare le posizioni, rotazioni e dimensioni degli oggetti presenti in una lista. Essa, infatti, riceve come parametri un vettore `newList` di GameObject con le nuove posizioni da caricare sul vettore di GameObject già esistenti `oldList`.

```
147 public void UpdateExistingObjects(GameObject[] newList, GameObject[]
        oldList)
148 {
149     int targetIndex;
150 
151     for(int i = 0; i < newList.Length; i++)
152     {
153         targetIndex = FindExistingObject(oldList, newList[i].name);
154         if(targetIndex != oldList.Length)
155         {
156             oldList[targetIndex].transform.localPosition = newList[i].
                    transform.localPosition;
157             oldList[targetIndex].transform.localRotation = newList[i].
                    transform.localRotation;
158             oldList[targetIndex].transform.localScale = newList[i].
                    transform.localScale;
159 
160             Physics(oldList[targetIndex]);
161         }
162         // Destroys old objects from the variables of this script that
                are not used anymore
163         Destroy(newList[i]);
164     }
165 }
```

Codice 6.6: CoordinatesScript.cs

`FindExistingObject()` è una funzione che ricerca un GameObject da un vettore anch'esso di tipo GameObject. I suoi parametri sono dunque il vettore e la stringa contenente il nome da cercare. Se nel vettore c'è un oggetto con il nome che si ricerca, la funzione restituisce l'indice corrispondente, altrimenti restituisce un numero pari alla lunghezza del vettore. È consigliato leggere il valore di ritorno della funzione e controllare subito se questo sia pari alla lunghezza del vettore, indicando quindi che l'oggetto non è stato trovato.

```
168 public int FindExistingObject(GameObject[] list, string target)
169 {
170     int i;
171     for(i = 0; i < list.Length; i++)
172     {
173         if (list[i].name == target)
174             return i;
175     }
176     return i;
177 }
```

Codice 6.7: CoordinatesScript.cs

`Physics()` richiede un GameObject come parametro e, dopo averlo attivato, ne resetta la velocità lineare e angolare.

```
180 public void Physics(GameObject OBJ)
181 {
182     OBJ.SetActive(true);
183 
184     OBJ.GetComponent<Rigidbody>().velocity = Vector3.zero;
185     OBJ.GetComponent<Rigidbody>().angularVelocity = Vector3.zero;
186 }
```

Codice 6.8: CoordinatesScript.cs

Infine, sono state scritte le seguenti funzioni di conversione tra diversi tipi:

- `GOToSO()`: riceve un GameObject e restituisce il SaveObject corrispondente.
- `SOToGo()`: riceve un SaveObject, crea un GameObject con solamente i dati ricevuti e lo restituisce.
- `Vector3ToArray()`: riceve una copia di un Vector3 (tipo specifico di Unity) e restituisce un vettore di tre float con gli stessi valori.

- `ArrayToVector3()`: riceve una copia di un vettore di tre float e restituisce un Vector3 con gli stessi valori.
- `QuaternionToArray()`: riceve una copia di un Quaternion (tipo specifico di Unity) e restituisce un vettore di quattro float con gli stessi valori.
- `ArrayToQuaternion()`: riceve una copia di un vettore di quattro float e restituisce un Quaternion con gli stessi valori.

```
188 // Reads GameObject data and returns SaveObject.
189 public SaveObject GOToSO(GameObject go)
190 {
191     SaveObject so = new SaveObject();
192     so.name = go.name;
193     so.position = Vector3ToArray(go.transform.localPosition);
194     so.rotation = QuaternionToArray(go.transform.localRotation);
195     so.scale = Vector3ToArray(go.transform.localScale);
196 
197     return so;
198 }
199 
200 // Reads SaveObject and creates GameObject with this info. Then
        returns GameObject
201 public GameObject SOToGO(SaveObject so)
202 {
203     GameObject go = new GameObject();
204     go.name = so.name;
205     go.transform.localPosition = ArrayToVector3(so.position);
206     go.transform.localRotation = ArrayToQuaternion(so.rotation);
207     go.transform.localScale = ArrayToVector3(so.scale);
208 
209     return go;
210 }
211 
212 // Converts from Vector3 to float array and returns array
213 public float[] Vector3ToArray(Vector3 vector)
214 {
215     float[] array = new float[3];
216     array[0] = vector.x;
217     array[1] = vector.y;
218     array[2] = vector.z;
219 
220     return array;
221 }
```

```
222
223 // Converts from float array to Vector3 and returns Vector3
224 public Vector3 ArrayToVector3(float[] array)
225 {
226     Vector3 vector = new Vector3();
227     vector.x = array[0];
228     vector.y = array[1];
229     vector.z = array[2];
230
231     return vector;
232 }
233
234 // Converts from Quaternion to float array and returns array
235 public float[] QuaternionToArray(Quaternion quat)
236 {
237     float[] array = new float[4];
238     array[0] = quat.x;
239     array[1] = quat.y;
240     array[2] = quat.z;
241     array[3] = quat.w;
242
243     return array;
244 }
245
246 // Converts from float array to Quaternion and returns Quaternion
247 public Quaternion ArrayToQuaternion(float[] array)
248 {
249     Quaternion quat = new Quaternion();
250     quat.x = array[0];
251     quat.y = array[1];
252     quat.z = array[2];
253     quat.w = array[3];
254
255     return quat;
256 }
```

Codice 6.9: CoordinatesScript.cs

SaveObject

Questa classe pubblica definisce il formato di dati serializzabili. Per i nostri scopi, vengono salvati i valori di posizione, rotazione e scala di ogni singolo oggetto.

```
1 [System.Serializable]
```

```
2
3  // Defines serializable data format for relevant properties of
       GameObjects
4  public class SaveObject
5  {
6     public string name;
7     public float[] position;
8     public float[] rotation;
9     public float[] scale;
10 }
```

Codice 6.10: SaveObject.cs

SwitchScenarioScript

Questa classe pubblica permette al progetto di cambiare scena quando vengono premuti i tasti che la richiamano. È stata pensata per essere associata a un GameObject che gestisce il cambio scene.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  // Allows scene changes during execution, has to be assigned to a
       GameObject (button)
7  public class SwitchScenarioScript : MonoBehaviour
8  {
9     public void ChangeScene(string sceneName)
10    {
11       SceneManager.LoadScene(sceneName);
12    }
13 }
```

Codice 6.11: SwitchScenarioScript.cs

ExitScript

Questa classe pubblica contiene il codice per terminare l'esecuzione dell'applicazione quando viene premuto il tasto Close nella scena del menu principale.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
```

```
4
5  // Script to exit the application, needs to be assigned to a button
6  public class ExitScript : MonoBehaviour
7  {
8     public void Exit()
9     {
10       Application.Quit();
11    }
12 }
```

Codice 6.12: ExitScript.cs

6.4 Salvataggio file

Quando il progetto Unity viene eseguito su un computer, i file generati dal programma vengono salvati nel percorso interno al progetto: `Assets/StreamingAssets/SavedData` e nel nome presentano il numero della scena. È importante sottolineare che una volta compilato il progetto e caricato il file eseguibile sul visore, questo percorso non è più significativo e i file vanno cercati nel percorso dato dalla stringa `Application.streamingAssetsPath` [75].

Il salvataggio su file di dati rilevanti viene effettuato attraverso un processo di serializzazione in formato `.json` (codice 6.13). La scelta dell'utilizzo della serializzazione, pur non strettamente necessaria per il lavoro di questa tesi, deve essere vista come un'impostazione strutturale e un'opportunità per chiunque volesse migliorare e ampliare il progetto. Le prime versioni dell'applicazione, infatti, prevedevano la scrittura di un semplice file di testo, dove le informazioni erano presentate in formato sequenziale. Si è deciso di modificare il progetto e utilizzare la serializzazione perché essa permette di salvare (e successivamente di leggere) le informazioni degli oggetti velocemente in un formato riconosciuto nel settore delle applicazioni Web, utilizzando sia apposite funzioni interne di Unity sia semplici script scritti in occasione di questo lavoro.

```
1 {"name":"Wrench","position":[10.0,8.5,0.0],"rotation
     ":[0.0,0.0,0.0,1.0],"scale":[15.0,15.0,15.0]}
2 {"name":"72T-Gear","position":[12.0,8.5,0.0],"rotation
     ":[0.0,0.0,0.0,1.0],"scale":[10.0,10.0,10.0]}
3 {"name":"Pliers","position":[12.0,6.0,0.0],"rotation
     ":[-0.707106,0.0,0.0,0.707106],"scale":[5.0,5.0,5.0]}
```

Codice 6.13: DataObjectSet1.json

Figura 6.2: Esempio di posizionamento degli oggetti

Capitolo 7

Conclusioni

La realtà virtuale, ad oggi sicuramente la più diffusa, differisce notevolmente da quella aumentata e mista. Quest'ultime infatti permettono di integrare il mondo digitale con quello reale, tramite degli ologrammi, senza dover simulare un ambiente totalmente astratto, come nel caso della realtà virtuale.

L'elaborato si concentra principalmente sul visore HoloLens 2 di Microsoft, tuttavia il mercato offre altri prodotti più o meno ingombranti, che sono stati progettati per uso ludico o professionale. Alcuni restano solo dei progetti mai realizzati, altri sono stati commercializzati ma non su larga scala, dato il particolare tipo di tecnologia.

Osservando nel dettaglio il prodotto di Microsoft, è stato possibile notare diverse caratteristiche peculiari che lo differenziano sia dal modello precedente, sia da altri visori in commercio. Partendo dal display, che possiede una particolare tecnologia waveguide per la visualizzazione degli ologrammi, fino ad arrivare all'ergonomia, appositamente studiata per permettere un eccellente bilanciamento, HoloLens 2 è un insieme di nuove tecnologie adottate per risolvere molti limiti presenti nel mondo dei visori a realtà aumentata.

Sono già presenti molti strumenti software che permettono di sviluppare applicativi per visori. In particolare si fa riferimento a Unity come IDE di sviluppo grafico.

Si può prevedere un forte sviluppo della tecnologia dei visori a realtà aumentata e mista, soprattutto in un contesto lavorativo. Questo perché offrono una valida alternativa ad attività quali: creazione di mockup per testare i prodotti, simulazioni in campo medico o industriale, e molte altre, che rappresentano spesso grossi costi per l'azienda. Inoltre può fornire importanti ausili per semplificare diverse mansioni altrimenti complicate.

Per la stesura di questo elaborato è stata sviluppata un'applicazione, pienamente funzionante, che permette di simulare un banco da lavoro in una catena di montaggio o assemblaggio. Questa applicazione è un'ottima base per future migliorie e integrazioni, sia dal lato hardware che software.

Infatti successivamente è stata sviluppata la seconda versione di questa applicazione, più mirata all'ambito industriale introducendo oggetti più realistici. Inoltre, permette il salvataggio e il caricamento delle posizioni degli oggetti sfruttando la serializzazione e la scrittura su file `.json` modificabili in un secondo momento dall'utente. Durante lo sviluppo sono state riscontrate alcune difficoltà con l'utilizzo del motore di gioco Unity e con il caricamento dell'applicazione su HoloLens. In particolare, le informazioni in rete su come sviluppare un'applicazione per questo visore sono ancora limitate. Il programma, però, risulta funzionante e si appresta a future modifiche e miglioramenti.

Attualmente l'applicazione presenta due limiti di utilizzo: l'impossibilità di fare comparire due copie dello stesso oggetto e l'impossibilità di disattivare un modello se non si vuole salvare la sua posizione.

Inoltre, la visualizzazione sul visore dei file di salvataggio risulta tuttora problematica. Il sistema operativo Windows Holographic OS, infatti, non permette di visualizzare le cartelle delle varie applicazioni installate e quindi non è possibile cercare i file di salvataggio. Detto ciò, le funzioni di salvataggio e caricamento all'interno dell'applicazione funzionano correttamente e i file sono visibili tramite computer quando si simula l'applicazione con Unity.

I file dell'applicazione di questa tesi sono reperibili al link `https://github.com/pettepiero/AR_Workspace_Builder` in maniera libera cosicché chiunque possa apportare modifiche e ampliarne il campo di utilizzo.

Appendice A

Importare modelli in Unity

Questo progetto prevede l'importazione di modelli tridimensionali di oggetti reali, allo scopo di proiettare l'utente in uno scenario virtuale quanto più reale possibile. Il motore di gioco Unity utilizzato per la creazione di questa applicazione consente l'importazione di modelli 3D anche di diversi formati [76]. In genere i più utilizzati sono file in formato `.obj`, `.fbx` e `.prefab`. Questi formati di file servono per definire le geometrie 3D: in particolare il formato `.obj` è di tipo open-source, il formato `.fbx` è di proprietà di Autodesk, mentre il formato `.prefab` è il più adatto a Unity perchè consente di salvare tutte le proprietà e i componenti del GameObject, ovvero il modello geometrico definito oggetto di gioco in Unity.

Per questo progetto sono stati utilizzati file di modelli nel formato `.fbx` con licenza d'uso gratuita, in quanto contengono al loro interno anche le mesh dell'oggetto, e alcuni modelli `.prefab` scaricati dal Unity Asset Store, perché sono i modelli più semplici da importare e aventi migliori mesh. Per quanto riguarda i file in formato `.obj`, non è stato possibile utilizzarli perché non contenevano le mesh del rispettivo modello. Si rimanda ad un futuro approfondimento per l'utilizzo di questo formato di file.

Il procedimento per importare i file dei modelli nell'ambiente di gioco è breve:

1. Scaricare il file e copiarlo all'interno di una cartella del progetto di Unity; il percorso del progetto si può trovare dal manager Unity Hub nella scheda `Projects`. È preferibile creare una nuova cartella e non utilizzare quelle già presenti.

2. Aprire il progetto di Unity e il file del modello verrà letto e importato in automatico. Nel tab `Projects` → `Assets` dell'editor comparirà la nuova cartella creata contenente i file dei modelli.

3. Trascinare il file del modello nella scena aperta dal tab `Hierarchy`.

4. Selezionare l'oggetto importato nella scena per modificare i suoi parametri e aggiungere componenti attraverso il tab `Inspector`.

   In caso di errore quando si modificano i parametri del modello, assicurarsi che esso abbia nome e icona bianche nel tab `Hierarchy`, altrimenti in caso di nome e icona azzurre premere tasto destro e selezionare `Prefab` → `Unpack Completely`.

Il passo 1 e 2 verranno effettuati in automatico se si scaricano i modelli dal Unity Asset Store perché attraverso il sito, dopo l'acquisto, si aprirà in automatico un pop-up che chiederà di aggiungere il pacchetto nel progetto attualmente aperto. Basterà poi scaricare il pacchetto acquistato tramite il tab `Package Manager` che si aprirà e infine premere il tasto Import di fianco al tasto Download precedentemente premuto. Verrà creata in automatico la cartella all'interno del progetto e in seguito sarà possibile proseguire con il passo 3.

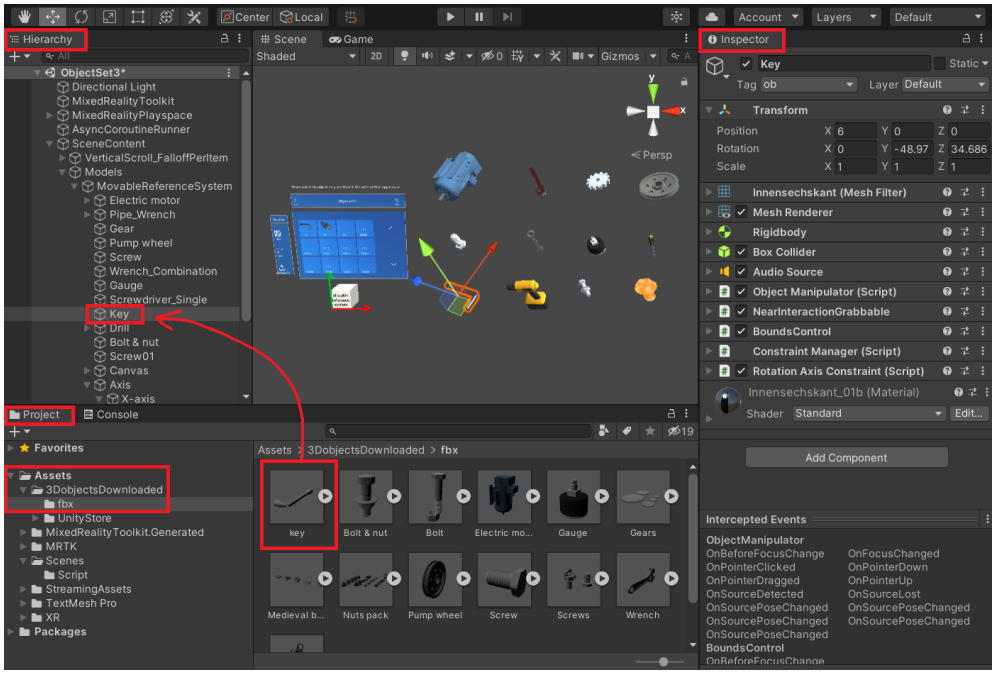

Figura A.1: Importazione di un modello in formato .fbx nell'editor Unity

Appendice B

Compilazione e caricamento dell'applicazione su HoloLens 2

Per compilare e caricare il progetto di Unity su HoloLens sono necessarie alcune impostazioni.

In Unity, tramite la finestra `File → Build Settings` selezionare `Universal Windows Platform` e premere il tasto `Switch Platform` in basso per cambiare la piattaforma di sviluppo. Premere il tasto `Add Open Scene` per aggiungere singolarmente ogni scena di gioco. In questo passaggio è importante aggiungere tutte le scene del proprio progetto, altrimenti quelle non aggiunte non compariranno nell'applicazione finale.

Dopodiché impostare i vari parametri di `Universal Windows Platform` come da figura B.1. Si sottolinea che `Target Device` deve essere impostato su `Any device` e `Architecture` su `x64` in quanto il progetto di Unity verrà inizialmente compilato per questa architettura in modo che possa essere letto correttamente da Visual Studio, che è eseguito sul proprio computer in genere con architettura `x64`. Dopodiché il programma verrà ricompilato una seconda volta con Visual Studio per l'architettura `ARM64` di HoloLens 2. Il parametro `Build Configuration` settato su `Release` serve per evitare errori durante la compilazione con Visual Studio, come descritto nella documentazione fornita da Microsoft [77]. Tutti gli altri parametri dovrebbero essere già impostati di default come in figura B.1.

Infine premendo il tasto `Build`, che ha sostituito il precedente `Switch Platform`, verrà effettivamente compilato il progetto e al termine si aprirà la cartella di destinazione scelta con all'interno il file solution[16] di Visual Studio.

[16]File con estensione estensione `.sln`

Questo file solution è necessario aprirlo con il programma Visual Studio per poter caricare il progetto compilato nel dispositivo HoloLens.

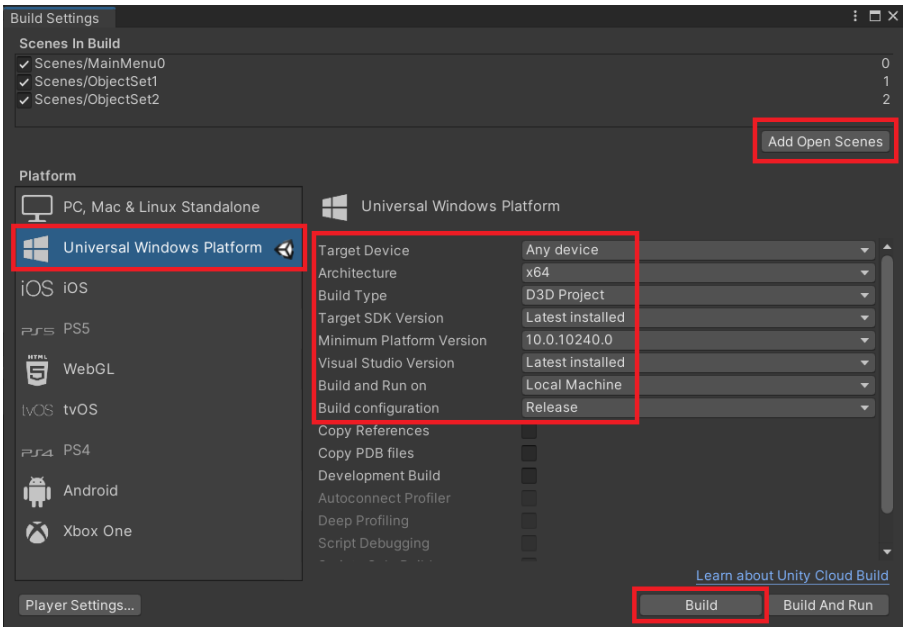

Figura B.1: Impostazioni per compilare il progetto in Unity

A questo punto, prima di effettuare l'effettivo caricamento dell'applicazione su HoloLens, è necessario impostare il sistema operativo Windows del computer in modalità sviluppatore: ovvero basta recarsi in `Impostazioni` → `Aggiornamento e sicurezza` → `Per sviluppatori`. Un procedimento simile è necessario farlo nel HoloLens andando in `Settings` → `Update & Security` → `For developers` e attivare la modalità sviluppatore con il primo switch che si trova in alto, attivare `Device discovery` con il secondo switch e infine attivare anche `Device portal` con il terzo e ultimo switch che si trova in questa finestra; come mostrato in figura B.4.

Tornando in Visual Studio, il caricamento dell'applicazione può essere effettuato in due modi: il primo mediante cavo USB richiede meno passaggi, invece il secondo mediante rete Wi-Fi è più comodo durante la fase di sviluppo ma richiede qualche impostazione aggiuntiva[17]. Per entrambi i metodi è necessario impostare dal primo menu a tendina la stessa modalità selezionata in `Build configuration` di Unity (ovvero `Release`) e dal secondo menu a tendina l'architettura di HoloLens 2 ovvero `ARM64`, come mostrato in figura B.2.

[17]Per un approfondimento si invita alla lettura della documentazione di Microsoft [78].

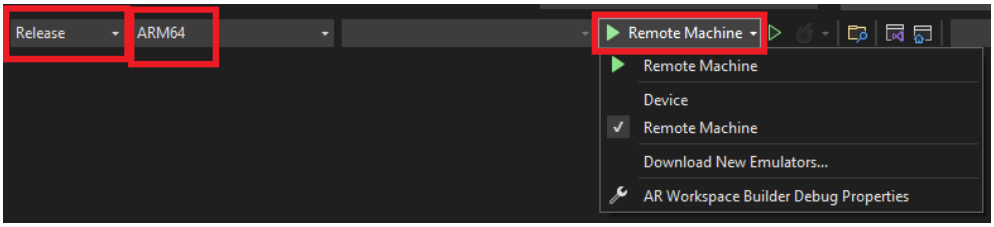

Figura B.2: Impostazioni per compilare il progetto in Visual Studio

1. USB

   Collegare HoloLens al computer mediante cavo USB. Dal terzo menu a tendina selezionare `Device`, come mostrato in figura B.2. Poi premere la freccia verde per avviare il debug con il caricamento dell'applicazione su HoloLens.

2. Wi-Fi

   Dal terzo menu a tendina selezionare `Remote Machine`, come mostrato in figura B.2. Andare in `Project` → `Properties`. Nella finestra che si apre assicurarsi di avere come configurazione attiva quella precedentemente selezionata, ovvero `Release`, e piattaforma `ARM64`. Tramite pannello laterale selezionare la voce `Debugging` e controllare che sia selezionato `Remote Machine`. Poi inserire l'indirizzo IP del dispositivo HoloLens nel campo `Machine Name`, come da figura B.3.

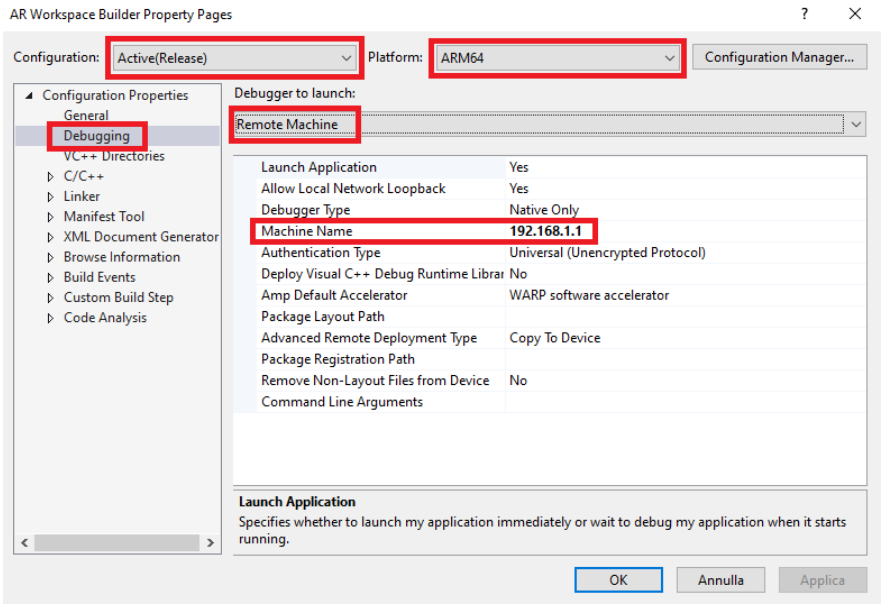

Figura B.3: Parametri da impostare per compilare e caricare il progetto con Wi-Fi

È necessario che HoloLens sia collegato alla stessa rete Wi-Fi del computer usato per il caricamento dell'applicazione. L'indirizzo IP di HoloLens può essere trovato al suo interno in `Settings` → `Update & Security` → `For developers` in fondo alla pagina dopo aver precedentemente abilitato `Device Portal`, come mostrato in figura B.4.

Infine premere `Ok` della finestra aperta e premere la freccia verde dalle opzioni in alto nell'interfaccia di Visual Studio per avviare il debug con il caricamento dell'applicazione su HoloLens.

Quanto appena descritto serve per eseguire l'attività di debug dell'applicazione con conseguente caricamento della stessa sul dispositivo. Invece, se si vuole solamente caricare l'applicazione senza eseguire il debug, basterà premere `Build` → `Deploy solution` dalle opzioni in alto nell'interfaccia di Visual Studio.

Quando avviene il caricamento dell'applicazione su HoloLens, la prima volta sarà necessario associare il visore al computer. Visual Studio mostrerà una finestra dove chiede di inserire in codice PIN. Questo codice si trova all'interno di HoloLens andando in `Settings` → `Update & Security` → `For developers`, come mostrato in figura B.4. Premendo il tasto `Pair` verrà mostrato il codice PIN da inserire nella finestra di Visual Studio, così il computer sarà associato e il caricamento dell'applicazione potrà andare a buon fine.

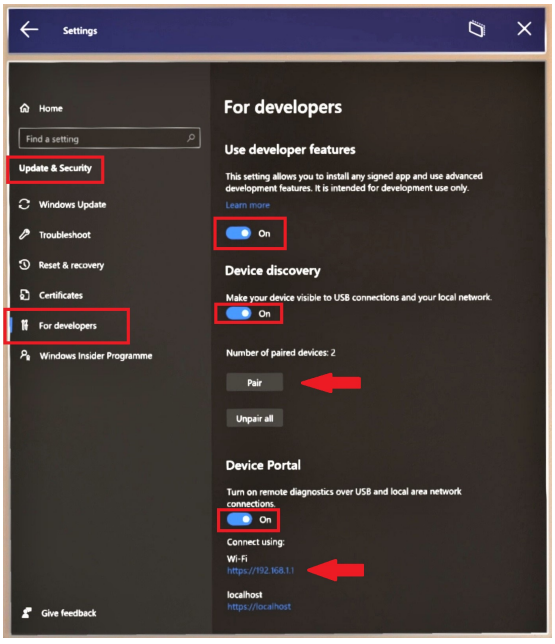

Figura B.4: Finestra impostazioni visualizzata tramite HoloLens

Bibliografia

Capitolo 2

[1] Antonio Laudazi. *Niente sarà più come prima*. 1ª ed. Dario Flaccovio Editore, 2019 (cit. alle pp. 3, 9).

[2] Paul Milgram e Fumio Kishino. «A taxonomy of mixed reality visual displays». In: *IEICE TRANSACTIONS on Information and Systems* 77.12 (1994), pp. 1321–1329 (cit. alle pp. 4–6).

[3] Ben D. Lawson e Kay M. Stanney. «Editorial: Cybersickness in Virtual Reality and Augmented Reality». In: *Frontiers in Virtual Reality* 2 (2021), p. 131 (cit. a p. 10).

[4] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality*. Association for Computing Machinery e Morgan & Claypool, 2015 (cit. alle pp. 10–12, 14).

[5] David M. Hoffman et al. «Vergence–accommodation conflicts hinder visual performance and cause visual fatigue». In: *Journal of vision* 8.3 (2008), pp. 33–33 (cit. a p. 13).

[6] Alla Vovk et al. «Simulator Sickness in Augmented Reality Training Using the Microsoft HoloLens». In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018 (cit. a p. 15).

Capitolo 3

[7] Graciela Guerrero, Lenin Gómez e José Achig. «Holonote: Text editor of augmented reality oriented to people with motor disabilities.» In: *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE. 2020, pp. 1–7 (cit. a p. 21).

[8] Beatrice Aruanno, Franca Garzotto e Mario Covarrubias Rodriguez. «MemHolo: mixed reality experiences for subjects with Alzheimer's disease». In: *Multimedia Tools and Applications*. 2019, pp. 1–9 (cit. a p. 21).

[9] Corinne Wyss et al. «Innovative Teacher Education with the Augmented Reality Device Microsoft HoloLens—Results of an Exploratory Study and Pedagogical Considerations». In: *Multimodal Technologies and Interaction* 5.8 (2021), p. 45 (cit. a p. 22).

[10] David A. Moses et al. «Real-time decoding of question-and-answer speech dialogue using human cortical activity». In: *Nature communications* 10.1 (2019), pp. 1–14 (cit. a p. 30).

Capitolo 4

[11] Yao Zhou, Jufan Zhang e Fengzhou Fang. «Advances in the design of optical see-through displays». In: *Advanced Optical Technologies* -1 (giu. 2020) (cit. a p. 35).

[12] Linden Ball. «Eye-tracking and reasoning: What your eyes tell about your inferences». In: nov. 2013 (cit. a p. 40).

[13] Colm Slattery e Yuzo Shida. «ADI ToF Depth Sensing Technology: New and Emerging Applications in Industrial, Automotive Markets, and More». In: *41 Understanding the Fundamentals of Earthquake Signal Sensing Networks* (2019), p. 52 (cit. alle pp. 40, 41).

[14] Larry Li et al. «Time-of-flight camera–an introduction». In: *Technical white paper* SLOA190B (2014) (cit. a p. 42).

[15] Miles Hansard et al. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer Science & Business Media, 2012 (cit. a p. 43).

Sitografia

Capitolo 2

[16] Diego Barbera. *Live View, su Google Maps arriva la realtà aumentata*. A cura di Wired. 8 Ago. 2019. URL: https://www.wired.it/mobile/app/2019/08/08/google-live-view-maps-novita (cit. a p. 3).

[17] LightField Forum, cur. *What is the Light Field?* URL: http://lightfield-forum.com/what-is-the-lightfield (cit. a p. 13).

Capitolo 3

[18] Apple. *App Measure*. URL: https://support.apple.com/en-us/HT208924 (cit. a p. 17).

[19] Ikea. *App Ikea Place*. URL: https://www.ikea.com/au/en/customer-service/mobile-apps/say-hej-to-ikea-place-pub1f8af050 (cit. a p. 17).

[20] Warby Parker. *App Warby Parker*. URL: https://www.warbyparker.com/app (cit. a p. 17).

[21] Wanna Fashion. *App Wanna Kicks*. URL: https://wanna.fashion (cit. a p. 17).

[22] Paul Tassi. *'Pokémon GO' Has Made 5 Billion In Five Years*. A cura di Forbes. 6 Lug. 2021. URL: https://www.forbes.com/sites/paultassi/2021/07/06/pokmon-go-has-made-5-billion-in-five-years (cit. a p. 17).

[23] Emiliano Ragoni. *Realtà aumentata in auto, il parabrezza diventa un maxi display*. A cura di Wired. 17 Dic. 2019. URL: https://www.wired.it/gadget/motori/2019/12/17/realta-aumentata-in-auto-il-parabrezza-diventa-un-maxi-display/ (cit. a p. 18).

[24] Boeing. *Boeing Tests Augmented Reality in the Factory*. 19 Gen. 2018. URL: https://www.boeing.com/features/2018/01/augmented-reality-01-18.page (cit. a p. 18).

[25] Airbus. *Airbus develops world's first Mixed Reality Trainer for A350 XWB*. 14 Nov. 2017. URL: https://www.airbus.com/newsroom/press-releases/en/2017/11/airbus-develops-worlds-first-mixed-reality-trainer-for-a350-xwb.html (cit. a p. 18).

[26] Scott Erickson. *Microsoft HoloLens enables thyssenkrupp to transform the global elevator industry*. A cura di Windows Blog. 15 Set. 2016. URL: https://blogs.windows.com/devices/2016/09/15/microsoft-hololens-enables-thyssenkrupp-to-transform-the-global-elevator-industry (cit. a p. 18).

[27] Sarah Ramsey. *NASA, Microsoft Collaborate to Bring Science Fiction to Science Fact*. A cura di NASA. 7 Set. 2017. URL: https://www.nasa.gov/press-release/nasa-microsoft-collaborate-to-bring-science-fiction-to-science-fact (cit. a p. 18).

[28] Frank Tavares. *NASA's Using Augmented Reality to Transform Air Traffic Management*. A cura di NASA. 25 Nov. 2020. URL: https://www.nasa.gov/feature/ames/augmented-reality-air-traffic-management (cit. a p. 18).

[29] Università di Bologna. *RETINA - Resilient Synthetic Vision for Advanced Control Tower Air Navigation Service Provision*. A cura di unibo.it. URL: https://site.unibo.it/almagoals/en/projects/retina-resilient-synthetic-vision-for-advanced-control-tower-air-navigation-service-provision (cit. a p. 19).

[30] EUROCONTROL. *Leading the way on augmenting air traffic control*. A cura di eurocontrol.int. 15 Feb. 2018. URL: https://www.eurocontrol.int/news/augmenting-air-traffic-control (cit. a p. 19).

[31] Lorenzo Longhitano. *Microsoft vuole usare Hololens per mappare tutto il mondo*. A cura di Wired. 6 Nov. 2017. URL: https://www.wired.it/gadget/computer/2017/11/06/hololens-mappe-3d (cit. a p. 19).

[32] Chen et al. *Augmented Reality*. A cura di United States Patent e Trademark Office. 26 Apr. 2016. URL: https://pdfaiw.uspto.gov/.aiw?PageNum=0&docid=20170053447 (cit. a p. 19).

[33] Microsoft Customer Stories, cur. *Audi: Mixed reality is worth more than a thousand words*. 6 Lug. 2021. URL: https://customers.microsoft.com/en-us/story/1355894238698027163-audi-scxes-reflekt-viscopic-hololens-en (cit. a p. 20).

[34] Jung BIM-Services. *Hololens 2 / RPT600*. A cura di YouTube. 19 Feb. 2021. URL: https://youtu.be/8AUY-BViLxY (cit. a p. 20).

[35] Trimble. *Trimble AR solutions*. URL: https://fieldtech.trimble.com/en/products/mixed-reality (cit. a p. 20).

[36] Lorraine Bardeen. *How mixed reality is changing the game for healthcare, from performing live surgeries to delivering ultrasounds in 3D*. A cura di Windows Blog. 8 Mar. 2018. URL: https://blogs.windows.com/windowsexperience/2018/03/08/how-mixed-reality-is-changing-the-game-for-healthcare-from-performing-live-surgeries-to-delivering-ultrasounds-in-3d (cit. a p. 21).

[37] Benedetta Paoletti. *La realtà aumentata (AR) utilizzata nella chirurgia sostitutiva del ginocchio*. A cura di Close-up Engineering. 28 Gen. 2021. URL: https://biomedicalcue.it/realta-aumentata-ar-chirurgia-sostitutiva-ginocchio (cit. a p. 21).

[38] Alessandro Mastrofini. *La realtà aumentata viene utilizzata per la prima volta per operare la spalla*. A cura di Close-up Engineering. 2 Lug. 2021. URL: https://biomedicalcue.it/realta-aumentata-spalla (cit. a p. 21).

[39] University of Alberta. *ProjectDR*. URL: https://www.ualberta.ca/rehabilitation/research/rehabilitation-robotics/current-projects/projectdr.html (cit. a p. 21).

[40] *AccuVein*. URL: https://www.accuvein.com/why-accuvein/ar (cit. a p. 21).

[41] Roguish. *Windows and HoloLens Heart Rate Monitor Plugin*. 19 Nov. 2018. URL: https://assetstore.unity.com/packages/tools/input-management/windows-and-hololens-heart-rate-monitor-plugin-76113 (cit. a p. 21).

[42] Medivis. *AnatomyX*. URL: https://www.medivis.com/anatomyx (cit. a p. 22).

[43] Microsoft. *Holoportation*. URL: https://www.microsoft.com/en-us/research/project/holoportation-3 (cit. a p. 23).

[44] Microsoft. *Mesh*. URL: https://www.microsoft.com/en-us/mesh (cit. a p. 23).

[45] Deborah Bach. *U.S. Army to use HoloLens technology in high-tech headsets for soldiers*. A cura di Microsoft. 8 Giu. 2021. URL: https://news.microsoft.com/transform/u-s-army-to-use-hololens-technology-in-high-tech-headsets-for-soldiers (cit. a p. 23).

[46] Natuzzi. *The Natuzzi Augmented Store @ NRF New York*. URL: https://www.natuzzi.com/news/the-natuzzi-augmented-store-nrf-new-york-1203.html (cit. a p. 23).

[47] Luca Francescangeli. *Sempre più aziende provano a usare la realtà aumentata in fabbrica*. A cura di Wired. 13 Ott. 2021. URL: https://www.wired.it/economia/start-up/2021/10/13/realta-aumentata-fabbrica (cit. a p. 23).

[48] Andrea Daniele Signorelli. *Davvero i visori per la realtà aumentata sostituiranno i nostri smartphone?* A cura di Wired. 5 Apr. 2021. URL: https://www.wired.it/attualita/tech/2021/04/05/realta-virtuale-visori-smartphone-apple-facebook (cit. a p. 24).

[49] Andrea Daniele Signorelli. *Il sogno infranto della realtà aumentata*. A cura di Wired. 7 Mag. 2020. URL: https://www.wired.it/attualita/tech/2020/05/07/realta-aumentata-flop-magic-leap (cit. a p. 26).

[50] Wikipedia. *Magic Leap*. URL: https://en.wikipedia.org/wiki/Magic_Leap#Acquisitions (cit. a p. 26).

[51] Emanuela Dicosola. *BIM, realtà aumentata e realtà virtuale: quale futuro per l'edilizia?* A cura di Close-up Engineering. 13 Apr. 2019. URL: https://buildingcue.it/bim-realta-aumentata-e-realta-virtuale-quale-futuro-per-ledilizia (cit. a p. 26).

[52] Adi Robertson. *AR headset company Meta shutting down after assets sold to unknown company*. A cura di The Verge. 18 Gen. 2019. URL: https://www.theverge.com/2019/1/18/18187315/meta-vision-ar-

headset-company-asset-sale-unknown-buyer-insolvent (cit. a p. 27).

[53] Adi Robertson. *Augmented reality headset company Daqri is reportedly shutting down*. A cura di The Verge. 13 Set. 2019. URL: https://www.theverge.com/2019/9/13/20864556/daqri-ar-headset-smart-glasses-startup-shutdown-asset-sale-layoffs (cit. a p. 27).

[54] Alessio Caprodossi. *C'è un nuovo visore per la realtà aumentata e virtuale*. A cura di Wired. 8 Ott. 2021. URL: https://www.wired.it/gadget/videogiochi/2021/10/08/lynxr1-visore-realta-aumentata-realta-virtuale (cit. a p. 28).

[55] Qualcomm. *Everything you need to know about 5G*. URL: https://www.qualcomm.com/5g/what-is-5g (cit. a p. 29).

[56] Alessia Camera. *Realtà virtuale e aumentata, ecco come la usano le aziende*. A cura di Wired. 6 Ago. 2019. URL: https://www.wired.it/economia/business/2019/08/06/realta-virtuale-aumentata-aziende (cit. a p. 29).

[57] Redazione Close-up Engineering. *La Realtà Virtuale ed Aumentata per la Formazione 4.0*. A cura di Close-up Engineering. 15 Nov. 2016. URL: https://managementcue.it/realta-virtuale-e-aumentata-formazione-professionale-4-0 (cit. a p. 29).

[58] Sean Endicott. *How Microsoft HoloLens is helping surgeons work together across thousands of miles*. A cura di Windows Central. 10 Feb. 2021. URL: https://www.windowscentral.com/how-hololens-helping-surgeons-work-together-across-thousands-miles (cit. a p. 29).

[59] Andrea Daniele Signorelli. *Il sogno infranto della realtà aumentata*. A cura di Wired. 7 Mag. 2020. URL: https://www.wired.it/attualita/tech/2020/05/07/realta-aumentata-flop-magic-leap (cit. a p. 30).

[60] Gabriele La Greca. *Un wearable in grado di leggere il pensiero? Nuovi progressi da Facebook*. A cura di Close-up Engineering. 1 Ago. 2019. URL: https://systemscue.it/wearable-leggere-pensiero-facebook (cit. a p. 30).

Capitolo 4

[61] Microsoft. *About HoloLens 2*. 2 Nov. 2021. URL: https://docs.microsoft.com/en-us/hololens/hololens2-hardware (cit. alle pp. 31, 32, 48).

[62] Microsoft. *About HoloLens 2: Device capabilities*. 2 Nov. 2021. URL: https://docs.microsoft.com/en-us/hololens/hololens2-hardware#device-capabilities (cit. a p. 31).

[63] Qualcomm. *Snapdragon 850 Mobile Compute Platform*. URL: https://www.qualcomm.com/products/snapdragon-850-mobile-compute-platform (cit. a p. 32).

[64] Wikipedia. *ARM architecture*. URL: https://en.wikipedia.org/wiki/ARM_architecture (cit. a p. 32).

[65] Microsoft. *HoloLens 2 release notes*. 14 Ott. 2021. URL: https://docs.microsoft.com/en-us/hololens/hololens-release-notes (cit. a p. 33).

[66] Wikipedia. *Punto prossimo*. URL: https://it.wikipedia.org/wiki/Punto_prossimo (cit. a p. 34).

[67] Karl Guttag. *AR/MR Combiners Part 2 – HoloLens*. A cura di kguttag.com. 27 Ott. 2016. URL: https://kguttag.com/2016/10/27/armr-combiners-part-2-hololens (cit. alle pp. 35, 36).

[68] Saarikko et al. *Waveguide*. A cura di United States Patent e Trademark Office. 9 Feb. 2015. URL: https://pdfaiw.uspto.gov/.aiw?PageNum=0&docid=20160231568 (cit. alle pp. 36–38).

[69] Bryn Farnsworth. *What is Eye Tracking and How Does it Work?* A cura di imotions.com. 2 Apr. 2019. URL: https://imotions.com/blog/eye-tracking-work (cit. a p. 40).

[70] Microsoft. *Spatial mapping*. 19 Ott. 2021. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping (cit. alle pp. 42, 67).

[71] Microsoft. *HoloLens (1st gen) hardware*. 30 Ago. 2021. URL: https://docs.microsoft.com/en-us/hololens/hololens1-hardware (cit. a p. 48).

[72] Wikipedia. *Peripheral vision*. URL: https://en.wikipedia.org/wiki/Peripheral_vision#Outer_boundaries (cit. a p. 50).

Capitolo 5

[73] Fondazionergo. *Analisi lavoro*. URL: https://www.fondazionergo.it/chi-siamo/analisi-lavoro (cit. a p. 55).

Capitolo 6

[74] Microsoft. *What is the Mixed Reality Toolkit*. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05 (cit. a p. 66).

[75] Unity Documentation. *Application.streamingAssetsPath*. URL: docs.unity3d.com/ScriptReference/Application-streamingAssetsPath.html (cit. a p. 77).

Appendice A

[76] Unity Documentation. *Supported Model file formats*. URL: https://docs.unity3d.com/2020.1/Documentation/Manual/3D-formats.html (cit. a p. 81).

Appendice B

[77] Microsoft. *Compilation options*. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-visual-studio?tabs=hl2#compilation-options (cit. a p. 83).

[78] Microsoft. *Using Visual Studio to deploy and debug*. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-visual-studio?tabs=hl2 (cit. a p. 84).